UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS REITOR JOÃO DAVID FERREIRA LIMA
PROGRAMA DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Raquel Cristina Schaly Behrens

EXPERT BEE: UM SISTEMA ESPECIALISTA DE APOIO À APRENDIZAGEM INTEGRADA AO MOODLE E BEECROWD

Florianópolis, Santa Catarina – Brasil
2024

Raquel Cristina Schaly Behrens

EXPERT BEE: UM SISTEMA ESPECIALISTA DE APOIO À APRENDIZAGEM INTEGRADA AO MOODLE E BEECROWD

Trabalho de Conclusão de Curso submetido ao Programa de Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharela em Ciência da Computação.
Orientador(a): Prof. Maicon Rafael Zatelli, Dr.
Coorientador(a): Prof. Antonio Carlos Mariani

Florianópolis, Santa Catarina – Brasil
2024

Raquel Cristina Schaly Behrens

EXPERT BEE: UM SISTEMA ESPECIALISTA DE APOIO À APRENDIZAGEM INTEGRADA AO MOODLE E BEECROWD

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharela em Ciência da Computação, e foi aprovado em sua forma final pelo Programa de Graduação em Ciência da Computação do INE – Departamento de Informática e Estatística, CTC – Centro Tecnológico da Universidade Federal de Santa Catarina.

Florianópolis, Santa Catarina – Brasil, 28 de novembro de 2024.

Profa. Lúcia Helena Martins Pacheco, Dra.
Coordenador(a) do Programa de Graduação
em Ciência da Computação

Banca Examinadora:

Prof. Maicon Rafael Zatelli, Dr.
Orientador(a)
Universidade Federal de Santa Catarina

Prof. Antonio Carlos Mariani
Coorientador(a)
Universidade Federal de Santa Catarina

Prof. Álvaro Junio Pereira Franco, Dr.
Avaliador
Universidade Federal de Santa Catarina

Profa. Luciana de Oliveira Rech, Dra.
Avaliadora
Universidade Federal de Santa Catarina

Ao meu querido irmão, Lucas Guilherme Schaly Behrens (in memoriam),
que nos deixou há pouco tempo e faz muita falta. Você marcou profundamente
a vida de todos que tiveram o privilégio de conhecê-lo.
A saudade é eterna.

AGRADECIMENTOS

Agradeço a Deus pela força que me concedeu em toda essa caminhada, pois tudo posso naquele que me fortalece.

Agradeço à minha família, e em especial aos meus pais, Valter Germano Behrens e Ana Lidia Schaly Behrens, e à minha avó Denilce Jung, por possibilitarem toda a minha trajetória acadêmica e pelo amparo incondicional.

Agradeço ao meu namorado, Pedro Augusto Probst Bennemann, por estar ao meu lado desde o início dessa caminhada, me incentivando e apoiando.

Aos meus queridos amigos, Gabriella Schmid Scapini e Jose Daniel Alves do Prado, que sempre estiveram ao meu lado. Agradeço por todos os momentos compartilhados, e que venham muitos outros.

Ao meu orientador, o professor Maicon Rafael Zatelli, agradeço pela atenção, paciência e orientação, que foram essenciais para que o projeto fosse concluído.

À UFSC e a todos os seus professores que sempre proporcionaram um ensino de alta qualidade.

E a todos os meus amigos, que fizeram parte dessa trajetória e que a tornaram mais leve e divertida.

RESUMO

A utilização da tecnologia na educação tem o potencial de fortalecer a relação de ensino e aprendizagem entre educadores e alunos, permitindo o acesso a materiais, cronogramas e fóruns por meio de ambientes virtuais. O Moodle é um exemplo de plataforma virtual que oferece suporte pedagógico a instituições de ensino, e é amplamente utilizado dentre as universidades. Ele, além de simular uma sala de aula online, possui ferramentas de suporte a cursos da área de Tecnologia da Informação e da Computação, como o VPL. O VPL é um módulo integrado ao Moodle e gerenciador de tarefas de programação, mas que apresenta algumas limitações relacionadas à interface e à usabilidade. Em contrapartida, existem plataformas não vinculadas ao Moodle que possuem um juiz online para problemas de programação, como o Beecrowd. O Beecrowd, ao possuir questões prontas de programação, cada uma com baterias de testes, que fornecem feedback em tempo real, permite aos usuários desenvolver habilidades de programação e resolução de problemas. Além disso, essa plataforma desperta um interesse crescente nos ambientes pedagógicos. Isso ocorre, em parte, devido à sua vasta coleção de questões e também por estar disponível em português brasileiro. Isso torna a plataforma acessível a um público mais amplo, especialmente àqueles que estão dando os primeiros passos na área e ainda não possuem proficiência em inglês. Dessa forma, o objetivo deste trabalho consiste em favorecer o uso da plataforma Beecrowd entre docentes, aproveitando a integração LTI já existente entre o Moodle e o Beecrowd, desenvolvida pela equipe da plataforma. Além disso, busca-se disponibilizar uma ferramenta baseada em sistema especialista para auxiliar professores e alunos no esclarecimento de dúvidas recorrentes sobre questões do Beecrowd, facilitando a adaptação de ambos ao uso da plataforma.

Palavras-chaves: beecrowd. moodle. sistema especialista. prolog. swi-prolog. LTI. API. react. typescript. aprendizagem. programação.

ABSTRACT

The use of technology in education has the potential to strengthen the teaching and learning relationship between educators and students, allowing access to materials, timetables and forums through virtual environments. Moodle is an example of a virtual platform that offers pedagogical support to educational institutions, and is widely used among universities. As well as simulating an online classroom, it has tools to support Information Technology and Computing courses, such as VPL. VPL is a Moodle-integrated module and programming task manager, but it has some limitations related to interface and usability. On the other hand, there are platforms not linked to Moodle that have an online judge for programming problems, such as Beecrowd. Beecrowd, by having ready-made programming questions, each with test batteries, which provide real-time feedback, allows users to develop programming and problem-solving skills. In addition, this platform is attracting growing interest in educational environments. This is partly due to its vast collection of questions and also because it is available in Brazilian Portuguese. This makes the platform accessible to a wider audience, especially those who are taking their first steps in the field and are not yet proficient in reading English. Therefore, the aim of this study is to favor the use of the Beecrowd platform among educators, leveraging the existing LTI integration between Moodle and Beecrowd, developed by the platform's team. Additionally, it aims to provide a tool based on an expert system to assist both teachers and students in clarifying recurring doubts about Beecrowd problems, facilitating their adaptation to the platform.

Keywords: beecrowd. moodle. expert system. prolog. swi-prolog. LTI. API. react. typescript. learning. programming.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
AVA	Ambiente Virtual de Aprendizagem
IDE	*Integrated Development Environment*
LMS	*Learning Management System*
LTI	*Learning Tools Interoperability*
RF	Requisito Funcional
RG	Regra de Negócio
RNF	Requisito Não Funcional
SGBD	Sistema Gerenciador de Banco de Dados
URI	Universidade Regional Integrada
VLE	*Virtual Learning Environment*
VPL	*Virtual Programming Lab*

SUMÁRIO

1 INTRODUÇÃO

A utilização da tecnologia no âmbito educacional aumenta o potencial de um vínculo proveitoso entre educadores e alunos, possibilitando a disponibilização de materiais, cronogramas, fóruns, entre outras atividades, por ambientes virtuais, e motivando, assim, os educandos (FRANCISCO; JÚNIOR; AMBRÓSIO, 2016, p. 18-19). Isto posto, há ambientes virtuais de aprendizado que servem de apoio às instituições de ensino no gerenciamento pedagógico dos cursos, como o software livre Moodle – (*Modular Object-Oriented Dynamic Learning Environment*) – Ambiente de Aprendizagem Dinâmico Orientado a Objetos Modulares. O Moodle disponibiliza diversas ferramentas computacionais que proporcionam acesso a materiais didáticos, tarefas interativas e integração entre os membros do curso em que estão matriculados, reproduzindo, dessa forma, uma sala de aula (LIMA, J. M. M., 2021).

Por conseguinte, a fim de proporcionar uma maior assistência à cursos da área de tecnologia da informação, algumas iniciativas de integração de recursos de suporte à disciplinas de programação no ambiente Moodle foram elaboradas, como o VPL (*Virtual Programming Lab*) (FRANÇA; SOARES, 2011, p. 712).

O VPL é um módulo integrável ao Moodle que possibilita o desenvolvimento remoto de programas, permitindo realizar as seguintes atividades no navegador: editar o código-fonte dos programas, executá-los interativamente, realizar testes para revisá-los, definir restrições de edição e evitar colagem de texto externo (RODRÍGUEZ-DELPINO, 2023).

Contudo, Freitas (2016, p. 129) constata que, em sua pesquisa de avaliação do uso do VPL em disciplinas de programação no curso de graduação de Tecnologias de Informação e Comunicação da UFSC, apesar da ferramenta cumprir seu papel de auxílio virtual à prática de programação, ela possui diversos problemas de interface gráfica e de usabilidade. Destacou também falta de mensagens de ajuda apresentadas pela interface ao encontrar-se erros na submissão do código, a obrigatoriedade de casos de testes serem inseridos manualmente pelo criador da atividade, e a existência de botões não intuitivos para criação de atividades, como alguns dos defeitos encontrados, e sugeriu uma lista de melhorias necessárias nesse módulo.

Em contrapartida, Cruz *et al.* (2022, p. 5) destaca a plataforma Beecrowd como uma ferramenta que possibilita a resolução de problemas e desenvolvimento de algoritmos pelos alunos, ao mesmo tempo em que incentiva a participação em maratonas de programação e auxilia o educador na correção automatizada das respostas às questões. Do mesmo modo, Bez e Tonin (2014, p. 239) mostram que a construção do Beecrowd (antigo URI Online Judge) foi inteiramente focada nas necessidades dos professores e, sobretudo, nas necessidades dos alunos. Nos aspectos didáticos e pedagógicos, é possível que o professor utilize o módulo acadêmico do portal a fim

de fornecer aos alunos listas de exercícios, acompanhando o desempenho individual de cada um, podendo categorizar as listas por tema e estabelecer prazos para sua conclusão. Ainda, o portal possui recursos extremamente importantes para uma vida estudantil acadêmica, como "um nível de problemas para iniciantes, sistema de recompensa por *badges*, um módulo acadêmico completo para acompanhamento de listas de exercícios pelos alunos, *ranking* dos alunos por Universidade, entre outros." (BEZ; TONIN, 2014, p. 239).

À vista disso, e ressaltando que o maior problema no ensino de algoritmos e programação para novos estudantes é atender com eficiência à grande diversidade de alunos e seus diferentes modos e ritmos de aprendizado (BEZ; TONIN, 2012, p. 1), "a possibilidade de utilização de ferramentas online com grande disponibilidade para aprendizagem ativa pode ser uma forma de possibilitar que cada aluno aprenda em seu próprio ritmo e velocidade" (CRUZ *et al.*, 2022, p. 5). Ademais, o uso de ferramentas online de auto aprendizado faz com que os educandos possam desenvolver suas habilidades de programação, desenvolvimento de código e solução de problemas com confiança e segurança, seguindo seu próprio ritmo de aprendizado (BEZ; TONIN, 2014, p. 239-240).

Berssanette e Francisco (2018, p. 248), salientaram que o uso do Beecrowd contribui para o processo de aprendizagem de programação dos alunos ao proporcionar *feedback* em tempo real aos estudantes e estimular a competitividade entre os alunos, por meio da gamificação presente na ferramenta, com *badges* e *ranking* dos estudantes. Também foi evidenciado o crescimento do anseio dos alunos por aprofundarem seus conhecimentos na resolução de problemas, e o envolvimento mais intenso dos mesmos na disciplina.

Equitativamente, Ferreira (2022, p. 31) informa que a plataforma Beecrowd possui estudantes de mais de 240 países registrados, e frisou alguns dos seus recursos interessantes, como: a existência de uma base de problemas classificados em categorias e níveis de dificuldade, e a presença de um modelo de solução para cada problema, o qual é comparado com os dados de saída do programa do usuário, informando se o programa possui algum percentual de erro.

De maneira geral, os alunos respondem de forma positiva ao *feedback* imediato. No ambiente de laboratório, a rápida avaliação de cada pergunta motiva os alunos a buscar notas positivas, sendo raros os casos em que avançam sem corrigir eventuais erros. Eles apreciam a capacidade de acompanhar seu progresso durante todo o laboratório. Em exames, o *feedback* imediato elimina a incerteza sobre o desempenho, permitindo que os alunos saiam da sala de exame com clareza sobre suas notas (LOBB; HARLOW, 2016, p. 49).

Ainda, de acordo com Gustavo Marques Lima (2022, p. 24), é notável que a plataforma Beecrowd seja uma das poucas no Brasil a disponibilizar questões em português brasileiro, o que a torna uma escolha significativa por parte dos professores para a utili-

zação em sala de aula. Essa preferência amplia consideravelmente a acessibilidade do juiz online, beneficiando principalmente os iniciantes na área que ainda não possuem habilidades de leitura em inglês.

Vale ressaltar que o autor também identificou duas outras plataformas, a Neps Academy e a CodeBench, que possuem questões em portugês brasileiro. Entretanto, essas alternativas não oferecem a mesma gama de recursos encontrados no Beecrowd. A Neps Academy, por exemplo, não permite o registro de turmas e a inscrição de alunos, funcionalidades que o Beecrowd oferece. Por outro lado, a CodeBench, embora seja amigável para professores e alunos, disponibiliza problemas privados. Nessa plataforma, apenas é possível visualizar as questões por meio de turmas com acesso restrito criadas pelos professores, o que impede que os alunos tentem resolver exercícios fora do ambiente de sala de aula (LIMA, G. M., 2022, p. 41).

Além disso, os autores Bez e Tonin (2012, p. 41) testaram o uso dos juizes online BOCA e UVa Online Judge em sala de aula. Quanto ao BOCA, os professores explicaram que era necessário fazer uma limpeza no *site* a cada semestre, e novos problemas tinham de ser registrados. E, toda vez que isso tinha de ser feito, a classificação dos problemas e dos usuários eram perdidas. Já no uso do UVa Online Judge, apesar desse juiz conter uma enorme variedade de problemas em todos os assuntos de algoritmos, o período de manutenção do servidor geralmente coincidia com os horários das aulas, ficando off-line, e impossibilitando a dependência da ferramenta (BEZ; TONIN, 2012, p. 1).

Bez e Tonin (2014, p. 248) também registraram que, uma semana após o lançamento online da plataforma Beecrowd (conhecida como URI Online Judge na época), estudantes da área de Computação em todo o país já acessaram o portal, e, após um mês, os dados do Google Analytics revelaram a presença constante de usuários em todos os estados brasileiros e além-fronteiras. Após cerca de um ano e meio de operação, o Beecrowd mostrou-se sendo recebido, tanto nacional quanto internacionalmente, sendo regularmente utilizado por estudantes ao redor do globo, apresentando uma notável taxa de crescimento de usuários.

Dessa maneira, ao considerar as limitações do módulo VPL, incluindo problemas de interface gráfica e usabilidade, e as desvantagens dos diversos juízes online mencionados anteriormente, destaca-se a popularidade do Beecrowd dentro e fora do Brasil, bem como seus diversos recursos avançados. Esses recursos foram desenvolvidos para atender às necessidades de alunos e professores, promovendo efetivamente o aprendizado. O crescente interesse da comunidade acadêmica em adotar o Beecrowd reflete uma demanda crescente por essa plataforma, que tem demonstrado sucesso em alcançar seus objetivos (FERREIRA, 2022, p. 31). Portanto, deve-se estimular o uso do portal em ambientes acadêmicos, a fim de melhorar a qualidade do ensino de programação e resolução de problemas nas salas de aula.

1.1 OBJETIVOS

Neste contexto, o presente trabalho visa favorecer o uso da plataforma Beecrowd entre os docentes, aproveitando a integração LTI (*Learning Tools Interoperability*) já estabelecida entre o Moodle e o Beecrowd, desenvolvida pela equipe do Beecrowd. A LTI é um padrão que facilita a troca segura de dados entre o Sistema de Gestão de Aprendizado (LMS) e ferramentas de aprendizagem externas, centralizando o acesso a conteúdos internos e externos e facilitando logins entre diferentes plataformas (VERDAGUER, 2021). Assim, a LTI do Beecrowd permite que os usuários acessem o Beecrowd diretamente do Moodle, de maneira rápida e intuitiva por meio de um botão, tornando o uso da plataforma mais acessível e ágil.

Para complementar essa integração e oferecer um suporte abrangente, será disponibilizada uma aplicação web de apoio, com um sistema especialista que atende tanto professores quanto alunos. Para os professores, a ferramenta ajudará a esclarecer dúvidas frequentes dos alunos. Já para os alunos, ela permitirá que resolvam suas dúvidas de forma independente, sem depender da disponibilidade do professor.

Como as questões do Beecrowd podem ser reutilizadas a cada semestre em atividades de ensino e prática de programação, é comum que os alunos enfrentem dúvidas semelhantes ao resolverem esses problemas, o que gera um padrão previsível de questionamentos. Dada a diversidade de questões e os critérios rigorosos do sistema de correção do Beecrowd — que exige, por exemplo, formatação exata das respostas e a aprovação em múltiplos casos de teste —, essa ferramenta de apoio pretende facilitar a adaptação dos professores, fornecendo respostas a problemas comuns e orientações sobre as expectativas de correção do juíz online do Beecrowd, além de contribuir para uma melhor experiência de ensino e aprendizado na plataforma.

1.1.1 Objetivo Geral

Favorecer o uso da plataforma Beecrowd entre os docentes, para facilitar e aprimorar o ensino da prática de programação. Será aproveitada a integração LTI do Beecrowd com o ambiente Moodle já implementada, a qual facilita o acesso à plataforma e otimiza o aproveitamento de seu extenso banco de questões, oferecendo uma ampla variedade de exercícios de forma mais eficiente. Além disso, será disponibilizada uma ferramenta baseada em sistema especialista para auxiliar tanto professores quanto alunos no esclarecimento de dúvidas recorrentes sobre as questões do Beecrowd, facilitando a adaptação de ambos ao uso da plataforma.

1.1.2 Objetivos Específicos

(a) Desenvolver um manual de instruções para auxiliar os professores no uso da LTI do Beecrowd, permitindo que compreendam e utilizem a ferramenta de forma

eficaz, com ênfase nos recursos robustos do Beecrowd e seu vasto banco de questões;

(b) Criar uma plataforma com um sistema especialista voltado ao esclarecimento de dúvidas frequentes dos alunos sobre as questões do Beecrowd. Este sistema atenderá as dúvidas recorrentes que surgem ao longo do semestre, já que muitas questões e suas respectivas dificuldades permanecem semelhantes;

(c) Elaborar um manual de instruções para orientar os docentes no processo de adicionar respostas a novas dúvidas no sistema especialista.

1.2 MÉTODOS DE PESQUISA

Na seção de fundamentação teórica, realiza-se uma análise abrangente sobre juízes online e suas aplicações no contexto acadêmico, com foco específico na plataforma Beecrowd e no BOCA *Online Contest Administrator*, investigando suas operações e funcionalidades. É importante frisar que enquanto a plataforma do Beecrowd foi desenvolvida para apoiar o ensino, o BOCA é voltado para a avaliação de submissões de código em maratonas de programação.

Nesta mesma seção, são relatadas e explicadas outras alternativas para o uso de ambientes auxiliares no ensino de programação, como o VPL e o CodeRunner, integrados ao AVA Moodle. Detalhes de funcionamento e execução de cada um são observados. Além disso, conduz-se uma análise aprofundada do ambiente AVA Moodle, explorando o conceito de *LTI External Tools* e suas relações com o Moodle, juntamente com os detalhes dos campos a serem preenchidos para a configuração de uma ferramenta externa LTI. Os conceitos LTI, Sistema Especialista e sua relação com a linguagem de programação Prolog e o projeto SWI-Prolog, e API também são expostos nessa seção, essenciais para o entendimento do desenvolvimento do sistema especialista.

Na seção de trabalhos relacionados, realiza-se uma pesquisa bibliográfica sobre a integração de juízes online e outros ambientes de ensino de programação com o Moodle. O objetivo é avaliar a eficácia do juiz online Beecrowd na aprendizagem dos alunos, além de analisar como outros projetos semelhantes foram desenvolvidos, comparando suas abordagens com o presente trabalho. Adicionalmente, são discutidos dois estudos focados na criação de sistemas especialistas web, explorando suas características e destacando as principais diferenças em relação à proposta deste trabalho.

Na seção de desenvolvimento, são explorados os aspectos relacionados à LTI do Beecrowd, que facilita a integração entre o Moodle e a plataforma Beecrowd. Nessa etapa, são apresentados manuais detalhados para a configuração e utilização da LTI no Moodle. Além disso, a seção inclui diagramas, requisitos funcionais e não funcio-

nais, bem como regras de negócio, que juntos definem a arquitetura geral do sistema especialista desenvolvido. Também são descritas as tecnologias selecionadas para o desenvolvimento, acompanhadas de uma explicação detalhada sobre a implementação da aplicação web do sistema especialista. Por fim, são exibidas imagens ilustrando o resultado final da aplicação.

Na seção de validação, será realizada uma análise da proposta, destacando as vantagens e desvantagens de utilizar a integração LTI entre o Moodle e o Beecrowd, bem como os benefícios e limitações de empregar um sistema especialista para resolver dúvidas relacionadas às questões do Beecrowd. Além disso, essa seção apresentará uma avaliação prática do uso da aplicação pelos alunos da disciplina *Programação Orientada a Objetos I* da UFSC, no segundo semestre de 2024, verificando se as dúvidas dos estudantes foram efetivamente resolvidas pelo sistema especialista.

Por fim, na seção de conclusão, será apresentado um resumo das principais ações realizadas ao longo do trabalho, com ênfase na integração do Moodle com o Beecrowd por meio de LTI e no desenvolvimento da ferramenta com sistema especialista para apoiar os alunos. Além disso, serão discutidas as possibilidades de trabalhos futuros, com sugestões de aprimoramentos e novas direções para a ferramenta.

1.3 ESTRUTURA DO TRABALHO

A estrutura deste trabalho é organizada da seguinte maneira: a seção de Fundamentação Teórica apresenta os conceitos essenciais para compreender o desenvolvimento do projeto. Em Trabalhos Relacionados, são comparadas abordagens de integração de juízes online com o Moodle, além de artigos que tratam do desenvolvimento de sistemas especialistas web. O capítulo de Desenvolvimento apresenta a arquitetura da integração LTI, as tecnologias utilizadas e a implementação do sistema especialista. Na seção de Validação, são avaliadas as vantagens e limitações do sistema. A Conclusão discute um resumo das ações realizadas no trabalho e apresenta sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos necessários para a compreensão da proposta. Serão mostrados tópicos como: juízes online, destacando o impacto dessa tecnologia na formação dos estudantes; Beecrowd, detalhando suas funcionalidades como plataforma de avaliação; BOCA *Online Contest Administrator*, VPL (*Virtual Programming Lab*) e *CodeRunner*, ferramentas de apoio ao aprendizado de programação, mas apresentam limitações para o uso em sala de aula. Em seguida, será apresentado o Moodle, com ênfase em suas características gerais e na administração de cursos, usuários e *sites*. A integração de ferramentas externas por meio de LTI também será abordada, destacando sua importância na interoperabilidade entre sistemas. Por fim, o capítulo abordará o conceito de web API, explicando seus fundamentos e funcionamento básico.

2.1 JUÍZES ONLINE

Os autores Wasik *et al.* (2018) definem os juízes online como "sistemas projetados para a avaliação confiável do código-fonte do algoritmo enviado pelos usuários, que é compilado e testado em seguida em um ambiente homogêneo". Da mesma forma, Santos e Ribeiro (2011, p. 1) explicam a principal função dos juízes online: "avaliar códigos fonte que foram enviados em uma determinada linguagem de programação". Essa avaliação automática é feita a partir de casos de teste, ou seja, cada caso de teste possui um conjunto de entradas e saídas, e verifica-se as respostas aos casos de teste da solução do usuário, com as respostas cadastradas para aquela questão no *site* (FRANCISCO; JÚNIOR; AMBRÓSIO, 2016, p. 12).

Diversos juízes online podem ser encontrados na Internet, dentre eles estão o Beecrowd (CRUZ *et al.*, 2022) e o Codebench (RIBEIRO *et al.*, 2018, p. 806), plataformas online com problemas de programação competitiva, e que aceitam soluções para esses desafios, avaliando-as quanto à sua eficácia e eficiência. Também há o BOCA *Online Contest Administrator*, um sistema que visa apoiar as competições de programação na correção de soluções apresentadas pelos competidores (CAMPOS; FERREIRA, 2004).

Ribeiro *et al.* (2018, p. 807-808) expôs que o Codebench é utilizado na UFAM (Universidade Federal do Amazonas) para dar suporte a professores e estudantes em disciplinas iniciais de programação. Assim, por meio desse sistema, professores e tutores das disciplinas podem disponibilizar exercícios de programação, listas e até mesmo provas para seus alunos, que, por sua vez, podem desenvolver soluções em uma IDE (*Integrated Development Environment*) acoplada ao próprio sistema. Consequentemente, após a adoção do Codebench como ferramenta de apoio a uma metodologia de ensino híbrido de programação na UFAM, houve um aumento no índice de aprova-

ções dos alunos da disciplina Introdução à Programação de Computadores (GALVÃO; FERNANDES; GADELHA, 2016, p. 148-149).

Semelhantemente, Francisco, Júnior e Ambrósio (2016, p. 18-19) ressaltam os benefícios da utilização de um juiz online no ensino de matérias iniciais de programação: "[...] Aprendizagem no ritmo do aluno, auto-aprendizagem e redução da carga de trabalho do professor, são alguns dos benefícios apontados que contribuem não só em ambientes tradicionais de ensino, mas em ambientes de Educação a Distância (EAD) e em MOOC's. A liberdade de definir listas de exercícios e a disponibilidade de instrumentos para acompanhar os alunos são questões importantes para o professor".

Da mesma maneira, o desempenho dos estudantes do curso de Engenharia de Software da Universidade de Brasília foi influenciado diretamente pelo uso de problemas da maratona de programação e de juízes eletrônicos: observou-se que 50,3% dos alunos apresentaram um aumento no desempenho em disciplinas de programação (SALES; JUNIOR; SALES, 2016, p. 218).

Sob outra perspectiva, o desenvolvimento dos juízes online também é voltado ao uso em competições de programação (SANTOS; RIBEIRO, 2011, p. 964-965), já que são utilizados juízes eletrônicos, corretores automáticos de problemas, em maratonas de programação (LIMA, G. M., 2022, p. 34).

É interessante observar que o formato das questões em maratonas de programação é muito semelhante ao das questões em Juízes Online, e muitos dos exercícios disponíveis em Juízes Online originaram-se de competições de programação. Por conseguinte, a introdução dos estudantes a essas plataformas cria um ambiente mais familiar para as maratonas de programação, promovendo um engajamento positivo dos alunos com essa atividade e gerando impactos favoráveis em seu desempenho em disciplinas gerais de programação (LIMA, G. M., 2022, p. 33-34).

Outrossim, Campos e Ferreira (2004, p. 2) mostram as vantagens do incentivo à participação de alunos em competições de programação: "Competições de programação são uma excelente oportunidade para desenvolver nos alunos diversas habilidades que serão extremamente úteis no seu crescimento profissional. Nelas, os alunos aprendem de forma divertida conceitos importantes sobre estruturas de dados, algoritmos e desenvolvimento de software enquanto, ao mesmo tempo, aprendem a trabalhar em grupo e sob pressão" (CAMPOS; FERREIRA, 2004, p. 2).

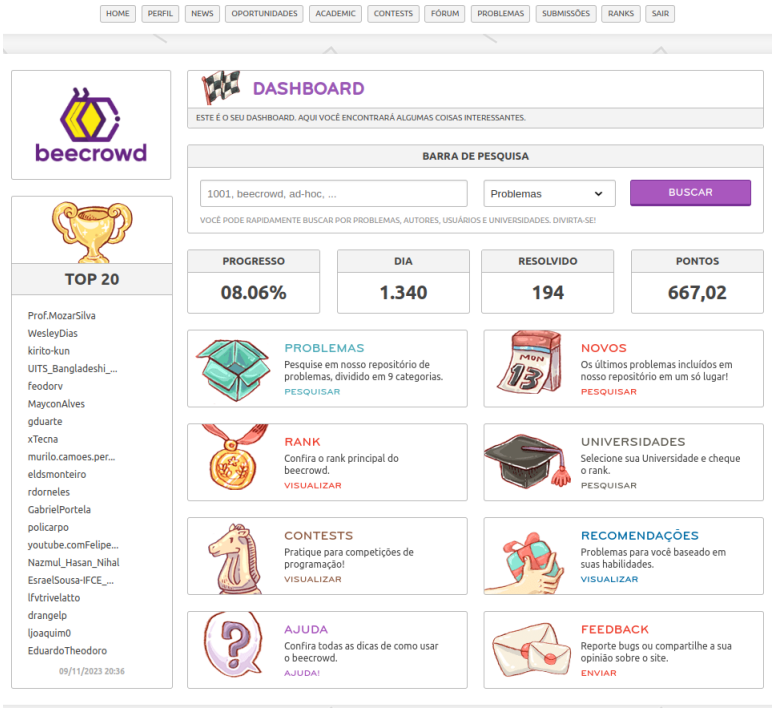
Portanto, além de seu papel essencial em competições de maratona de programação, os juízes online têm se destacado como ferramentas valiosas no ensino de programação básica. Como exposto acima, a integração dessas plataformas no ambiente educacional proporciona vantagens significativas, como a aprendizagem no ritmo do aluno e a facilidade dos professores em gerenciar uma turma. Portanto, essas ferramentas desempenham uma função fundamental na promoção do engajamento dos estudantes e no aprimoramento de seu desempenho em disciplinas gerais de programação, contribuindo significativamente para o campo educacional.

2.2 BEECROWD

O Beecrowd, desde janeiro de 2013 até outubro de 2021 conhecido como URI Online Judge (PIEKARSKI *et al.*, 2023, p. 5), é um portal online criado com o objetivo principal de tornar a prática de programação mais dinâmica, interessante e estimulante para aqueles que acabaram de ingressar na arte de programar (BEZ; TONIN, 2012, p. 1).

A plataforma é um projeto desenvolvido na Universidade Regional Integrada - URI - Campus de Erechim, inicialmente com o intuito de criar um *site* que tivesse um juiz automático, capaz de atender às necessidades dos professores e dos alunos, visando principalmente a interatividade, flexibilidade e novas fontes de informação e usabilidade (BEZ; TONIN, 2012, p. 1). O ambiente foi estruturado de forma a ser agradável, didaticamente organizado, e está disponível 24 horas por dia, oferecendo suporte tanto em inglês quanto em português (BEZ; TONIN, 2014, p. 239). A Figura 1 mostra a página inicial do Beecrowd, após o usuário logar-se na plataforma.

Figura 1 – Portal Beecrowd - Home

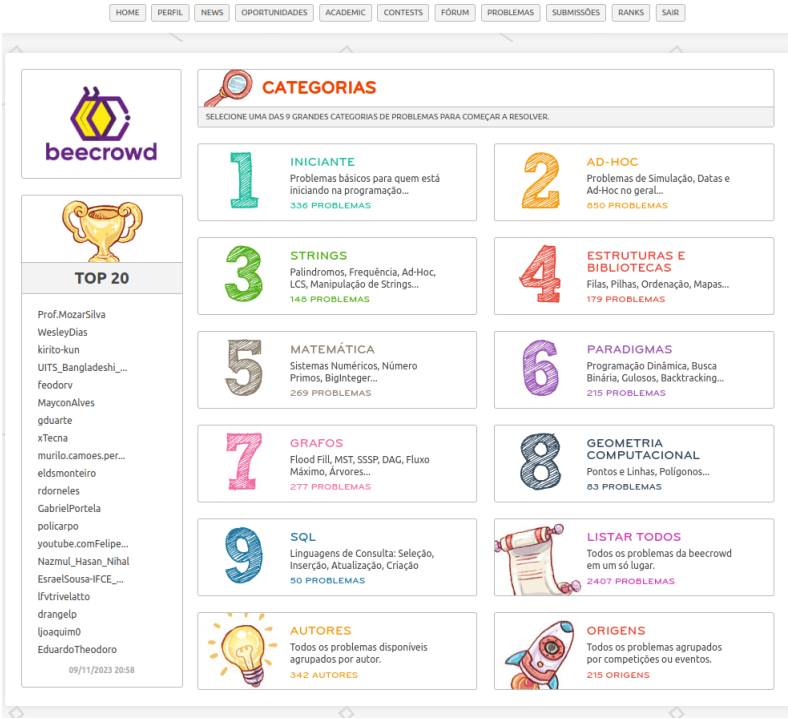


Fonte: (BEECROWD, 2024)

Além disso, o Beecrowd contém problemas no estilo do ICPC - *International Collegiate Programming Contest* da ACM, e disponibiliza um juiz online que avalia em tempo real as submissões dos algoritmos dos usuários (BERSSANETTE; FRANCISCO, 2018, p. 350). Os problemas são categorizados por assunto e por nível de dificuldade, e há flexibilidade na escolha da linguagem de programação com a qual se deseja resolver os exercícios (BEZ; TONIN, 2012, p. 1).

A Figura 2 mostra as diferentes categorizações dos exercícios, que são: Iniciante: problemas básicos para quem está iniciando na programação; Ad-Hoc[1]: problemas de simulação, datas e Ad-Hoc no geral; Strings: palíndromos, frequência, ad-hoc, LCS, manipulação de strings; Estruturas e Bibliotecas: filas, pilhas, ordenação, mapas; Matemática: sistemas numéricos, números primos, bigInteger; Paradigmas: programação dinâmica, busca binária, algoritmos gulosos, backtracking; Grafos: flood fill, MST, SSSP, DAG, fluxo máximo, árvores; Geometria Computacional: pontos e linhas, polígonos; e SQL: linguagens de consulta - seleção, inserção, atualização, criação.

Figura 2 – Portal Beecrowd - Categorias



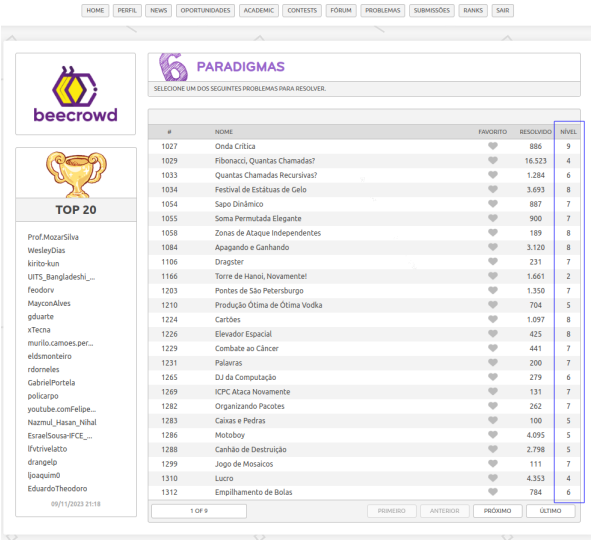
Fonte: (BEECROWD, 2024)

Já na Figura 3 estão listados diversos exercícios dentro da categoria Paradigmas, e, para cada problema, há um nível classificado (destacado na imagem com um retângulo azul). O nível representa a dificuldade da questão, e pode variar de 1 até 10, sendo 1 o nível mais fácil, e 10 o nível mais difícil (BEECROWD, 2024). Dessa maneira, impede-se que estudantes iniciantes enfrentem frustrações ao tentar resolver problemas que demandam conhecimento mais avançado, prática e técnicas (BEZ; TONIN, 2014, p. 239).

Ademais, o portal Beecrowd não apenas fornece uma plataforma desafiadora para competições de programação, mas também se destaca como um recurso educacional abrangente. Entre os seus diferenciais, evidenciam-se materiais de estudo, tutoriais sobre algoritmos e programação, e um fórum que facilita a colaboração entre

[1] https://www.geeksforgeeks.org/what-are-ad-hoc-problems-in-competitive-programming/

Figura 3 – Portal Beecrowd - Exercícios dentro da categoria Paradigmas



Fonte: adaptado de (BEECROWD, 2024)

os usuários. Além desses aspectos, o portal foi projetado com características específicas que o tornam uma ferramenta eficaz de apoio às aulas de Algoritmos e Estruturas de Dados. Por meio de problemas que abordam conceitos fundamentais dessas disciplinas, o Beecrowd oferece uma abordagem prática que contribui para uma compreensão mais sólida por parte dos estudantes (BEZ; TONIN, 2012, p. 2).

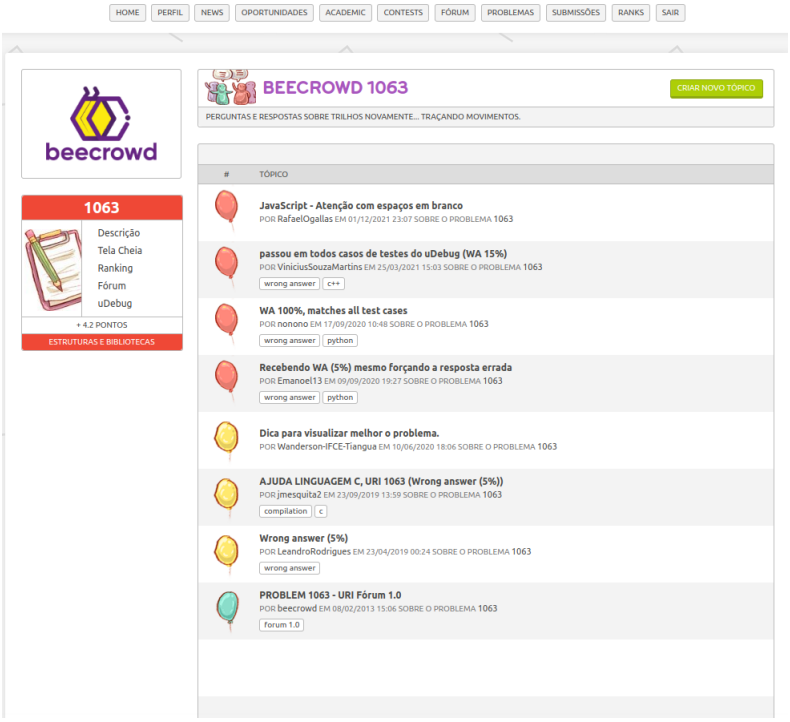
A Figura 4 mostra o fórum do exercício 1063 do Beecrowd, que pode ser acessado facilmente em Fórum -> Estruturas e Bibliotecas -> 1063, ou pelo próprio exercício 1063, no seu botão “Fórum”. Nesse fórum, diversas perguntas foram criadas para os alunos conversarem entre si sobre erros e problemas que tiveram.

Além de ser um recurso fundamental para o aprimoramento das habilidades de programação dos alunos, o Beecrowd disponibiliza recursos diferenciados que se mostram valiosos para professores na área de Tecnologia da Informação. Essas características não apenas enriquecem o ambiente de aprendizado, mas também facilitam o processo de ensino e avaliação em cursos específicos dessa área. Os professores podem, por exemplo, criar disciplinas com alunos e listas de exercícios, separadas por assunto e delimitadas por prazos, caso desejado, e acompanhar a evolução dos alunos (BEZ; TONIN, 2014, p. 239).

Na Figura 5 está a página Academic, do portal Beecrowd, onde alocam-se as disciplinas do usuário. Dentro de cada disciplina se encontram as listas de exercícios disponibilizadas aos alunos pelos professores, bem como algumas orientações aos usuários.

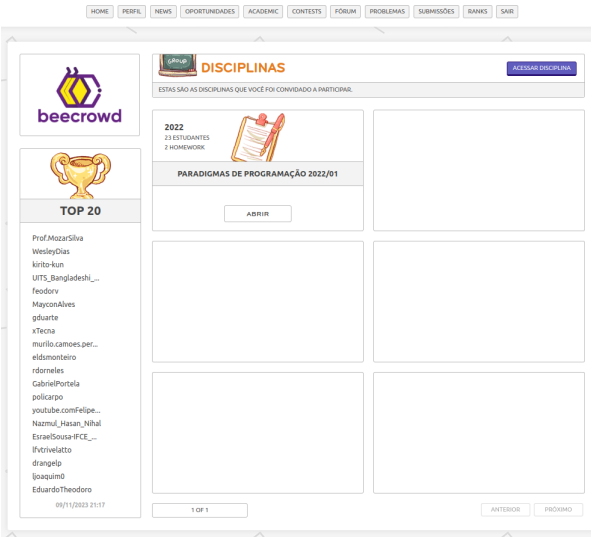
A construção do Beecrowd foi completamente centrada nas necessidades dos professores e, principalmente, nas necessidades dos alunos (BEZ; TONIN, 2014, p. 239). Ele integra recursos dos melhores portais de programação do mundo e, adicio-

Figura 4 – Portal Beecrowd - Fórum do Problema 1063



Fonte: (BEECROWD, 2024)

Figura 5 – Portal Beecrowd - Academic



Fonte: (BEECROWD, 2024)

nalmente, oferece funcionalidades exclusivas, como um nível de problemas destinado a iniciantes, um sistema de recompensas por meio de *badges*, e um módulo acadêmico completo para o acompanhamento das listas de exercícios, incluindo o *ranking* de estudantes por universidade, entre outras inovações (BEZ; TONIN, 2014, p. 239).

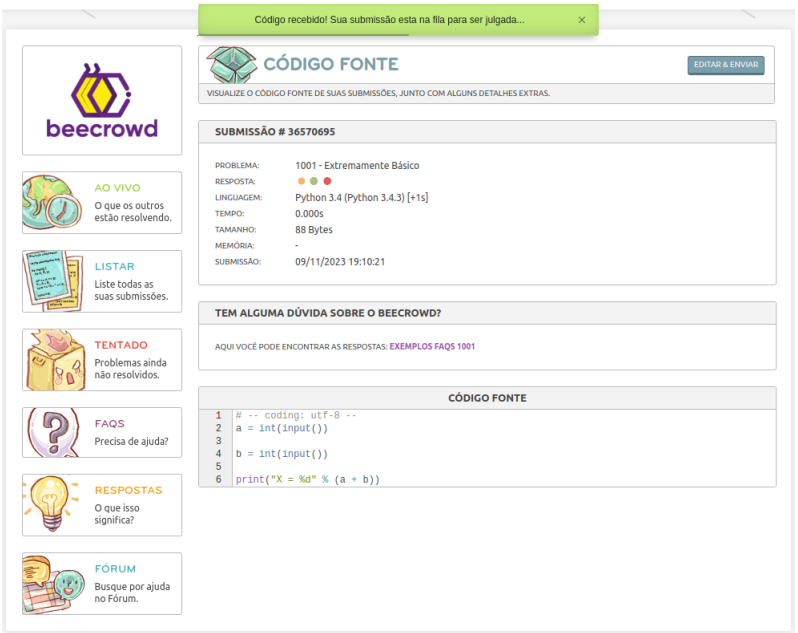
A Figura 5 exemplifica um exercício do Beecrowd, o 1001: Extremamente Básico. Cada exercício é acompanhado por sua descrição, requisitos de entrada para o progra-

ma e a saída esperada do algoritmo a ser desenvolvido. Na seção à direita, o usuário pode selecionar a linguagem de programação desejada e utilizar a IDE integrada para desenvolver seu algoritmo. Ao concluir, o usuário deve clicar no botão "Enviar"para que a plataforma avalie sua solução. Após o envio, o aluno é redirecionado para uma tela onde o resultado é exibido na seção "Resposta"(Figura 6).

Figura 6 – Portal Beecrowd - Exercício 1001

Fonte: (BEECROWD, 2024)

Figura 7 – Portal Beecrowd - Exercício Submetido



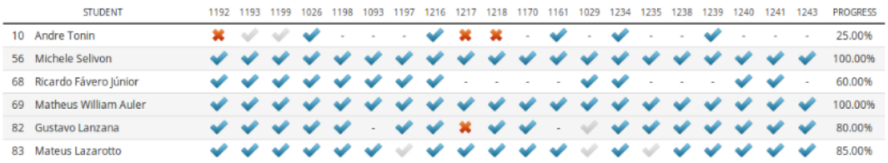
Fonte: (BEECROWD, 2024)

As respostas possíveis do Juíz Online do Beecrowd são: *Accepted, Compilation Error, Runtime Error, Time Limited Exceeded, Presentation Error, Wrong Answer, Closed*. O significado de cada resposta pode ser encontrado na plataforma, oferecendo ao usuário uma indicação do que pode estar incorreto em seu algoritmo (BEECROWD, 2024).

O Beecrowd também possui o ambiente Academic, e é esse ambiente que fornece ao professor diversos recursos (ACADEMIC, 2024), como:

- Criar disciplinas, podendo convidar alunos para participar dela;
- Criar listas de exercícios e acompanhar facilmente o progresso de seus estudantes (Veja Figura 8), podendo ter uma melhor visão sobre como eles estão aplicando os tópicos apresentados em aula e onde eles estão encontrando dificuldades;
- Selecionar exercícios de programação do banco de dados do Beecrowd para essas listas, o qual possui mais de 2.000 problemas distintos;
- Visualizar todas soluções enviadas pelos alunos para suas listas de exercícios através de uma organizada linha do tempo;
- Definir data de início, prazo e duração das listas, considerando somente as submissões recebidas durante o tempo determinado;
- Visualizar *feedback* sobre soluções copiadas entre estudantes e de repositórios online.

Figura 8 – Portal Beecrowd Academic - Progresso dos Alunos



Fonte: (BEZ; TONIN, 2012, p. 244)

Por fim, o Beecrowd oferece suporte a várias linguagens de programação e versões, dentre as quais o usuário pode escolher para desenvolver suas soluções. Todas essas linguagens estão listadas abaixo, na Figura 9.

2.3 BOCA ONLINE CONTEST ADMINISTRATOR

O BOCA *Online Contest Administrator* é um ambiente utilizado para gerenciar competições que envolvem programação de computadores (GALASSO; MOREIRA,

Figura 9 – Portal Beecrowd Academic – Linguagens e Versões

Fonte: (BEECROWD, 2024)

2014, p. 22), e foi desenvolvido para ser usado na Maratona de Programação da Sociedade Brasileira de Computação (CAMPOS; FERREIRA, 2004, p. 2).

Por ser um juiz online (BEZ; TONIN, 2012, p. 1), realiza a avaliação automática dos algoritmos submetidos, classificando-os como corretos ou incorretos (GALASSO; MOREIRA, 2014, p. 22). Ou seja, na competição, a correção dos programas enviados pelos times ocorre de forma online, e o resultado deve ser transmitido ao time o mais rápido possível. Após os competidores concluírem uma questão, enviam-na para o BOCA, que julga a solução fornecida e indica se está correta ou incorreta. Se estiver incorreta, o aluno tem a oportunidade de corrigi-la e submetê-la novamente, ainda durante o tempo da prova, mas sem ter conhecimento específico sobre o erro na questão (CAMPOS; FERREIRA, 2004, p. 4).

A utilização de interfaces web possibilita que o sistema BOCA seja aplicado na organização de competições distribuídas, permitindo que equipes estejam localizadas em qualquer lugar com acesso à internet. Essa abordagem é adequada para simulações e outras modalidades de competições (CAMPOS; FERREIRA, 2004, p. 20).

Além disso, o BOCA proporciona a flexibilidade de ser utilizado em competições com várias sedes (ou *sites*), todos operando simultaneamente, com a correção centralizada de submissões e dúvidas. Isso viabiliza competições em larga escala, alcançando regiões distantes de um país, como o Brasil, como evidenciado na Maratona de Programação. A centralização no processamento de submissões e dúvidas é essencial para assegurar igualdade de condições a todos os participantes, independentemente de sua localização, uma vez que tudo é processado pelo mesmo grupo de juízes (CAMPOS; FERREIRA, 2004, p. 10-11).

Considerando que o BOCA foi desenvolvido apenas com o objetivo de auxiliar na avaliação de submissões de código em maratonas de programação, um benefício adicional do sistema é sua aplicabilidade no contexto de aprendizado, oferecendo suporte a disciplinas que envolvem a submissão e correção de trabalhos de programação (CAMPOS; FERREIRA, 2004, p. 2). Por isso, ele foi utilizado por alunos de Ciência da Computação na URI por algum tempo, em sala de aula, para resolver problemas relacionados aos conteúdos abordados. Contudo, a cada semestre, era necessário

realizar uma limpeza no *site* e registrar novos problemas, incluindo todos os seus arquivos de entrada e saída, enunciado, e tempos limites, dependendo da matéria ensinada (BEZ; TONIN, 2012, p. 1). Os autores Bez e Tonin (2012, p. 1). destacaram que lidar com essa quantidade de problemas não é viável, pois o sistema não foi criado para esse propósito. Cada vez que o sistema era limpo, tanto a classificação dos problemas quanto a classificação dos usuários eram perdidas.

Assim, apesar de o BOCA não precisar de cadastro em nenhum portal, e ainda ter acesso a todo código fonte dos exercícios, ele não é a opção ideal para ser o sistema de juiz online utilizado em sala de aula. Ainda assim, o sistema revela-se uma escolha excelente para competições de programação. O crescimento extraordinário dessas competições tem despertado um interesse crescente entre os alunos de cursos de computação no Brasil. Esse interesse é extremamente positivo, pois motiva os alunos a estudarem conceitos fundamentais da computação, tais como estruturas de dados, análise de algoritmos, técnicas e linguagens de programação (CAMPOS; FERREIRA, 2004, p. 11).

2.4 VPL

O *Virtual Programming Lab* (VPL) é um módulo de atividades para o Moodle que gerencia tarefas de programação (VPL, 2021). Ele permite a edição de código-fonte diretamente no navegador e a execução de programas de forma interativa sem sair do ambiente Moodle. Tanto alunos quanto professores podem realizar testes para revisar os programas, enquanto a funcionalidade de pesquisa de similaridade entre arquivos auxilia na detecção de plágio. Além disso, o VPL oferece a possibilidade de definir restrições de edição e prevenir a colagem de texto externo, contribuindo para um ambiente acadêmico mais controlado e transparente (VPL, 2021).

Segundo Rodríguez-Del-Pino, Royo e Figueroa (2012, p. 1), o VPL é um ambiente de desenvolvimento simples para os alunos, oferecendo recursos de avaliação automática. Para os instrutores, funciona como um sistema de gerenciamento do trabalho dos alunos, facilitando a preparação de tarefas, gerenciando envios, verificando plágio e conduzindo avaliações eficientes e flexíveis com base em testes de programas. Além disso, é independente da linguagem de programação utilizada nos exercícios e prioriza questões críticas de segurança.

Para criar uma atividade VPL, é necessário preencher um formulário de configuração básica, seguindo o padrão comum a outros módulos do Moodle. Esse formulário abrange elementos essenciais, como nome da atividade, breve descrição, período de disponibilidade (com datas de início, término e visibilidade), opções de avaliação, entre outros. Adicionalmente, a atividade VPL permite a inclusão de dados específicos, como o número máximo de arquivos a serem enviados, o tamanho máximo desses arquivos, bem como restrições relacionadas à edição, senha, entre outros (VPL, 2021).

Ao concluir o preenchimento do formulário de configuração básica, o criador da tarefa tem a possibilidade de ajustar cinco grupos adicionais de recursos: descrição completa, casos de teste, opções, arquivos solicitados e recursos avançados. Portanto, caso o professor queira que os códigos submetidos pelos alunos sejam testados, ele mesmo precisa inserir todos os casos de teste manualmente na tarefa (VPL, 2021).

Assim, sempre que o professor desejar apresentar novas tarefas de programação para suas turmas a cada semestre, será necessário criar uma nova atividade utilizando o VPL, preenchendo integralmente o formulário de configuração básica e incluindo os casos de teste necessários.

2.5 CODERUNNER

O CodeRunner é um plugin de código aberto gratuito para o Moodle, projetado para avaliar respostas de programação em várias linguagens. Utilizado principalmente em cursos de programação de computadores, permite que os alunos enviem código em resposta a perguntas específicas e visualizem os resultados dos testes imediatamente. Os professores podem executar programas para avaliar as respostas dos alunos, utilizando uma abordagem adaptativa (CODERUNNER, 2023). Assim, os alunos desenvolvem e testam seu código usando um ambiente de desenvolvimento normal e enviam o código ao CodeRunner por meio de um navegador da Web somente quando acreditam que ele está correto (LOBB; HARLOW, 2016, p. 47).

Funcionando como um tipo de pergunta padrão no Moodle, o CodeRunner foi desenvolvido para integrar-se facilmente com outros tipos de perguntas computadorizadas, como múltipla escolha, numérica, resposta curta, correspondência e até mesmo questões de redação avaliadas por humanos (LOBB; HARLOW, 2016, p. 48).

Ao contrário do VPL, o CodeRunner opera no modo de teste adaptativo, onde os alunos podem verificar imediatamente se o código passa nos testes definidos na pergunta. Eles podem corrigir e reenviar o código, geralmente com uma pequena penalidade. A avaliação é baseada nos casos de teste aprovados pelo aluno (MOODLE, 2023).

Adicionalmente, o CodeRunner proporciona um sistema flexível de penalidades, permitindo que os criadores ajustem a avaliação conforme o contexto. Por exemplo, é possível configurar algumas perguntas sem penalidades para reenvios, outras que concedem um ou dois envios gratuitos antes de aplicar uma penalidade incremental de 10% para cada tentativa subsequente errada, e ainda outras que impõem uma penalidade de 100% mesmo para um único reenvio equivocado. Vale ressaltar a alta escalabilidade do CodeRunner: ele acomoda desde questões simples de codificação até tarefas significativamente complexas (LOBB; HARLOW, 2016, p. 48).

Para garantir a segurança, o CodeRunner necessita do software sandbox ("Jobe") instalado em uma máquina separada com medidas adequadas de segurança.

Recomenda-se o uso de um servidor Moodle separado para questionários baseados no CodeRunner, especialmente em testes e exames finais, tanto por motivos de carga quanto para que vários recursos de comunicação do Moodle, como bate-papo e mensagens, possam ser desativados sem afetar outras turmas (CODERUNNER, 2023).

Assim como no VPL, é necessário que o criador da pergunta configure seus detalhes e seus casos de teste (CODERUNNER, 2023).

A Universidade de Canterbury utiliza o CodeRunner para ministrar cursos de programação em Python, C, Octave e Matlab. Esse plugin é especialmente eficaz em cursos introdutórios de programação, proporcionando prática intensiva com pequenos problemas de programação que ensinam diversas construções e técnicas de linguagem (LOBB; HARLOW, 2016, p. 48).

Adicionalmente, o CodeRunner também é aplicável em níveis mais avançados, sendo utilizado em cursos teóricos de Ciência da Computação, inteligência artificial (com programação em Clojure) e programação na web para avaliação de *sites* desenvolvidos pelos alunos (LOBB; HARLOW, 2016, p. 48).

2.6 MOODLE

“Moodle é uma plataforma de aprendizagem projetada para fornecer a educadores, administradores e alunos um único sistema robusto, seguro e integrado para criar ambientes de aprendizagem personalizados” (MOODLE, 2023). Ou seja, o Moodle é um Sistema de Gerenciamento de Aprendizagem (LMS) poderoso e altamente extensível (MOODLE, 2023).

Desenvolvido pelo projeto Moodle, a plataforma possui uma interface web amplamente acessível, permitindo que estudantes, tutores e administradores realizem tarefas diárias de qualquer lugar do mundo. Com suporte para várias opções técnicas, garante bom desempenho, sendo utilizado por instituições de destaque, como o Open Polytechnic da Nova Zelândia, que atende mais de 45.000 estudantes e oferece 6.500 cursos registrados (MOODLE, 2023).

Projetado e auditado para segurança, o Moodle conta com funções específicas para administradores, professores, alunos e visitantes, com um conjunto claro de privilégios. Comprometido com a segurança e privacidade, mantém controles atualizados contra acesso não autorizado, perda de dados e uso indevido. Pode ser implantado em nuvem privada segura ou servidor para controle total (MOODLE, 2023).

Reconhecido globalmente em ambientes de aprendizagem, está disponível em mais de 60 idiomas e é utilizado por instituições renomadas, incluindo London School of Economics, Shell e Microsoft, contando com mais de 213 milhões de usuários em nível acadêmico e empresarial (MOODLE, 2023).

O Moodle oferece um conjunto robusto de ferramentas centradas no aluno e ambientes colaborativos para facilitar o ensino e aprendizagem. Com uma interface

simples, é fornecido gratuitamente como software de código aberto sob a GNU General Public License, permitindo adaptação para projetos comerciais e não comerciais sem custos de licenciamento. Sua constante revisão atende às necessidades em evolução dos usuários e permite escalabilidade para turmas pequenas e grandes organizações (MOODLE, 2023).

Como software de código aberto, possibilita personalização e adaptação, com estrutura modular e design interoperável para o desenvolvimento de plugins e integração de aplicativos externos. Proporciona flexibilidade excepcional para apoiar a aprendizagem combinada e cursos totalmente online, configurável conforme necessário e integrando facilmente diversas ferramentas colaborativas (MOODLE, 2023).

Contando com forte apoio de uma comunidade global ativa, equipe de desenvolvedores dedicados e parceiros certificados, o Moodle recebe atualizações rápidas, correções de bugs e melhorias em lançamentos significativos a cada seis meses (MOODLE, 2023).

Atualmente, diversas instituições educacionais buscam modernizar seus sistemas de aprendizagem para facilitar o acesso ao conhecimento. Nesse contexto, o ambiente Moodle tem se destacado como uma escolha frequente. O Moodle adota o conceito construcionista social de educação, conforme destacado por (GALASSO; MOREIRA, 2014, p. 22). Esse enfoque pedagógico enfatiza as demandas contemporâneas por métodos de ensino mais dinâmicos e participativos.

2.6.1 Características Gerais do Moodle

O Moodle, flexível e de fácil instalação em plataformas que suportam PHP, exige apenas uma base de dados. Destaca-se por sua ênfase na segurança, implementando verificações rigorosas em formulários, validação de dados, e codificação de cookies. Também permite que um *site* Moodle suporte milhares de cursos, estes podendo ser categorizados e pesquisados.

2.6.1.1 Administração do site

- O *site* é administrado por um usuário administrador, permitindo ajustes de aparência por meio de extensões (plug-ins) de temas;
- Oferece suporte a extensões de módulos de atividade e pacotes de idiomas, possibilitando compatibilidade com mais de 60 idiomas;
- O código PHP, licenciado sob GPL, é claro e facilmente modificado para atender às necessidades individuais.

2.6.1.2 Administração dos usuários

- Objetiva reduzir a intervenção do administrador, suportando diversos mecanismos de autenticação, como email, LDAP, IMAP, POP3, NNTP, e base de dados externa;
- Permite que cada pessoa possua uma única conta para todo o servidor, com diferentes acessos configuráveis;
- Uma conta de administrador controla a criação de cursos e cria professores através da inscrição de usuários aos cursos;
- Somente é permitida a uma conta de criador de cursos criar e dar aula nos cursos;
- Os professores podem acrescentar uma “chave de inscrição” a seus cursos para manter fora os não inscritos. Eles podem fornecer essa chave diretamente ou através do email particular de cada um;
- Os professores podem incluir e excluir alunos manualmente;
- Cada usuário pode especificar faixas de horário para que cada compromisso seja ajustado a esses horários, e escolher o idioma de interface.

2.6.1.3 Administração do Curso

- Professores têm controle total sobre os parâmetros do curso, podendo restringir outros professores;
- Oferece flexibilidade na composição de atividades, como fóruns, jornais, questionários, recursos, pesquisas de opinião, tarefas e chats.
- Todas as notas para os fóruns, jornais, questionários e tarefas podem ser vistas em uma página;
- Rastreamento detalhado dos usuários, relatórios de atividade com gráficos, e história detalhada do envolvimento de cada aluno, como postagens.

2.6.2 LTI External tools

Ferramentas Externas LTI do Moodle são aplicativos adicionais que podem ser integrados ao curso, como conteúdo interativo, atividades ou avaliações. Os professores podem vincular as atividades diretamente na página do curso no Moodle, e os alunos podem acessá-las sem sair do curso no Moodle ou precisar fazer login em um sistema diferente. Dependendo de cada ferramenta, os professores também podem fazer com que as notas sejam enviadas de volta ao Moodle (MOODLE, 2023).

A conexão entre o Moodle e essas ferramentas é feita através do padrão *Learning Tools Interoperability* (LTI), que integra plataformas de aprendizado com ferramentas de aprendizado para criar uma experiência de aprendizado mais rica e contínua. Essencialmente, o padrão LTI permite uma troca segura e bidirecional de informações entre o Moodle e ferramentas de aprendizado externas (MOODLE, 2023).

O administrador tem permissão para gerenciar Ferramentas Externas LTI, podendo adicionar novas ferramentas aos seus cursos usando o botão 'Adicionar ferramenta' na página de Ferramentas Externas LTI (MOODLE, 2023).

As ferramentas adicionadas no nível do curso aparecerão na sua lista de atividades por padrão, sendo possível removê-las da lista de atividades acessando a página do curso > Mais > Ferramentas Externas LTI e desmarcando-as na coluna 'Mostrar na lista de atividades' (MOODLE, 2023).

Para configurar uma ferramenta externa LTI, é necessário ir até "Configurações da ferramenta", fornecer um nome para a ferramenta (e, opcionalmente, uma descrição). Para preencher os outros campos nesta seção, incluindo a versão LTI, deve-se seguir as instruções do fornecedor da sua ferramenta. Se a ferramenta não fornecer informações sobre algumas dessas configurações, você pode deixá-las em branco (MOODLE, 2023).

De acordo com Moodle (2023), os campos a serem preenchidos para configuração de uma ferramenta externa LTI são:

- URL da ferramenta – Esta é a URL para conectar ao *site*. Se o *site* Moodle usar SSL (estiver em HTTPS), só é possível usar uma ferramenta que também use SSL. É necessário certificar-se de que a URL da ferramenta tenha HTTPS antes de tentar usá-la;
- Versão LTI – A versão LTI da ferramenta que está sendo adicionada;
- Chave do consumidor – Isso informa ao *site* LTI compatível conectado que o Moodle está autorizado a se conectar. O "fornecedor da ferramenta", ou seja, o gerente do *site* LTI compatível conectado, fornecerá essa chave. Se o administrador estiver apenas vinculando a uma ferramenta sem acesso seguro ou compartilhamento de notas, não será necessário uma chave do consumidor. Se estiver vinculando a um curso ou atividade de outro *site* Moodle, o administrador pode adicionar qualquer chave do consumidor;
- Segredo compartilhado – Este é o "senha"para conectar à ferramenta – o *site* LTI compatível;
- Parâmetros personalizados – Na maioria das vezes, esse campo pode ficar em branco. O fornecedor da ferramenta pode usar isso para permitir que o administrador exiba um recurso específico;

- Contêiner de lançamento padrão – Esta é a forma como a ferramenta externa será exibida;
- Incorporar – O conteúdo aparecerá dentro da página de atividade onde a ferramenta é usada, sem afastar os usuários da página, mas ocultando quaisquer blocos que o curso possa ter;
- Incorporar sem blocos – O conteúdo será exibido na guia ou janela existente. Os usuários terão que navegar de volta para o curso usando o botão 'Voltar' assim que terminarem;
- Janela existente – O conteúdo será exibido na guia ou janela existente. Os usuários terão que navegar de volta para o curso usando o botão 'Voltar' assim que terminarem;
- Nova janela – A ferramenta externa será aberta em uma nova janela. (Uma nova janela ou guia será aberta com a ferramenta externa e a janela do navegador antiga contendo a página do curso não será alterada);
- URL de seleção de conteúdo – A URL de Seleção de Conteúdo será usada para abrir a página de seleção de conteúdo do fornecedor da ferramenta. Se estiver vazia, a URL da ferramenta será usada;
- URL do ícone – É possível exibir um ícone diferente do ícone padrão da Ferramenta Externa inserindo sua URL nesse campo;
- URL do ícone seguro – É possível inserir URL de um ícone diferente aqui se os alunos estiverem acessando o Moodle com segurança via SSL

2.7 LTI

LTI *Learning Tools Interoperability* é um padrão que permite a troca segura de dados entre o sistema de gestão de aprendizado (LMS) e ferramentas de aprendizagem externas, centralizando o acesso a conteúdo interno e externo, oferecendo uma experiência de aprendizado consistente, eliminando a necessidade de múltiplos logins para diferentes plataformas (VERDAGUER, 2021).

A especificação *Learning Tools Interoperability* foi inicialmente introduzida como Basic LTI em maio de 2010 pelo IMS Global Learning Consortium (agora conhecida como LTI 1.0). Desde então, essa especificação ganhou ampla aceitação como um método simples, mas eficaz, para integrar conteúdos e produtos de terceiros aos Ambientes Virtuais de Aprendizagem (VLEs). Hoje em dia, ela faz parte integral dos principais VLEs, como o Moodle, Blackboard Learn 9, Desire2Learn e o Canvas da Instructure (VICKERS; BOOTH, 2014, p.4).

2.8 SISTEMA ESPECIALISTA

De acordo com Singla (2013), um sistema especialista é um conjunto de programas que manipula conhecimento para resolver problemas em um domínio especializado que exige a *expertise* humana. Já Bohanec, Bratko e Rajkovic (1990) definem sistemas especialistas como sistemas inteligentes de informação que imitam, em certo grau, o comportamento de um especialista humano no domínio em questão.

Também chamados de sistemas baseados em conhecimento, esses programas de computador armazenam o conhecimento específico de um ou mais especialistas humanos (SINGLA, 2013). Os principais componentes de um sistema especialista são a base de conhecimento e o motor de inferência. A base de conhecimento armazena informações sobre um domínio específico de problemas, enquanto o motor de inferência utiliza esse conhecimento para resolver problemas indicados pelo usuário, gerando explicações orientadas a ele sobre as soluções propostas (BOHANEC; BRATKO; RAJKOVIC, 1990).

Assim, a base de conhecimento contém o conhecimento do domínio necessário para resolver problemas, geralmente na forma de regras, um dos paradigmas mais populares para a representação de conhecimento; e o motor de inferência é o núcleo do sistema, responsável por gerar conclusões a partir da base de conhecimento através de inferência ou raciocínio (SINGLA, 2013). Portanto, um sistema especialista para tomada de decisão precisa estabelecer uma base de conhecimento apropriada e usá-la para o cálculo de utilidade (BOHANEC; BRATKO; RAJKOVIC, 1990).

O autor Singla (2013) ressaltou que as principais características de um sistema especialista incluem interface com o usuário, representação de dados, inferência, capacidade de explicação e habilidade de lidar com incertezas. Entre as vantagens, esse tipo de sistema oferece resposta rápida, maior confiabilidade, redução de custos e de erros, uso de múltiplas *expertises*, banco de dados inteligente e mitigação de riscos. Contudo, o sistema apresenta algumas limitações, como a ausência de senso comum, a incapacidade de responder a casos excepcionais e a dificuldade de adaptação a ambientes em constante mudança (SINGLA, 2013).

2.8.0.1 Prolog

De acordo com Clark, McCabe e McCabe (1980), o Prolog, uma linguagem de programação baseada na lógica de predicados proposta por Kowalski, foi desenvolvida pelo grupo de Alain Colmerauer em Marselha. Sua implementação mais conhecida é o compilador DEC-10 de Edimburgo, que apresenta eficiência comparável à de LISP, mas com maior suporte a inferências, essencial em sistemas especialistas. Em Marselha, é aplicada no processamento de linguagem natural, enquanto em Edimburgo é usada para planejamento, resolução de problemas e desenvolvimento de compiladores (CLARK; MCCABE; MCCABE, 1980).

Clark, McCabe e McCabe (1980) também explicam que, na Hungria, a linguagem de programação Prolog foi amplamente utilizada em sistemas especialistas, especialmente na farmacologia, incluindo previsão de atividades biológicas e interações medicamentosas. Experiências como um sistema de diagnóstico de falhas no Imperial College demonstram o potencial de Prolog para sistemas especialistas, incentivando novas aplicações (CLARK; MCCABE; MCCABE, 1980) .

Já Russell e Norvig (2013) mostram que o Prolog é, de longe, a linguagem de programação lógica mais amplamente utilizada, sendo popular para prototipação rápida e manipulação simbólica, como no desenvolvimento de compiladores e análise de linguagem natural. Além disso, diversos sistemas especialistas foram implementados em Prolog para áreas como direito, medicina (SINGLA, 2013), finanças e outros domínios especializados (RUSSELL; NORVIG, 2013).

Adicionalmente, o Prolog é ideal para desenvolver serviços Web robustos devido à sua semântica segura e gerenciamento automático de memória (WIELEMAKER; HUANG; MEIJ, 2008).

Programas em Prolog consistem em conjuntos de cláusulas escritas com a seguinte notação: as variáveis são indicadas por letras maiúsculas, enquanto as constantes são escritas em letras minúsculas; e a estrutura das cláusulas, ao invés de A B C, utiliza-se C :- A, B, com vírgulas separando os elementos da conjunção (RUSSELL; NORVIG, 2013).

2.8.0.2 SWI-Prolog

O SWI-Prolog é uma implementação de Prolog baseada em um subconjunto da WAM (*Warren Abstract Machine*), que possui um ambiente robusto de desenvolvimento, priorizando compatibilidade, portabilidade, escalabilidade e estabilidade (WIELEMAKER, 2012).

O SWI-Prolog, projeto de código aberto, foi moldado para desenvolver protótipos de pesquisa, especialmente em sistemas interativos e que exigem intensivo uso de conhecimento (WIELEMAKER; SCHRIJVERS *et al.*, 2012). Além do sistema básico, foram integrados ao SWI-Prolog várias interfaces e bibliotecas de restrições. Assim, o SWI-Prolog atua como uma ferramenta integradora e versátil, permitindo a interação com diversos recursos externos e apoiando inovações da comunidade Prolog (WIELEMAKER; SCHRIJVERS *et al.*, 2012).

Como o SWI-Prolog inclui uma extensa gama de biblioteca, que permite integração com gráficos (XPCE), bancos de dados, redes, outras linguagens de programação e diferentes formatos de documentos, ele é uma escolha poderosa para o desenvolvimento de aplicações que requerem integração e lógica sofisticada, mantendo uma estrutura de código organizada e eficiente (WIELEMAKER, 2014, p. 1).

Outra vantagem do SWI-Prolog é a disponibilidade de bibliotecas para servi-

dor HTTP, o que facilita o desenvolvimento de controles de servidor (WIELEMAKER; LAGER; RIGUZZI, 2015). Essas bibliotecas foram projetadas para que o Prolog se comunique com outros componentes de aplicações Web por meio do protocolo HTTP padrão. Com isso, a arquitetura é capaz de gerenciar protocolos de transferência e realizar parsing, representação e geração dos principais formatos de documentos Web, como HTML, XML e RDF, de forma integrada e padronizada (WIELEMAKER; HUANG; MEIJ, 2008).

2.9 WEB API

Os *Web Services* (Serviços da Web) são servidores da Web dedicados a propósitos específicos, atendendo às demandas de um *site* ou de outros aplicativos. Para interagir com esses serviços, os programas clientes utilizam Interfaces de Programação de Aplicações (APIs). Em essência, uma API expõe conjuntos de dados e funções, facilitando a comunicação entre programas de computador e viabilizando a troca de informações. Uma API da Web representa a interface de um serviço da Web, sendo responsável por receber e responder diretamente às solicitações dos clientes (MASSÉ, 2011, p. 5).

A interação com uma Web API envolve a realização de solicitações (requests) por meio dos protocolos de comunicação HTTP ou HTTPS. Essas solicitações são essenciais para a comunicação entre programas clientes e serviços da Web, permitindo a troca de informações de maneira estruturada (RICHARDSON; AMUNDSEN; RUBY, 2011, p. 18-20).

Quando um programa cliente envia uma solicitação para uma Web API, ele está requisitando a execução de uma operação específica ou a obtenção de dados do serviço da Web. Essa solicitação é processada pelo serviço da Web, que fornece uma resposta (response) ao cliente. O formato comum para a estruturação dessas respostas é o JSON (JavaScript Object Notation), um padrão amplamente adotado para representar estruturas de dados (RICHARDSON; AMUNDSEN; RUBY, 2011, p. 19-21).

Essa dinâmica de solicitação e resposta é a base do funcionamento de uma Web API. As solicitações podem conter parâmetros que informam ao serviço da Web sobre a operação desejada, e a resposta contém os dados solicitados ou uma confirmação da operação realizada. Esse modelo facilita a integração de sistemas e a comunicação entre diferentes componentes de software, sendo fundamental em ambientes de desenvolvimento web e integração de aplicativos (RICHARDSON; AMUNDSEN; RUBY, 2011, p. 20-23).

3 TRABALHOS RELACIONADOS

Neste capítulo, inicialmente é apresentado um artigo que descreve um modelo de plano de ensino baseado na plataforma Beecrowd, destacando suas justificativas e aspectos positivos. Em seguida, são analisados quatro artigos sobre a integração de diferentes ambientes de desenvolvimento com o Moodle: BOCA, CodeRunner, VPL, e um estudo sobre a integração de juízes online em geral com o Moodle. Por fim, são discutidos dois estudos focados na criação de sistemas especialistas web, explorando suas características e abordagens.

3.1 UTILIZAÇÃO DA PLATAFORMA BEECROWD DE MARATONA DE PROGRAMAÇÃO COMO ESTRATÉGIA PARA O ENSINO DE ALGORITMOS

Neste estudo conduzido por (CRUZ *et al.*, 2022), é apresentada uma proposta de plano de ensino que se baseia no aproveitamento da plataforma Beecrowd para maratonas de programação, alinhada a abordagens pedagógicas ativas. Dessa maneira, são integrados conceitos como sala de aula invertida e gamificação, proporcionando uma abordagem dinâmica e envolvente no processo de aprendizagem e explorando as potencialidades da plataforma Beecrowd para oferecer uma experiência educacional inovadora.

Os autores têm como objetivo aplicar essa metodologia de maneira a diminuir a curva de aprendizagem dos alunos, apoiar os professores na implementação de abordagens pedagógicas ativas em sala de aula e ampliar as oportunidades de interação entre os alunos, contribuindo assim para o desenvolvimento do aprendizado.

Os autores destacam a gamificação e a sala de aula invertida como estratégias que integram métodos de ensino com diversos recursos de comunicação, incluindo materiais escritos, orais e audiovisuais, junto a atividades, desafios e informações contextualizadas. A gamificação é apresentada como a aplicação de elementos característicos de jogos, uma abordagem eficaz para transformar a forma como os estímulos são apresentados aos alunos, visando engajá-los e incentivá-los a resolver problemas da maneira mais eficiente possível.

De maneira semelhante, a sala de aula invertida é descrita como uma proposta para dinamizar a transmissão de informações durante as aulas, utilizando ferramentas online estruturadas e bem planejadas. Essa abordagem visa favorecer a iniciativa dos alunos ao estudarem previamente determinados conteúdos antes das aulas ministradas pelos professores.

Os autores ressaltaram a importância da disciplina de algoritmos como a base fundamental para o ensino de programação em diversos cursos na área da Informática. Essa disciplina aborda princípios essenciais de lógica de programação, visando desenvolver a capacidade dos estudantes em analisar e resolver problemas por meio

da criação de algoritmos. Contudo, enfrenta desafios significativos, evidenciados pelo elevado índice de evasão e reprovação, tornando-se um ponto crítico nos cursos de graduação e impactando a continuidade dos alunos.

A complexidade reside, em grande parte, na diversidade de alunos e em seus distintos estilos e ritmos de aprendizado, conforme indicado pelos autores. A adaptação eficiente a essa heterogeneidade representa um desafio no ensino de algoritmos e programação para novos estudantes. Nesse contexto, os autores propõem que a utilização de ferramentas online amplamente disponíveis para a aprendizagem ativa pode ser uma estratégia eficaz para permitir que cada aluno progrida em seu próprio ritmo e velocidade (CRUZ *et al.*, 2022, p. 5).

No que diz respeito à integração de questões de maratonas de programação em sala de aula, os autores destacam sua relevância ao promover a busca ativa pela resolução de problemas. Ao buscar soluções, os alunos não apenas adquirem conhecimento sobre tópicos ainda não abordados em aula, mas também consolidam conceitos previamente ensinados.

O Beecrowd se destaca como uma ferramenta online que não apenas viabiliza a resolução de problemas em competições de programação, mas também se apresenta como uma valiosa aliada para o trabalho do professor. Os autores do estudo evidenciam que os desafios oferecidos pela plataforma Beecrowd estão meticulosamente organizados em categorias e níveis de dificuldade. As categorias abrangem temas como Iniciante, AD-HOC, Strings, Estruturas e Bibliotecas, Matemática, Paradigmas, Grafos, Geometria Computacional e SQL. A escala de dificuldade varia de 1 a 10, sendo 1 as questões mais acessíveis e 10 as mais desafiadoras.

No âmbito do artigo analisado, foram selecionadas questões da categoria Iniciante, e a partir delas, elaborou-se um plano de ensino voltado para a implementação da metodologia ativa da maratona de programação. Este plano foi aplicado em seis semestres na disciplina de algoritmos, envolvendo alunos que, em sua maioria, não possuíam conhecimento prévio em programação. A avaliação da proposta foi conduzida ao longo de três semestres que adotaram as metodologias ativas e outros três semestres que não as utilizaram, de forma intercalada.

Através desse experimento e da análise de seus resultados, os autores constataram que a aplicação das metodologias de gamificação e sala de aula invertida contribuiu significativamente para o desempenho da turma, observando-se um crescimento notável. Além disso, as notas mais elevadas foram mais consistentes ao longo do tempo, demonstrando a eficácia dessas abordagens no contexto educacional.

Dessa forma, constatou-se que integrar o Beecrowd como um suporte para a implementação da gamificação como metodologia ativa na disciplina de algoritmos foi eficaz. A adoção do formato de maratona de programação possibilita ao professor envolver os alunos de forma mais imersiva, enquanto a plataforma oferece suporte adicional para o desenvolvimento da disciplina. Isso não apenas aprimora o processo

de ensino e aprendizagem, mas também contribui para a redução da evasão em cursos superiores de informática.

3.2 INTEGRAÇÃO DO AMBIENTE BOCA COM O AMBIENTE MOODLE PARA AVALIAÇÃO AUTOMÁTICA DE ALGORITMOS

O estudo de Galasso e Moreira (2014) explora a integração entre o sistema BOCA *Online Contest Administrator* e a plataforma Moodle, direcionada para o ensino de algoritmos. Essa integração visa proporcionar uma experiência simplificada para alunos e professores, facilitando o processo de avaliação automática de algoritmos no Moodle, já que o BOCA, além de gerenciar competições de programação, permitindo, assim, eventos como maratonas de programação, possui também um juiz automático para correção de algoritmos. Essa funcionalidade possibilita que os alunos recebam respostas imediatas sobre a correção de seus algoritmos, contribuindo para um aprendizado mais ágil e eficiente.

Na integração proposta, os professores podem criar questões de programação no Moodle, onde os alunos submetem códigos-fonte que são automaticamente avaliados pelo BOCA. Os algoritmos submetidos são compilados e executados com os dados de entrada predefinidos no Moodle, e suas saídas são comparadas com uma saída padrão para determinar se estão corretos. A resposta ao aluno é binária (correta ou incorreta), sem avaliação parcial, mas Moodle permite múltiplas tentativas, oferecendo a oportunidade de corrigir erros como problemas de compilação, formatação ou limite de tempo. Adicionalmente, o professor pode revisar manualmente os resultados, caso deseje realizar uma avaliação mais detalhada e parcial das respostas.

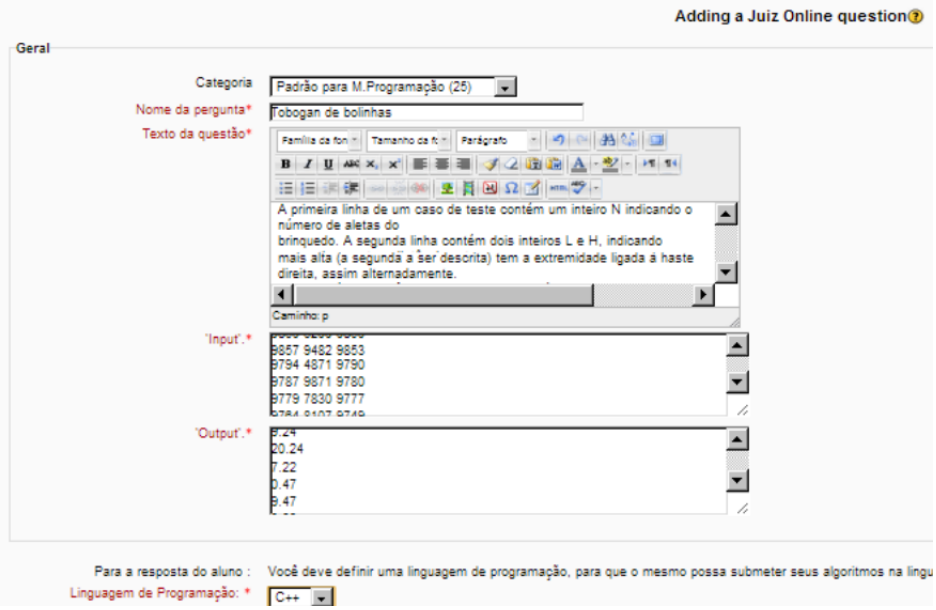
A Figura 10 representa uma edição da tela de cadastro da questão, proporcionando uma visão ilustrativa desta fase do processo, já a Figura 11 oferece uma representação visual da avaliação de uma questão como correta.

Para manter o desempenho do Moodle, a integração foi concebida de modo a garantir que os ambientes envolvidos fossem implementados em servidores independentes. Essa separação minimiza a possibilidade de sobrecarga no Moodle durante a avaliação dos algoritmos no BOCA. A integração foi desenvolvida com o PostgreSQL, e os bancos de dados do Moodle e do BOCA se comunicam via Dblink, um recurso que permite o acesso remoto a tabelas específicas.

As tabelas principais do BOCA envolvidas na avaliação automática incluem a “problemtable”, para cadastro de questões, “runtable”, para submissões dos alunos, “answertable”, para dados de correção, e “langtable”, com as linguagens de programação suportadas (C, C++ e Java). Os autores também criaram duas tabelas auxiliares, “tempaluno” e “tempprofessor”, para facilitar a transferência de dados entre o Moodle e o BOCA.

Os gatilhos (triggers) configurados no banco de dados são fundamentais para o

Figura 10 – Integração BOCA-Moodle - Edição de tela de cadastro de questão



Fonte: (GALASSO; MOREIRA, 2014, p. 26)

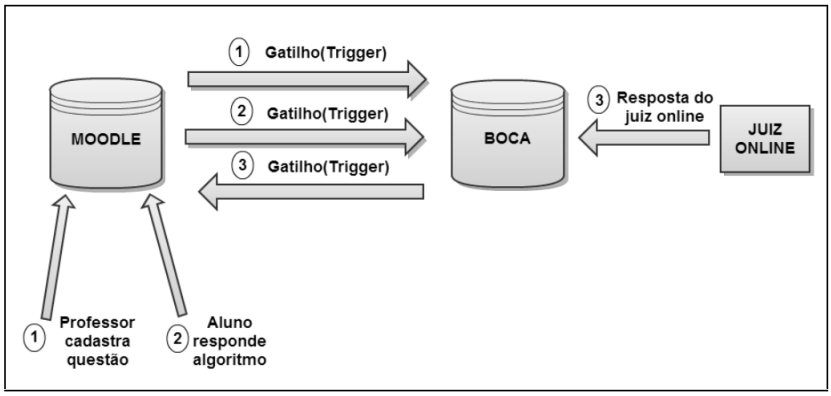
Figura 11 – Integração BOCA-Moodle - Visualização do retorno da avaliação da questão

Tentativa	Estado	Notas / 1,00	Nota / 100,00	Revisão	Comentários
Visualização prévia	Finalizadas Enviada(s) quarta, 17 julho 2013, 17:15	1,00	100,00	Revisão	Correto Algoritmo

Fonte: (GALASSO; MOREIRA, 2014, p. 27)

fluxo de dados entre Moodle e BOCA, eliminando a necessidade de alterações no código. O fluxo começa quando o professor cadastra uma questão no Moodle, acionando gatilhos que transferem os dados para o BOCA, onde a questão é registrada. Da mesma forma, quando um aluno responde, gatilhos adicionais transferem as submissões para a tabela de execução no BOCA.

Figura 12 – Integração BOCA-Moodle - Fluxo da interação das bases de dados do Moodle e do BOCA

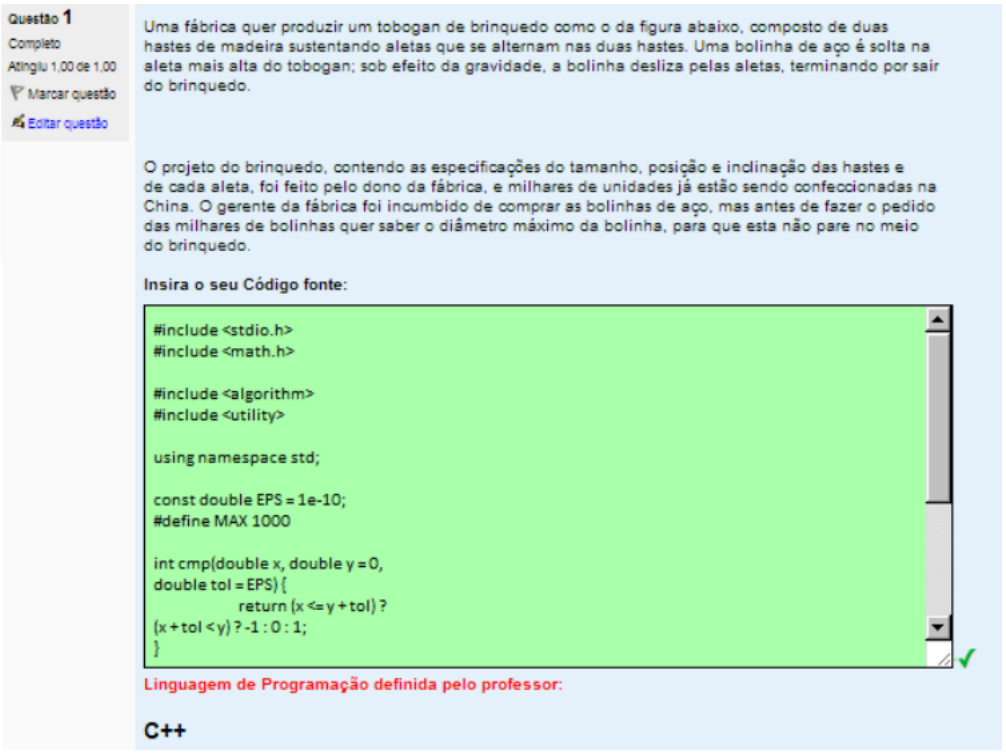


Fonte: (GALASSO; MOREIRA, 2014, p. 29)

Durante o julgamento das respostas, o BOCA executa scripts em intervalos de

um minuto, verificando novas submissões e atualizando o status no Moodle. Os alunos visualizam uma área verde para respostas corretas e vermelha para incorretas. Figuras incluídas no estudo ilustram etapas do cadastro das questões e o *feedback* visual ao aluno.

Figura 13 – Integração BOCA-Moodle - retorno da avaliação da questão como correta



Fonte: (GALASSO; MOREIRA, 2014, p. 26)

Houve desafios técnicos, como a tentativa inicial de enviar arquivos “zip” diretamente para o banco de dados, o que levou a problemas de codificação. Para contornar isso, os arquivos foram transferidos via FTP, necessitando a instalação de um servidor FTP no ambiente do BOCA. Ao final, os autores ressaltam a importância de uma instalação simplificada e recomendam uma versão do BOCA já configurada para integração com o Moodle. Essa abordagem visa facilitar a adoção da solução e sua eficácia no ensino de algoritmos.

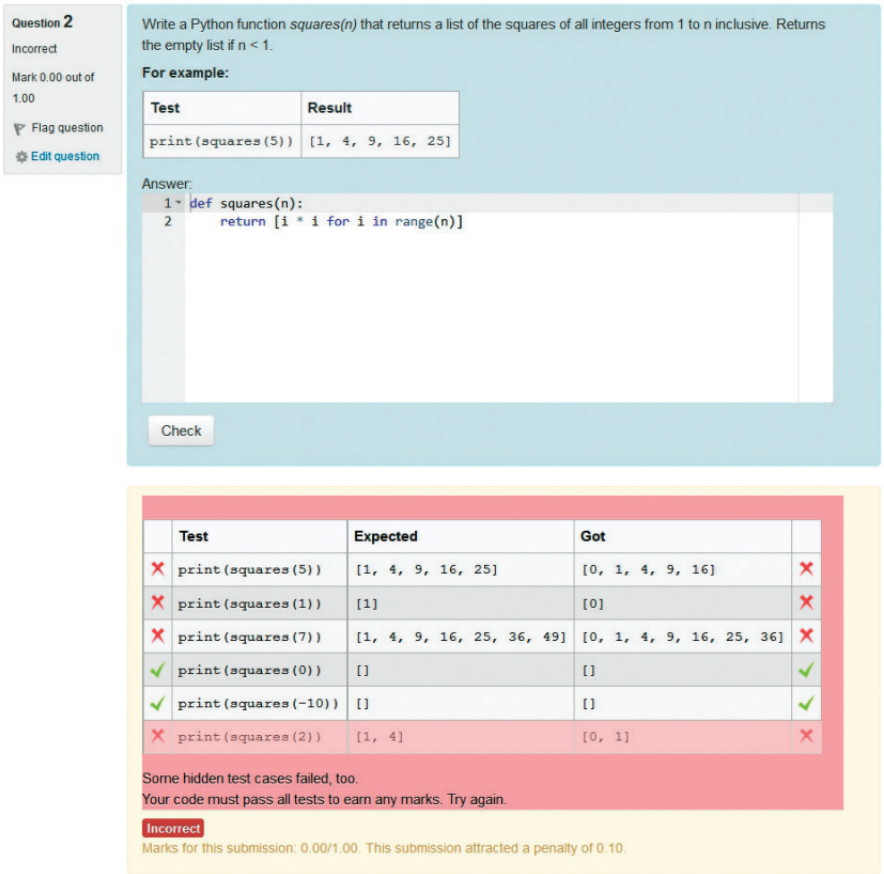
3.3 CODERUNNER: A TOOL FOR ASSESSING COMPUTER PROGRAMING SKILLS

O estudo de Lobb e Harlow (2016) examina as dificuldades encontradas na avaliação de alunos em disciplinas de programação utilizando métodos tradicionais, como provas de papel e caneta. O artigo destaca que a complexidade dos códigos e a natureza da programação moderna tornam a avaliação precisa um desafio, pois esses métodos não oferecem a possibilidade de realizar testes ou debug. O artigo

propõe o uso de ferramentas de avaliação automatizadas, como o CodeRunner, que proporciona correção e *feedback* imediato para os alunos.

Pelo CodeRunner, plugin do Moodle, os alunos recebem um *feedback* binário, onde as respostas corretas são indicadas por marcas verdes e as incorretas por cruzes vermelhas. A ferramenta adota o método de avaliação "tudo ou nada", ou seja, uma resposta só é considerada correta se atender completamente aos requisitos do teste, sendo a nota zero atribuída para respostas incorretas. Contudo, o sistema permite que os alunos façam múltiplas tentativas, oferecendo a chance de corrigir falhas como erros de compilação, formatação inadequada ou problemas de tempo de execução, com a possibilidade de aplicar penalidades em caso de envio tardio ou repetido.

Figura 14 – CodeRunner - A simples pergunta sobre Python, respondida erroneamente

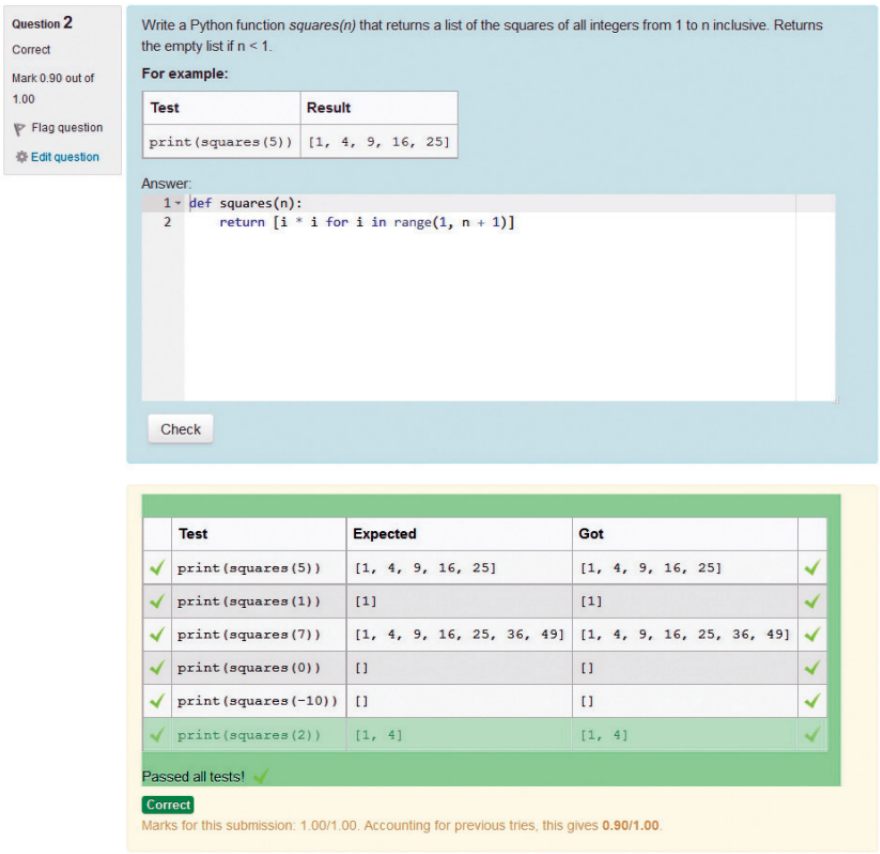


Fonte: (LOBB; HARLOW, 2016, p. 48)

O CodeRunner também apresenta uma abordagem flexível para a criação de perguntas, permitindo que os desenvolvedores de questões criem protótipos personalizados. Esses protótipos podem ser usados para perguntas que exigem funcionalidades adicionais ou para impor restrições, como o uso de determinado tipo de estrutura de controle no código. Além disso, a execução do código do aluno ocorre em um ambiente seguro, isolado do servidor principal, o que evita riscos de segurança.

Apesar de o CodeRunner ter se revelado altamente eficaz em diversas atividades

Figura 15 – CodeRunner - A simples pergunta sobre Python, respondida corretamente



Fonte: (LOBB; HARLOW, 2016, p. 48)

de avaliação, os autores destacaram algumas limitações fundamentais:

(a) O CodeRunner destaca-se em tarefas relativamente simples com especificações claras. Embora, teoricamente, qualquer pergunta com resposta mensurável por um programa de computador possa ser formulada como uma pergunta do CodeRunner, o esforço demandado para criar avaliadores para tarefas complexas muitas vezes inviabiliza essa abordagem, especialmente em turmas de maior tamanho;

(b) Ferramentas de qualidade de código, como o pylint, provaram ser muito valiosas para aumentar a conscientização sobre o estilo e melhorar a qualidade do código, mas ainda ocorrem abusos das regras de estilo, e a qualidade dos comentários e identificadores não pode ser avaliada por computador. Por isso, a avaliação humana ainda é reservada para a qualidade do código;

(c) A resposta a uma pergunta do CodeRunner deve ser apresentada como um único bloco de texto, predominantemente composto por código. Embora os autores das perguntas possam gerar diversos formatos de *feedback*, inclusive gráficos, a apresentação padrão dos resultados é tabular, com uma linha para cada caso de teste e correspondência direta entre a saída esperada e a saída real. Caso

os testes exijam blocos extensos de código ou resultem em saídas volumosas, a visualização dos resultados pode dificultar a compreensão pelos alunos do motivo pelo qual uma resposta foi marcada como incorreta;

(d) A avaliação de perguntas com resultados gráficos representa um desafio. Foi criado uma simulação do kit de ferramentas GUI tkinter em Python para avaliar interfaces gráficas na disciplina, incluindo perguntas que verificam a precisão de gráficos gerados por chamadas à biblioteca de gráficos do Matlab. Contudo, a avaliação da correção de imagens ou mesmo da saída de programas que envolvem gráficos complexos, como os da biblioteca de tartarugas, foi considerada uma tarefa difícil, senão impossível;

(e) Por último, é importante ressaltar que a elaboração de perguntas de qualidade e testes eficazes pode demandar considerável tempo, mesmo em cursos de programação de nível básico. O compartilhamento de bancos de dados de perguntas entre professores seria altamente benéfico nesse contexto.

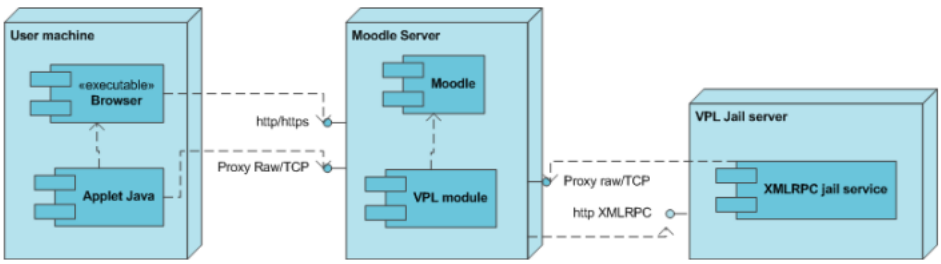
3.4 A VIRTUAL PROGRAMMING LAB FOR MOODLE WITH AUTOMATIC ASSESSMENT AND ANTI-PLAGIARISM FEATURES

Rodríguez-Del-Pino, Royo e Figueroa (2012) descreveram neste artigo o módulo Virtual Programming Lab (VPL) para o Moodle, desenvolvido na Universidade de Las Palmas de Gran Canaria (ULPGC), e lançado para uso gratuito sob a licença GNU/GPL.

Para empregar uma linguagem de programação específica no VPL, basta garantir que o compilador correspondente esteja devidamente instalado no sistema de execução. Atualmente, estão disponíveis sistemas de execução com instalações para diversas linguagens, incluindo Ada, C, C++, C#, FORTRAN, Haskell, Java, Octave, Pascal, Perl, PHP, Prolog, Python, Ruby, Scheme, SQL e VHDL.

O VPL é composto por três elementos: um módulo do Moodle, um editor de código baseado em navegador e um componente Jail, mostrados na Figura 16.

Figura 16 – VPL - Componentes



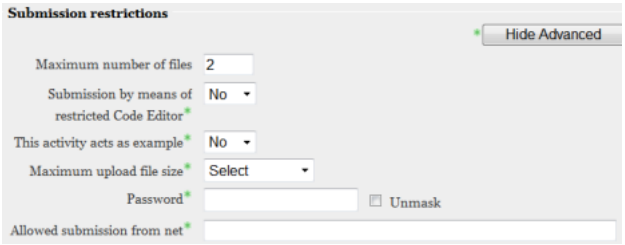
Fonte: (RODRÍGUEZ-DEL-PINO; ROYO; FIGUEROA, 2012, p. 2)

O editor de código, construído em Java, possibilita a edição, execução, depuração e avaliação de programas em um ambiente simples de desenvolvimento. Já o módulo do Moodle oferece características típicas, como backup, integração com o livro de notas e controle de eventos, além de recursos específicos, como gerenciamento de submissões, suporte à avaliação e prevenção contra plágio. E o componente Jail é um servidor que compila e executa códigos de alunos em um ambiente seguro, usando o comando linux chroot para restrições de leitura.

O VPL permite configurar, gerenciar e avaliar atividades de aprendizagem, classificadas por tipo (exemplos, exercícios de cloze, exercícios de desenvolvimento de código) e escopo (tarefas fora da sala de aula ou exames em sala).

A criação de uma atividade VPL inicia-se com o preenchimento do formulário de configuração básica no Moodle, incluindo nome, descrição, período de disponibilidade, opções de avaliação e outras configurações. Além disso, a Figura 17 mostra que é possível especificar detalhes como número máximo de arquivos, tamanho máximo, restrições de edição, rede e senha.

Figura 17 – VPL - Configuração básica de restrições



Fonte: (RODRÍGUEZ-DEL-PINO; ROYO; FIGUEROA, 2012, p. 3)

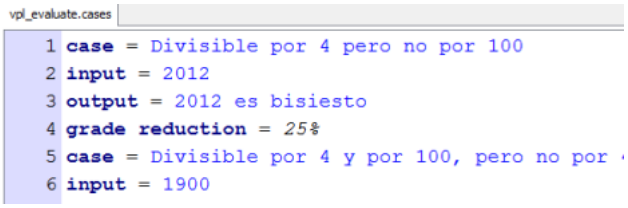
Após preencher o formulário básico, o instrutor pode ajustar cinco grupos adicionais de recursos na ferramenta VPL. A guia "Descrição completa"permite fornecer detalhes sobre o problema a ser resolvido. Em "Casos de teste", é possível configurar testes com descrição, entrada, saída esperada e penalizações em caso de falha. A guia "Options"configura aspectos gerais, como a base da atividade, permissões dos alunos e se os resultados automáticos contam para a nota final. "Arquivos solicitados"determina os nomes obrigatórios dos arquivos a serem enviados.

Essas configurações são suficientes, mas a guia "Advanced"oferece recursos adicionais para testes mais elaborados, como testes de funcionalidade avançados e outros tipos de teste, possibilitando uma avaliação abrangente da qualidade do código.

A Figura 18 mostra um exemplo de configuração de casos de teste, e a Figura 19 mostra a tela de configuração de casos de teste mais avançados.

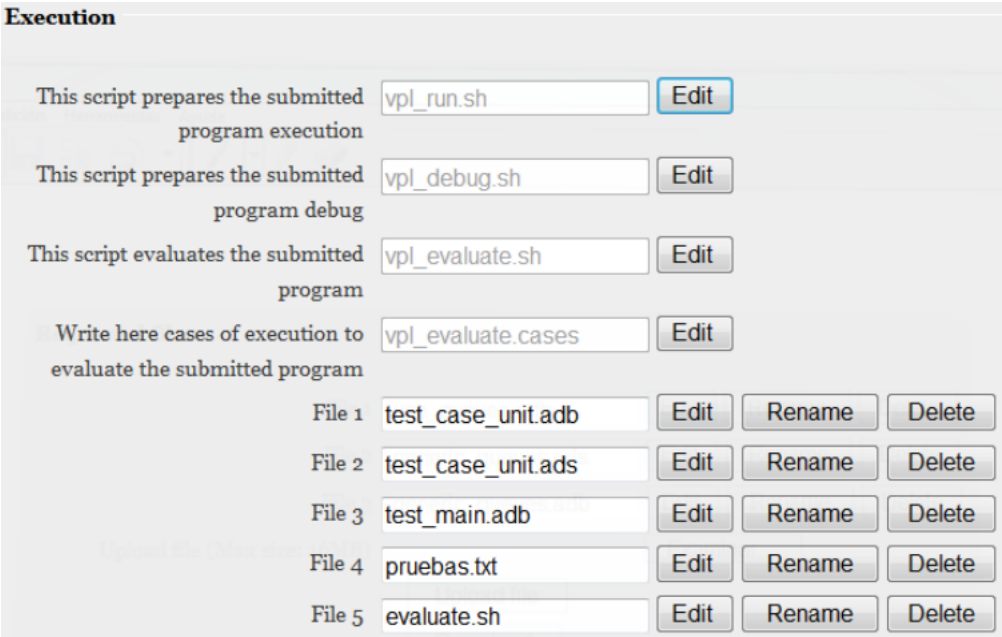
O VPL oferece avaliação automática e assistida por computador, permitindo a personalização através das opções na guia correspondente. É essencial configurar os testes para respaldar a avaliação. O VPL realiza automaticamente os testes configurados, fornecendo um relatório com testes falhos, comentários explicativos e uma

Figura 18 – VPL - Exemplo de configuração de casos de teste



Fonte: (RODRÍGUEZ-DEL-PINO; ROYO; FIGUEROA, 2012, p. 3)

Figura 19 – VPL - Arquivos de execução para testes avançados



Fonte: (RODRÍGUEZ-DEL-PINO; ROYO; FIGUEROA, 2012, p. 4)

proposta de nota. O avaliador humano tem a flexibilidade de ajustar o relatório, excluindo ou adicionando comentários, reutilizando itens da lista e recalculando a nota conforme necessário.

O plágio é abordado pelo VPL através de uma ferramenta de verificação de plágio no código-fonte. Essa ferramenta procura identificar plágio entre os envios de uma tarefa em um curso, considerando também fontes como envios anteriores ou tarefas similares de outros cursos. O processo de identificação de semelhanças entre os arquivos de origem envolve três etapas: tokenização, comparação e clusterização.

A tokenização visa obter uma assinatura normalizada para facilitar a comparação eficiente, envolvendo análise léxica, filtragem e normalização. Esse processo cria uma assinatura do programa, que é uma representação normalizada dos arquivos de código-fonte de um usuário, otimizando o processo de comparação, permitindo a detecção eficaz de plágio.

Em resumo, o Virtual Programming Lab (VPL) apresenta-se como uma ferramenta abrangente para o gerenciamento de atividades de programação. Com recursos

que vão desde a configuração detalhada de atividades no Moodle até a execução de testes avançados e a detecção de plágio, o VPL oferece uma variedade de funcionalidades para apoiar a aprendizagem prática e avaliação de habilidades de programação. Destaca-se, além disso, a necessidade de configurar todos os exercícios e seus casos de teste, para disponibilização destes aos alunos.

3.5 UMA FERRAMENTA BASEADA EM JUÍZES ONLINE PARA O APOIO ÀS ATIVIDADES DE PROGRAMAÇÃO DE COMPUTADORES NO MOODLE

No projeto conduzido por Chaves *et al.* (2013), foi desenvolvido o Módulo de Integração com Juízes Online (MOJO), ferramenta que integra os Juízes Online, como o Timus Online Judge e o Beecrowd (conhecido como URI Online Judge na época) ao Moodle, para evitar que os professores de disciplinas de programação ficassem sobrecarregados ao ter que elaborar, submeter, abaliar e fornecer o *feedback* necessário das questões aos alunos. O MOJO automatizou o processo de Elaboração, Submissão e Avaliação (ESA) das atividades de programação, fornecendo um ambiente unificado que facilita o acompanhamento dos alunos e amplia o acesso a um maior número de questões.

O MOJO, portanto, integra o Moodle com os Juízes Online para atividades de programação, sem interação direta entre as plataformas. Ele automatiza o processo ESA (Elaboração, Submissão e Avaliação), permitindo que professores submetam questões no Moodle, estudantes enviem suas respostas, e os Juízes Online avaliem automaticamente o código, com os resultados retornando ao Moodle para análise de professores e alunos. O professor pode monitorar o progresso dos estudantes e acessar os códigos submetidos, enquanto o Repositório de Integração armazena as questões para uso futuro.

Na arquitetura geral do MOJO, o Moodle gerencia a interface e o controle das atividades de programação, enquanto o MOJO se encarrega da comunicação e interação com os Juízes Online. A arquitetura conta com o Módulo Principal (MOP), responsável pelas funcionalidades essenciais e pelo controle do Módulo de Carga e Atualização (MOCA), que, por sua vez, realiza o carregamento inicial e atualizações constantes das questões, armazenando-as no Repositório de Integração. Esse repositório mantém as questões e resultados para uso e acompanhamento contínuo no Moodle.

Como os juízes online não oferecem APIs ou serviços Web, foram necessárias implementações específicas para cada juiz, utilizando PHP (a mesma linguagem do Moodle) e JavaScript para a interface. O banco de dados PostgreSQL foi escolhido por ser open-source e por permitir a criação de tabelas personalizadas para armazenar dados dos alunos, viabilizando um *feedback* direcionado e adequado.

O objetivo do MOJO é reduzir a carga de trabalho dos professores no processo ESA, permitindo um acompanhamento mais próximo dos alunos. Em uma pesquisa

inicial com sete professores, os resultados apontaram que 71,4% acreditam que a ferramenta pode economizar tempo; 28,6% gostariam de poder criar e avaliar suas próprias atividades automaticamente; 42,8% mencionaram questões de legibilidade do código; 28,6% observaram a clareza dos resultados; 85,7% valorizam o *feedback* mais rápido aos alunos; e 100% acreditam que a redução de tarefas possibilitará um acompanhamento mais eficaz, especialmente para alunos com dificuldades.

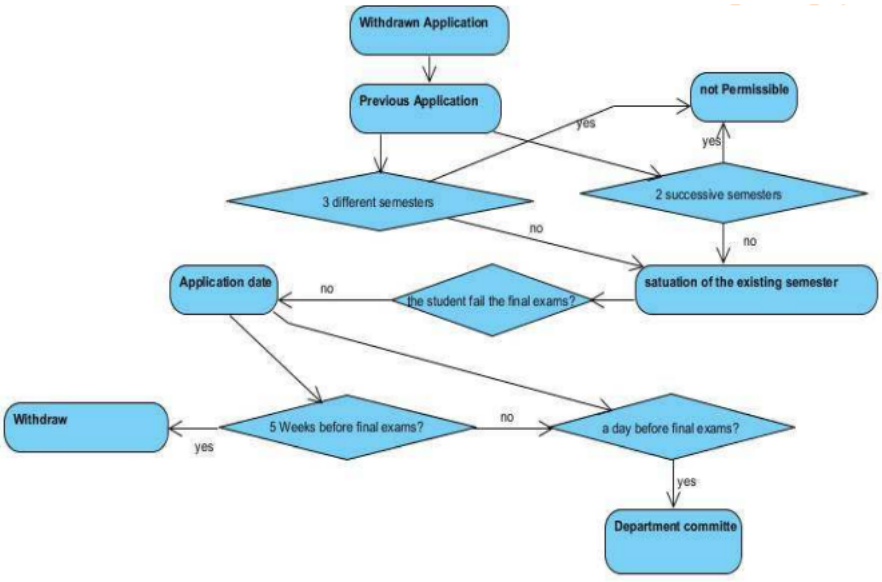
3.6 BUILDING OF A RULE-BASED EXPERT SYSTEM FOR ACADEMIC ADVISING VIA WEB EXPERT SYSTEM TOOLS

O projeto conduzido por NASR, TALAB e AL-GAHTANI (2018) aborda o desenvolvimento de um sistema especialista voltado para otimizar o aconselhamento acadêmico na Universidade King Khalid. A proposta central do estudo é criar uma ferramenta que simule as funções do conselheiro acadêmico humano, utilizando técnicas de inteligência empresarial para auxiliar na tomada de decisões informadas por estudantes e funcionários, baseadas nas regulamentações acadêmicas da instituição.

O processo de desenvolvimento do sistema é dividido em várias fases, cada uma com objetivos específicos:

Fase de Análise: Esta etapa inicial é dedicada à coleta de informações sobre as questões mais recorrentes entre os estudantes, além de procedimentos acadêmicos frequentes, como matrícula em cursos, transferências de departamento e pedidos de suspensão. A análise dessas necessidades foi realizada por meio de entrevistas com estudantes e conselheiros acadêmicos, utilizando uma abordagem de árvore de decisão (UML), mostrada na Figura 20, para entender melhor os requisitos do sistema.

Figura 20 – Sistema especialista baseado em regras - Árvore de decisão



Fonte: (NASR; TALAB; AL-GAHTANI, 2018)

Fase de Design: Com base na análise anterior, o design do sistema foi estruturado utilizando uma metodologia baseada em regras (rule-based). Esse método foi escolhido por sua adequação à modelagem dos processos de aconselhamento acadêmico, que envolvem uma lógica condicional do tipo "se-então". Durante essa fase, também foi projetada uma interface amigável, alinhada com o estilo visual e os padrões de interação do portal da universidade, garantindo que os usuários pudessem facilmente se adaptar ao sistema.

Fase de Implementação: A construção do sistema foi realizada utilizando o ES Builder, uma ferramenta de código aberto que facilita a criação de sistemas especialistas baseados em regras. O sistema foi projetado para ser acessível via plataforma web, o que possibilita seu uso por uma grande variedade de usuários, incluindo estudantes e conselheiros acadêmicos.

Fase de Testes e Avaliação: O sistema foi testado com diferentes cenários, utilizando tanto estudantes quanto conselheiros acadêmicos, a fim de validar sua eficácia. Os resultados indicaram que o sistema é capaz de fornecer respostas adequadas e de auxiliar no processo de aconselhamento de forma precisa e eficiente.

O estudo conclui que a implementação de sistemas especialistas no âmbito acadêmico tem um grande potencial para melhorar a eficiência do aconselhamento acadêmico, promovendo uma interação mais eficaz entre estudantes e conselheiros. Além disso, o sistema proposto pode se tornar uma ferramenta valiosa para a tomada de decisões informadas, não apenas para os estudantes, mas também para os docentes, que poderão fornecer orientações mais precisas e baseadas em dados. O sistema mostrou-se alinhado às expectativas dos usuários, que manifestaram interesse em expandir sua funcionalidade para cobrir mais áreas do aconselhamento acadêmico.

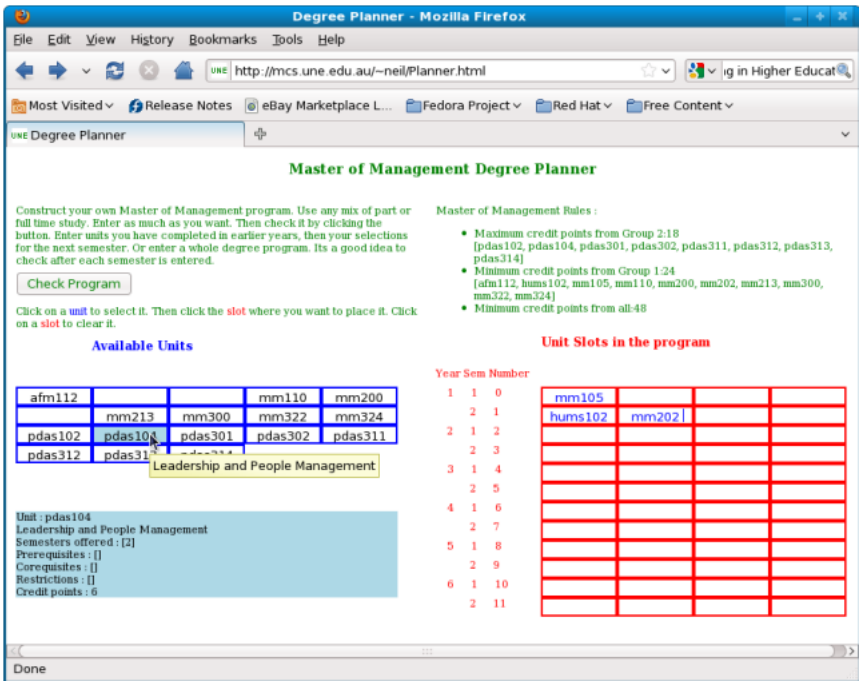
Este trabalho evidencia o papel transformador que a tecnologia pode desempenhar no setor educacional, especialmente no contexto do aconselhamento acadêmico, apontando para a necessidade de continuar a pesquisa e o desenvolvimento de sistemas dessa natureza para atender de forma mais abrangente às necessidades dos estudantes e professores.

3.7 AN INTERACTIVE WEB-BASED EXPERT SYSTEM DEGREE PLANNER

O artigo desenvolvido por Dunstan (2013) descreve o desenvolvimento de um planejador de grau acadêmico baseado na web (seu layout é mostrado na Figura 21), utilizando XML (Extensible Markup Language) e sistemas especialistas. O sistema possui um servidor que hospeda arquivos XML para cada grau acadêmico, descrevendo unidades e regras associadas. Essas regras são importadas para uma interface web genérica, permitindo a personalização do visual com uma paleta de unidades e horários disponíveis para os semestres. As regras do grau acadêmico em formato XML são convertidas em código Prolog, que permite ao sistema responder a consultas

complexas relacionadas ao planejamento de cursos.

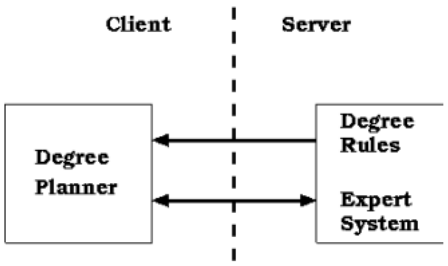
Figura 21 – Planejador de grau acadêmico - Layout da página



Fonte: (DUNSTAN, 2013)

O design do sistema é baseado na separação clara entre o tratamento da interface do usuário no cliente e o processamento de regras no servidor (Figura 22). O servidor utiliza um sistema especialista em Prolog para resolver consultas mais complexas, enquanto o cliente lida com informações mais simples por meio de JavaScript. O sistema é acessível via navegador e não requer plugins específicos, sendo adequado para dispositivos móveis, como smartphones.

Figura 22 – Planejador de grau acadêmico - Interação cliente-servidor



Fonte: (DUNSTAN, 2013)

A interface do usuário é composta por várias seções, incluindo uma paleta de unidades, um cronograma de programa semestral e um painel de exibição de detalhes das unidades selecionadas. Os estudantes podem explorar opções de inscrição e verificar se suas escolhas atendem aos requisitos do curso. O sistema permite que os

alunos ajustem seu programa acadêmico, respondendo a perguntas sobre requisitos de unidades e pré-requisitos, de forma interativa e personalizada.

Conclui-se que o uso de sistemas especialistas em planejadores de grau acadêmico baseados na web pode melhorar a interatividade e fornecer orientações mais precisas aos alunos, ajudando-os a construir um programa de graduação adequado aos seus interesses e cronograma, sem a necessidade de plugins ou instalações específicas.

3.8 COMPARAÇÃO ENTRE OS TRABALHOS

O estudo de (CRUZ *et al.*, 2022) ressalta a eficácia da integração da plataforma Beecrowd em um plano de ensino inovador para disciplinas de programação. Ao incorporar estratégias como sala de aula invertida e gamificação, a pesquisa conclui que o Beecrowd, utilizado como suporte para a gamificação, não só melhora o processo de ensino e aprendizagem, oferecendo suporte aos educadores e motivando os alunos, mas também contribui para a redução da evasão em cursos superiores de informática.

A aplicação das metodologias propostas no estudo dos autores demonstrou um impacto significativo no desempenho dos alunos ao longo do tempo na universidade dos autores, resultando em um crescimento notável e consistente nas notas. A ferramenta Beecrowd, destacada no estudo, é elogiada pela sua organização meticulosa de desafios em categorias e níveis de dificuldade, além de sua abordagem de maratona de programação, revelando-se eficaz em envolver os alunos de maneira imersiva.

A Tabela 1 apresenta uma comparação detalhada das vantagens do Beecrowd em relação a outras ferramentas integradas ao Moodle, como BOCA, VPL e CodeRunner, conforme discutido nos trabalhos relacionados. O objetivo é analisar os benefícios específicos do uso do Beecrowd, oferecendo uma base sólida para este estudo. Vale ressaltar que este trabalho propõe o incentivo ao uso da integração entre Moodle e Beecrowd por meio de LTI. Em contraste, o BOCA e o CodeRunner funcionam como tipos de questões no Moodle, enquanto o VPL é um módulo próprio da plataforma.

Também é importante lembrar que, enquanto o Beecrowd e o VPL foram desenvolvidos para apoiar o ensino, o BOCA foi desenvolvido para a avaliação de submissões de código em maratonas de programação.

A análise da tabela destaca o Beecrowd como notavelmente diferenciado em relação aos outros sistemas, sobretudo devido à presença de questões pré-existentes para os professores. Essa característica elimina a necessidade de cadastramento manual das questões de programação e seus respectivos testes, uma exigência presente nas plataformas BOCA, VPL e CodeRunner. Como confirmação, uma das limitações do CodeRunner apontadas por Lobb e Harlow (2016) é o fato de a criação de perguntas e testes de qualidade demandar tempo, além de ser mais adequado para tarefas simples e bem definidas, já que a criação de testes para atividades complexas exige muito

Tabela 1 – Comparação entre quatro integrações de sistemas, que auxiliam no ensino da programação, com o Moodle: Beecrowd, BOCA, CodeRunner e VPL

Aspecto	**Beecrowd**	**BOCA**	**VPL**	**CodeRunner**
Possui Juiz Online	X	X	X	X
Fornece *feedback* Imediato	X	X	X	X
Criação Manual de Questões		X	X	X
Questões Prontas para Professores	X			
Questões Prontas classificadas por categoria	X			
Inserção Manual de Casos de Teste		X	X	X
Suporta Diversas Linguagens	X	X	X	X
Pré-processamento de Submissões				X
Comentários de Testes Falhados			X	
Prevenção de Plágio	X		X	
Incentiva participar de maratonas de programação	X			

Fonte: Produzido pela autora.

esforço, especialmente em turmas grandes.

Um aspecto distintivo e relevante nos sistemas BOCA, VPL e CodeRunner é a abordagem de desenvolvimento que incorpora a utilização de servidores separados para o Moodle e os sistemas juízes online. Essa estratégia visa garantir a execução independente e eficiente de cada componente, proporcionando benefícios tanto em termos de desempenho quanto de segurança.

A adoção de servidores separados para o Moodle e os sistemas juízes online tem como objetivo otimizar a distribuição de carga, prevenindo sobrecargas e garantindo uma operação mais estável e responsiva. Essa separação também fortalece a segurança, pois ao isolar cada plataforma, diminui-se os riscos de conflitos e vulnerabilidades. Da mesma forma, ela possibilita uma escalabilidade mais eficiente, permitindo que cada sistema seja dimensionado conforme suas necessidades específicas, sem afetar o desempenho do outro. Isso é particularmente importante em ambientes acadêmicos, onde a demanda por plataformas e sistemas pode variar significativamente.

O trabalho de Rodríguez-Del-Pino, Royo e Figueroa (2012) mostra que o módulo Virtual Programming Lab (VPL) possui um editor de código em Java para editar, executar, depurar e avaliar programas em um ambiente simples de desenvolvimento. Dessa forma, para utilizar todas as funcionalidades é necessário um navegador com suporte a JavaScript e applets Java. Já os trabalhos de Lobb e Harlow (2016) e Galasso e Moreira (2014) mostram que tanto o CodeRunner quanto o BOCA disponibilizam caixas de texto nas quais os usuários podem escrever ou colar o código, permitindo sua subsequente avaliação.

Neste trabalho, a integração LTI fornecida pelo Beecrowd será utilizada, de modo que a criação de listas de exercícios, a resolução de exercícios pelos alunos e a correção das soluções pelo juiz online ocorrerão diretamente nos servidores do Beecrowd.

Além disso, a prevenção contra plágio é uma característica crucial e distintiva tanto no Beecrowd quanto no VPL, destacando-se como um diferencial significativo dessas plataformas. Essa funcionalidade tem a intenção de salvaguardar a integridade acadêmica, visando a autenticidade e a originalidade dos trabalhos submetidos pelos alunos.

No contexto do Beecrowd, de acordo com (CRUZ *et al.*, 2022), a prevenção contra plágio pode envolver mecanismos avançados de análise de código-fonte, identificando padrões suspeitos que indicam possível cópia indevida. Além disso, o sistema pode empregar algoritmos e técnicas que comparam soluções de diferentes alunos, buscando similaridades que ultrapassem limites aceitáveis de coincidência.

No caso do VPL, de acordo com Rodríguez-Del-Pino, Royo e Figueroa (2012), a prevenção contra plágio pode ser implementada através de verificações rigorosas durante o processo de submissão. Isso pode incluir a comparação automática de códigos submetidos em busca de trechos idênticos ou substancialmente semelhantes. Além disso, o VPL pode adotar métodos de análise mais avançados, como a detecção de técnicas de programação específicas que são características de trabalhos plagiados.

O projeto de Chaves *et al.* (2013) descreve o desenvolvimento do Módulo de Integração com Juízes Online (MOJO), uma ferramenta criada para integrar juízes *online*, como o Beecrowd (anteriormente URI Online Judge), ao Moodle. O MOJO visa delegar ao Moodle o gerenciamento da interface e do controle das atividades de programação, enquanto centraliza a comunicação e interação com os juízes online. Devido à ausência de APIs ou serviços web nos juízes à época, foram necessárias implementações específicas para cada plataforma, utilizando PHP (a mesma linguagem do Moodle) e JavaScript para a interface. O banco de dados PostgreSQL foi adotado por ser open-source e pela flexibilidade de criação de tabelas personalizadas para armazenamento de dados dos alunos, o que possibilita um *feedback* direcionado e preciso.

Este trabalho, contudo, utilizará a integração LTI fornecida pelo Beecrowd, a qual facilita a interação de docentes e alunos com a plataforma. Além disso, será disponibilizado um manual para orientar o uso da LTI, juntamente com um sistema especialista que auxiliará os alunos na resolução das questões e os docentes no esclarecimento de dúvidas dos estudantes.

É relevante notar que 100% dos professores entrevistados para avaliar a eficácia do MOJO acreditam que a redução de tarefas permitirá um acompanhamento mais eficaz dos estudantes, especialmente daqueles que apresentam maiores dificuldades.

Por fim, os projetos de NASR, TALAB e AL-GAHTANI (2018) e Dunstan (2013) abordam a construção de sistemas especialistas para o aconselhamento e planejamento acadêmico, porém apresentam enfoques e abordagens técnicas distintas.

No primeiro trabalho, NASR, TALAB e AL-GAHTANI (2018) desenvolve um sistema especialista baseado em regras para o aconselhamento acadêmico na Universidade de King Khalid. Este sistema é focado em replicar o processo de tomada de decisão dos conselheiros acadêmicos humanos, utilizando uma metodologia de árvore de decisão, implementada através do ES Builder, uma ferramenta de código aberto para sistemas especialistas baseados em regras, com uma interface web simples e uma lógica de "se-então". Essa escolha reflete a necessidade de simular de forma prática e direta o trabalho dos conselheiros humanos sem uma complexidade computacional elevada.

No segundo trabalho, Dunstan (2013) foca em um planejador de grau acadêmico, utilizando uma abordagem híbrida com XML e Prolog para gerenciar e responder a consultas complexas. As regras para cada programa acadêmico são estruturadas em XML e convertidas em Prolog, possibilitando que o sistema lide com consultas interativas e personalizadas. A arquitetura desse sistema é distribuída, com o cliente manipulando a interface e o servidor processando as regras em Prolog. Essa separação facilita o uso em dispositivos móveis e oferece flexibilidade ao permitir que os alunos construam e verifiquem seu programa acadêmico de maneira independente e intuitiva, sem plugins específicos.

Já neste trabalho, será utilizado o Prolog como servidor da aplicação, responsável pelo gerenciamento de requisições HTTP, armazenamento de dados de sessão e comunicação contínua com o front-end. O servidor proporcionará um ambiente dinâmico para o carregamento modular de arquivos específicos de cada questão, permitindo a adaptação das perguntas e a personalização das respostas conforme as interações do usuário. Além disso, configurará permissões de CORS para integrar facilmente com a interface web, que será desenvolvida utilizando o React, uma biblioteca JavaScript para construção de interfaces de usuário interativas e reativas.

4 DESENVOLVIMENTO

Nesta seção, será abordada a LTI do Beecrowd, que simplifica a integração entre o Moodle e a plataforma Beecrowd. Serão apresentados também os detalhes da implementação da aplicação web do sistema especialista, desenvolvida para auxiliar professores e alunos no esclarecimento de dúvidas recorrentes sobre as questões do Beecrowd, facilitando a adaptação de ambos ao uso da plataforma.

4.1 MANUAIS DE CONFIGURAÇÃO E USO DA LTI BEECROWD

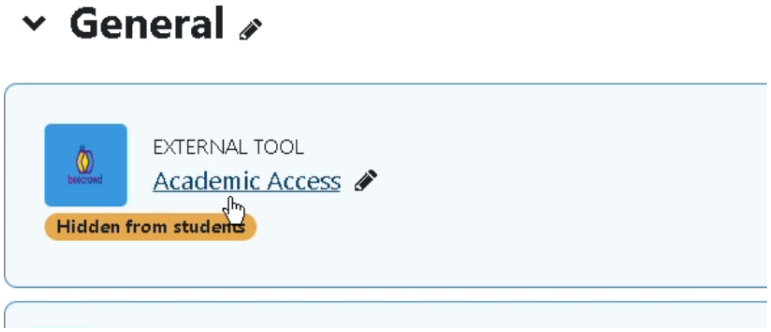
Com a disponibilização de uma ferramenta LTI pelo Beecrowd, visando facilitar sua integração com o Moodle, foi elaborado um manual detalhado para orientar o administrador do Moodle na configuração dessa LTI (Apêndice A).

Para configurar uma LTI externa, o administrador deve acessar "Administração do *site*", selecionar "Plugins"e, em "Módulos de atividade", escolher "Ferramenta externa"e clicar em "Gerenciar ferramentas". Em seguida, deve clicar em "Configurar uma ferramenta manualmente"e preencher os campos conforme as instruções especificadas no Manual de Configuração da LTI Beecrowd no Moodle (Apêndice A). Após a configuração, será possível visualizar os detalhes da ferramenta, os quais deverão ser enviados ao Beecrowd para estabelecer a comunicação entre as plataformas.

Além disso, foi desenvolvido um manual de uso da LTI Beecrowd (Apêndice B), direcionado aos professores que desejam utilizar o Beecrowd como suporte educacional em sala de aula. Esse manual instrui os docentes sobre como criar uma atividade do Beecrowd no Moodle, orientar o acesso dos estudantes e transferir as notas obtidas no Beecrowd para o Moodle.

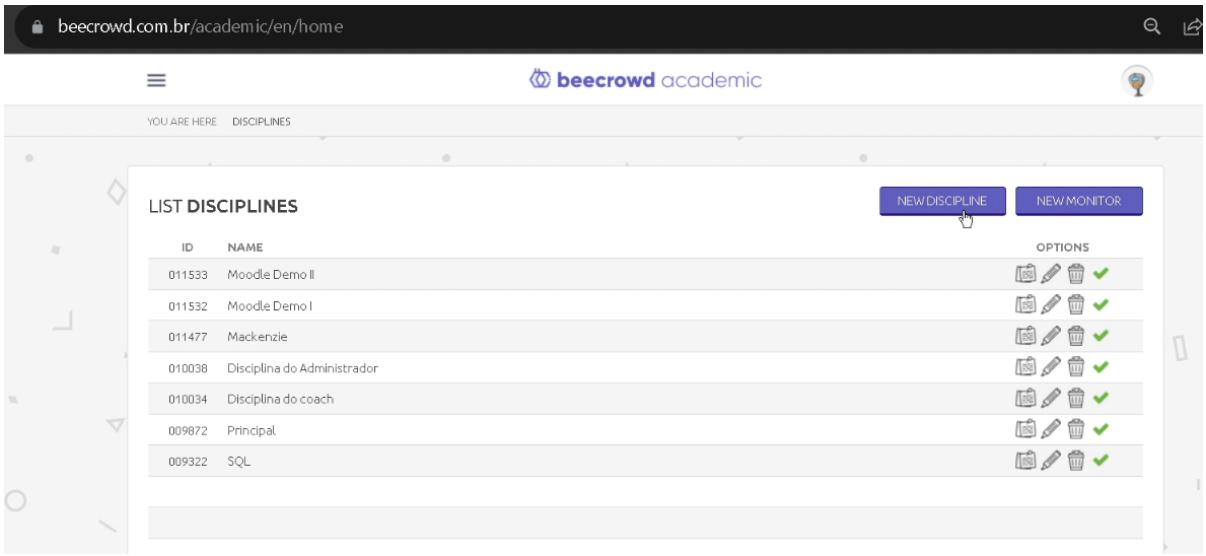
Após seguir o guia do Manual de Uso da LTI Beecrowd (Apêndice B), duas atividades ficam disponíveis na página do curso, apresentadas como "botões"visuais. A primeira atividade, que é ocultada para os estudantes (Figura 23), permite ao professor acessar o Beecrowd Academic (Figura 24) para configurar disciplinas, criar listas de exercícios para os alunos e enviar as notas para o Moodle.

Figura 23 – Atividade para o professor acessar o Beecrowd Academic



Fonte: Produzido pela autora.

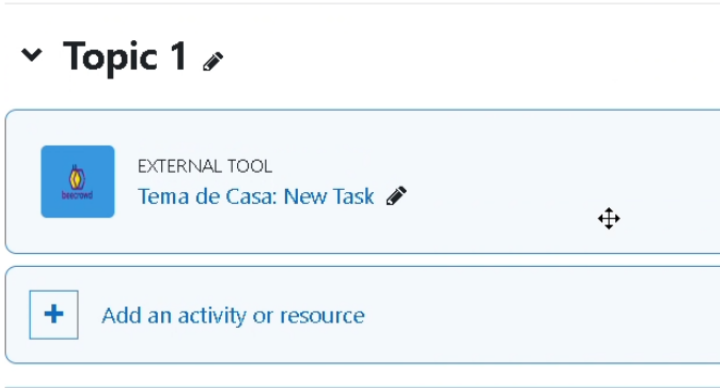
Figura 24 – Beecrowd Academic



Fonte: Produzido pela autora.

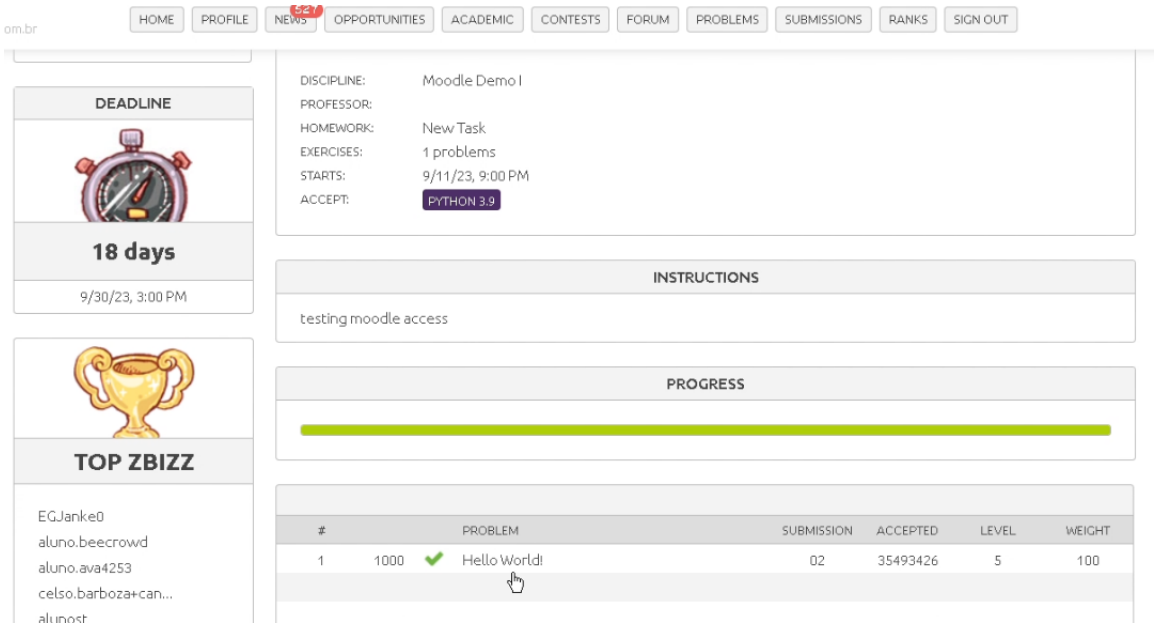
A segunda atividade, acessível aos estudantes (Figura 25), redireciona-os diretamente para a página no Beecrowd da lista de exercícios criada pelo professor (Figura 26), sem necessidade de cadastro ou login. Assim, os estudantes podem resolver as atividades diretamente a partir dessa página. Esse botão também permite ao professor acessar a lista de exercícios criada, podendo verificar o progresso dos alunos, as tentativas realizadas e, quando desejado, enviar as notas para o Moodle.

Figura 25 – Atividade para o aluno acessar o Beecrowd



Fonte: Produzido pela autora.

Figura 26 – Beecrowd do aluno



Fonte: Produzido pela autora.

4.2 SISTEMA ESPECIALISTA WEB

O objetivo do sistema especialista é oferecer suporte na resolução de dúvidas recorrentes dos estudantes em relação a questões do Beecrowd. A proposta é desenvolver uma aplicação acessível aos alunos via navegador, com um sistema de *chat* onde o aluno pode informar a questão sobre a qual necessita orientação. O sistema, então, conduz o aluno por meio de perguntas binárias (sim ou não) e, conforme as respostas, fornece dicas relevantes relacionadas ao exercício em questão.

Para viabilizar essa funcionalidade, o sistema especialista será estruturado com um conjunto de dados contendo perguntas direcionadas aos alunos e respostas predefinidas que orientam o aluno na resolução das atividades específicas da disciplina. Em disciplinas como Programação Orientada a Objetos I, onde as listas de exercícios frequentemente incluem questões já utilizadas em semestres anteriores, observa-se uma repetição de problemas e, consequentemente, de dúvidas recorrentes dos alunos.

A primeira etapa deste trabalho foi dedicada à coleta de dados sobre as questões, com o objetivo de identificar e catalogar as dúvidas mais recorrentes. Na segunda etapa, foram definidos os requisitos funcionais, os requisitos não funcionais e as regras de negócio da aplicação. Em seguida, na terceira etapa, realizou-se uma pesquisa sobre as tecnologias que seriam utilizadas no desenvolvimento. Na quarta etapa, foram elaborados os diagramas que definiram a estrutura e o funcionamento da aplicação. Por fim, a partir da quinta etapa, deu-se início ao desenvolvimento propriamente dito.

4.2.1 Coleta de Dados

Nesta fase, foram realizadas duas coletas de dados: uma com dicas de resolução fornecidas pelo professor para uma lista de exercícios, e outra com as dúvidas dos alunos sobre o uso de listas em Python, com foco em questões específicas.

Na primeira coleta, professor da disciplina de Programação Orientada a Objetos I, da Universidade Federal de Santa Catarina (UFSC), forneceu dicas de resoluções para cada questão de uma lista de exercícios que ainda não tinha sido repassada aos alunos, com o objetivo de avaliar a utilidade da aplicação após ela ser desenvolvida. Assim, quando os alunos fossem resolver a lista de exercícios, poderiam testar a eficácia da aplicação. As questões abordadas, que receberam as dicas de solução, incluem: 1261, 1281, 1430, 1449, 1483, 1763, 1991, 1953, 2091, 2482, 2492, 2654, 2949 e 2987.

Na segunda coleta, foram registradas as dúvidas dos alunos da disciplina de Programação Orientada a Objetos I, da UFSC, coletadas no dia 30 de outubro de 2024. Nessa data, os alunos estavam trabalhando em uma lista de exercícios sobre o uso de listas em Python. As questões abordadas incluíram os problemas de números 1187, 1435, 1715, 2436, 1383, 1184, 1181 e 1185. Um aspecto notável foi que, durante essa mesma aula, quatro alunos apresentaram a mesma dúvida em relação à questão 1435, especificamente perguntando: "Como resolvo essa questão?"

Durante as 2h30 de aula, a questão 1435 foi a que suscitou o maior número de dúvidas entre os estudantes. Em uma turma de 26 alunos – embora nem todos tenham alcançado essa questão no tempo da aula –, quatro solicitaram uma explicação geral sobre a abordagem para resolvê-la, enquanto dois buscaram orientação sobre a conversão dessa abordagem para o código. Adicionalmente, três alunos enfrentaram o erro "Presentation Error"e um aluno recebeu o erro "Run-time Error".

As dúvidas em relação à questão 1181 refletiram dificuldades comuns no uso de laços de repetição e na manipulação de matrizes. Muitos estudantes não obtinham a resposta correta, mesmo com uma solução aparentemente desenvolvida, devido à confusão na seleção dos elementos da matriz, com erros ao iterar pelas linhas em vez das colunas. Os alunos também enfrentaram dificuldades na estruturação do laço de repetição, especialmente ao calcular a soma dos elementos de uma linha específica e ao determinar o divisor correto para o cálculo da média. Problemas de formatação, que resultaram no erro "Presentation Error", também foram recorrentes, sendo necessário ajustar a saída para uma casa decimal com ’

Na questão 1184, as dificuldades estavam relacionadas à estruturação do laço de repetição para percorrer corretamente os elementos abaixo da diagonal principal da matriz. Os alunos foram orientados a usar um laço duplo, com a sequência correta de for i in range(0, 12) e for j in range(0, i). Além disso, houve dúvidas sobre a exclusão dos elementos da diagonal principal, que não deveriam ser somados. Como na questão

1181, os alunos enfrentaram dificuldades de formatação, com orientações para ajustar a saída utilizando ’

As dúvidas relativas à questão 1185 envolveram a manipulação de elementos acima da diagonal secundária da matriz, com os alunos sendo orientados a percorrer as linhas até o último índice da coluna, subtraindo o número da linha. A configuração correta dos laços de repetição foi outra dúvida comum, com muitos alunos invertendo os laços, o que afetou os resultados. A correção do divisor para o cálculo da média e os problemas de formatação também foram pontos críticos, com instruções para usar ’

A questão 1187 gerou dúvidas relacionadas aos laços de repetição necessários para somar os elementos da matriz, com os alunos questionando sobre o ajuste dos índices para evitar a soma dos elementos das diagonais. Também surgiram dúvidas sobre como os laços percorriam as linhas e colunas para formar a região triangular desejada, além de questionamentos sobre a lógica para calcular a média e evitar erros de divisão.

A questão 1383, sobre as regras do Sudoku, suscitou dúvidas sobre como validar se uma solução estava correta, além de dificuldades na transformação da lógica para código. A orientação dada foi garantir a não repetição de números em cada linha, coluna, e bloco 3x3, com orientações sobre o uso de depuração com o udebug e Thonny.

A questão 1715 gerou dúvidas sobre a multiplicação dos elementos das linhas para identificar jogadores que marcaram gols em todas as partidas. Os alunos tiveram dificuldades em entender como aplicar esse método corretamente e como garantir que o contador fosse atualizado. A depuração no udebug e Thonny foi útil para a maioria dos alunos, que ajustaram seus códigos após o *feedback*.

Por fim, a questão 2465 também gerou dúvidas em relação ao uso do comando while True para verificar os vizinhos da matriz. A principal dificuldade foi evitar que o robô retornasse à posição anterior após avançar para um vizinho com valor ’1’, sendo sugerido que a posição fosse marcada como zero antes de seguir para o próximo vizinho. As orientações permitiram que os alunos avançassem na resolução do problema, superando as dificuldades iniciais.

4.2.2 Requisitos da Aplicação

Esta seção descreve os requisitos essenciais para o desenvolvimento da aplicação, divididos em Regras de Negócio, Requisitos Funcionais e Não Funcionais. As Regras de Negócio definem os processos do sistema, enquanto os Requisitos Funcionais especificam as funcionalidades necessárias. Já os Requisitos Não Funcionais tratam de aspectos de qualidade, como desempenho, segurança e arquitetura do sistema.

4.2.2.1 Regras de Negócio

Esta subseção detalha as Regras de Negócio que definem os processos centrais e orientam o comportamento do sistema. Essas regras são essenciais para garantir que o sistema atenda aos objetivos específicos e forneça uma experiência funcional e direcionada aos usuários.

RN1 – **Identificação da Questão**: O usuário deve informar o código da questão do Beecrowd que está tentando resolver para que o sistema possa fornecer dicas específicas.

RN2 – **Perguntas Binárias**: O sistema deve formular perguntas binárias (sim/não) para guiar o usuário, baseando-se em possíveis erros ou dificuldades comuns para cada questão.

RN3 – **Personalização de Orientações**: Com base nas respostas do usuário, o sistema deve ajustar as perguntas para fornecer a dica mais apropriada.

RN4 – **Encerramento do Fluxo**: Ao atingir um número máximo de perguntas, o sistema deve fornecer uma dica final ao usuário.

4.2.2.2 Requisitos Funcionais

Os Requisitos Funcionais descrevem as funcionalidades que o sistema deve possuir para cumprir os objetivos propostos. Esses requisitos representam ações e processos que os usuários podem realizar na aplicação, garantindo que ela seja funcional e eficaz.

RF1 – **Comunicação por Chat**: usuário deve interagir com o sistema por uma interface de *chat* intuitiva, que apresente respostas sequenciais.

RF2 – **Busca e Seleção de Questão**: O sistema deve permitir que o usuário informe o código da questão do Beecrowd que deseja resolver.

RF3 – **Fluxo de Perguntas Binárias**: O sistema apresenta perguntas binárias adaptadas conforme as respostas do usuário.

RF4 – **Sistema de Dicas**:Após a análise das respostas, o sistema deve oferecer dicas relevantes para o problema informado.

RF5 – **Instruções sobre Erros Comuns do Beecrowd**: O sistema deve ter uma seção que informa ao usuário descrições dos principais erros retornados pelo Beecrowd, contidos na seção 2.2.

4.2.2.3 Requisitos Não Funcionais

Nesta subseção, são apresentados os Requisitos Não Funcionais, que especificam características de qualidade e restrições que o sistema deve atender. Eles incluem aspectos como desempenho, usabilidade, segurança e arquitetura, garantindo que o sistema não apenas funcione, mas também ofereça uma experiência eficiente e confiável aos usuários.

RNF1 – **Desempenho**:

- O sistema deve responder em tempo real a cada interação do usuário, mantendo a fluidez do *chat*.

RNF2 – **Usabilidade**:

- O sistema deve ser intuitivo e fácil de usar, com uma interface de *chat* amigável.
- Deve ser compatível com os principais navegadores modernos.

RNF3 – **Escalabilidade**:

- O sistema deve suportar o uso simultâneo de múltiplos usuários.

RNF4 – **Segurança**:

- As sessões de *chat* devem ser isoladas, para evitar vazamento de informações entre os usuários.

RNF5 – **Manutenibilidade**:

- O sistema deve ter um código modular, facilitando a inclusão de novas perguntas e dicas para futuras questões do Beecrowd.

RNF6 – **Arquitetura Backend e Frontend**:

- O sistema deve ter um *backend* desenvolvido em Prolog, utilizando SWI-Prolog para lidar com requisições HTTP.
- O *backend* deve definir configurações de CORS apropriadas para permitir a comunicação entre cliente e servidor.
- O *backend* deve seguir o padrão de API REST e devolver respostas no formato JSON.
- O *backend* deve conter o sistema especialista, que processa as perguntas e respostas para oferecer suporte aos estudantes.
- O *frontend* deve ser desenvolvido em React e TypeScript, proporcionando uma interface interativa e amigável para os alunos.

4.2.3 Tecnologias Usadas

4.2.3.1 Prolog e SWI-Prolog

Para o desenvolvimento do *backend* da aplicação, utilizou-se a linguagem Prolog, com o auxílio da implementação SWI-Prolog. O SWI-Prolog é uma versão amplamente utilizada do Prolog, que inclui uma série de bibliotecas essenciais para a manipulação de requisições HTTP e dados JSON, como http/thread_httpd, http/http_dispatch, http/json, http/http_session, entre outras. Essas bibliotecas permitem a configuração de um servidor HTTP para o processamento de requisições web e a manipulação de dados JSON. Além disso, elas possibilitam o gerenciamento de sessões, o que viabiliza a integração com aplicações externas e permite o uso simultâneo da aplicação por diferentes usuários (WIELEMAKER, 2024).

4.2.3.2 API REST

O servidor do backend desse trabalho definiu os *endpoints* HTTP (/server e /questions) para o recebimento de requisições POST e GET, permitindo a interação com o *backend* por meio de uma arquitetura baseada em REST.

O termo REST (Representational State Transfer) é um estilo de comunicação baseado em padrões e protocolos da web, como HTTP, que permite a troca de dados entre sistemas de forma escalável e eficiente. Nesse modelo, as requisições são realizadas utilizando os métodos HTTP padrão (como GET, POST, PUT e DELETE), facilitando a comunicação entre sistemas de maneira eficiente e escalável (MASSE, 2011).

4.2.3.3 JSON

JSON (JavaScript Object Notation) é um formato leve e baseado em texto para intercâmbio de dados, independente de linguagem de programação. Derivado do padrão ECMAScript, o JSON define um conjunto simples de regras de formatação para a representação portátil de dados estruturados. Ele busca remover inconsistências com outras especificações e oferece orientações para melhorar a interoperabilidade entre sistemas (BRAY, 2017).

A biblioteca http/http_json do SWI-Prolog é utilizada para formatar as respostas no formato JSON, possibilitando que o *backend* envie dados estruturados para o *frontend* em um formato amplamente compatível com diversos frameworks web, incluindo o React. Essa abordagem facilita a troca de informações entre o servidor e a interface do usuário, garantindo a interoperabilidade e a eficiência na comunicação entre as camadas da aplicação.

4.2.3.4 HTTP e CORS

CORS (Cross-Origin Resource Sharing) é um mecanismo que permite que requisições do lado do cliente acessem recursos de origens diferentes. Ele define algoritmos que possibilitam que uma API faça requisições a recursos de outra origem, controlando o acesso por meio de cabeçalhos de resposta, como o *Access-Control-Allow-Origin* (KESTEREN, 2010).

HTTP (Hypertext Transfer Protocol) é um protocolo de aplicação para sistemas distribuídos e colaborativos de informações hipermídia. É um protocolo genérico e sem estado, utilizado em diversas tarefas além do hipertexto, como servidores de nomes e sistemas de gerenciamento de objetos distribuídos. O HTTP permite a negociação e tipificação da representação dos dados, possibilitando a construção de sistemas independentes dos dados transferidos (FIELDING, 1999).

As bibliotecas http/thread_httpd, http/http_dispatch e http/http_cors, do SWI-Prolog, configuram um servidor HTTP básico, sendo responsáveis pelo tratamento de requisições e respostas. Elas também gerenciam as permissões de CORS, permitindo que o *frontend* (React) acesse o *backend* Prolog, mesmo quando este está hospedado em uma origem diferente.

4.2.3.5 Sessões HTTP

Uma sessão HTTP é mantida por meio do uso de cookies, que são pequenos arquivos de dados armazenados nos agentes de usuário (como navegadores) pelos servidores HTTP. Os cabeçalhos Cookie e Set-Cookie permitem que os servidores mantenham o estado de uma sessão, mesmo em um protocolo essencialmente sem estado como o HTTP. Isso significa que, através das sessões, os servidores podem armazenar informações sobre o usuário, como preferências ou dados de autenticação, e usá-las em requisições subsequentes, garantindo uma experiência contínua. Embora os cookies tenham questões históricas relacionadas à segurança e privacidade, os cabeçalhos Cookie e Set-Cookie são amplamente utilizados para gerenciar sessões de usuário, permitindo, por exemplo, o login persistente em *sites* ou a manutenção de um carrinho de compras em uma loja online (BARTH, 2011).

A biblioteca http/http_session do SWI-Prolog possibilita o gerenciamento de sessões de usuário, permitindo o armazenamento de variáveis de sessão, como questionNumber e answers. Esse recurso viabiliza a persistência de informações entre requisições subsequentes de um mesmo usuário, assegurando a continuidade do estado da aplicação ao longo da interação.

4.2.3.6 React e Typescript (Frontend)

O *frontend* foi feito em React[1] e Typescript[2], interagindo com o *backend* por meio de requisições HTTP, acessando *endpoints* para enviar as respostas do usuário e buscar o resultado final da questão.

React é uma biblioteca para a construção de interfaces de usuário, tanto para a web quanto para aplicativos nativos. Ela permite criar interfaces a partir de unidades individuais chamadas componentes, que podem ser reutilizados e combinados para formar interfaces complexas e interativas. Cada componente em React gerencia seu próprio estado e pode ser renderizado de forma eficiente conforme as mudanças nos dados, facilitando o desenvolvimento de aplicações dinâmicas e de fácil manutenção. React pode ser utilizado com outras linguagens que compilam para JavaScript[3], como TypeScript.

TypeScript é uma linguagem de programação que é um superconjunto do JavaScript, adicionando tipagem estática. Isso permite que os desenvolvedores definam tipos para variáveis e funções, ajudando a identificar erros antes da execução do código. TypeScript melhora a manutenção e segurança do código, especialmente em projetos grandes, e é amplamente utilizado com frameworks como React para criar aplicações mais escaláveis e robustas.

4.2.4 Arquitetura de Software

A aplicação propõe uma interação dinâmica em que o usuário, ao informar o número da questão sobre a qual deseja obter orientações, inicia um diálogo com o sistema de *chat*. O *chat* responde com perguntas específicas para compreender melhor as necessidades do usuário, permitindo direcionar as orientações de forma mais precisa. Conforme o usuário responde com "sim"ou "não"a cada pergunta, o sistema adapta as próximas questões e sugestões, guiando-o progressivamente até fornecer uma orientação final ajustada ao contexto das respostas acumuladas.

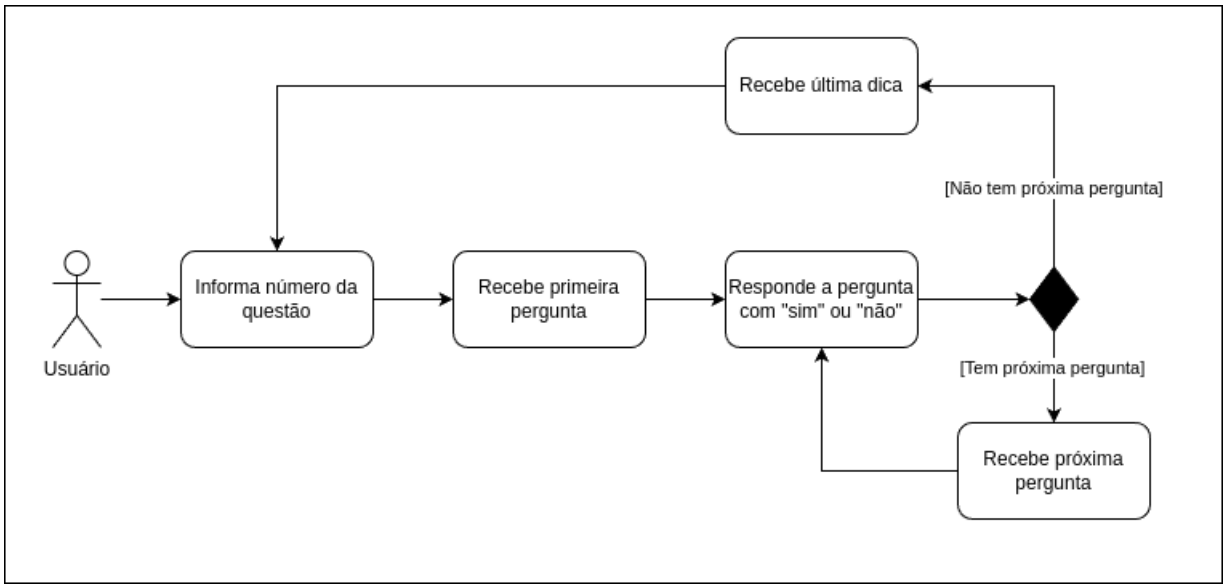
A Figura 27 apresenta o diagrama de atividades que detalha o fluxo de interação entre o usuário e o *chat* da aplicação. Após informar o número da questão sobre a qual deseja obter orientações, o usuário recebe uma pergunta inicial, respondendo com "sim"ou "não". Se houver mais perguntas, o *chat* continua a apresentá-las até que todas sejam respondidas, conduzindo o usuário a uma orientação final baseada nas respostas acumuladas. Ao final, o usuário pode iniciar o processo novamente com outro número de questão ou com o mesmo, caso deseje explorar outras sugestões e perguntas relacionadas.

[1] `https://react.dev/`

[2] `https://www.typescriptlang.org/`

[3] `https://www.javascript.com/`

Figura 27 – Diagrama de Atividades do Usuário da Aplicação



Fonte: Produzido pela autora.

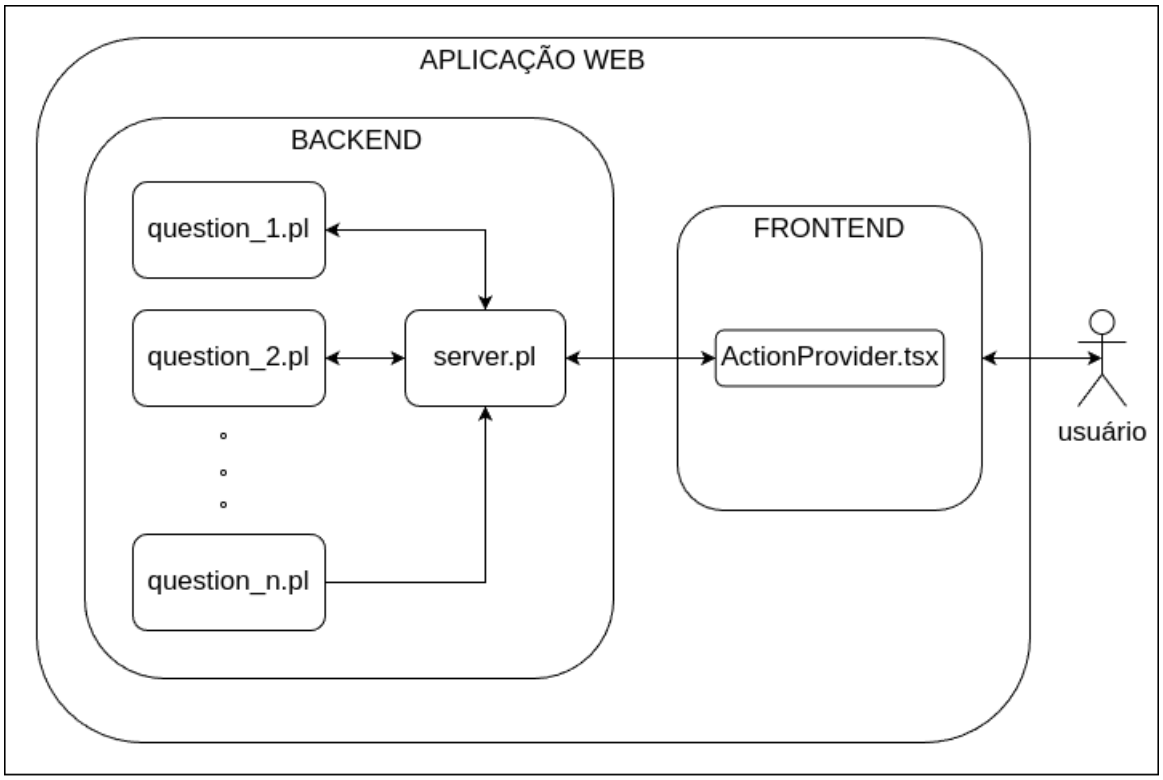
A Figura 28 representa a estrutura da aplicação, mostrando a interação entre o *frontend* e o *backend* através de requisições. No *frontend*, o arquivo *ActionProvider.tsx* gerencia as mensagens enviadas pelo usuário no *chat*, processando tanto o número da questão informada, quanto as respostas de "sim"ou "não". Por meio dessas requisições, o *frontend* envia esses dados ao *backend*, que, em resposta, fornece perguntas e dicas relacionadas à questão solicitada. Caso o usuário peça uma nova dica e o *backend* não tenha mais sugestões para aquela questão, ele envia a orientação final da questão, que é então apresentada ao usuário.

4.2.5 Elementos do Frontend

O *frontend* da aplicação foi desenvolvido utilizando as tecnologias React e TypeScript, conforme especificado no requisito não funcional **RNF6**, descrito na seção 4.2.2. Essas tecnologias foram escolhidas por sua capacidade de criar interfaces dinâmicas, escaláveis e de fácil manutenção, proporcionando uma experiência de usuário fluida e interativa **(RNF2)**.

Conforme o requisito funcional **RF1** e a regra de negócio **RN1**, foi implementado um sistema de *chat* como interface principal de interação com o usuário. Por meio deste *chat*, o usuário pode informar o código da questão do Beecrowd que está tentando resolver **(RF2)**. Além disso, conforme a regra de negócio **RN2**, o sistema especialista se comunica com o usuário por meio do *chat*, exibindo mensagens sequenciais baseadas nas respostas do sistema **(RF3)** em tempo real **(RNF1)**, e dicas finais **(RF4)**. O *chat* também permite que o usuário forneça respostas binárias (sim/não), conforme o fluxo de perguntas definido para personalizar a orientação fornecida, conforme as regras de

Figura 28 – Arquitetura da Aplicação do Sistema Especialista



Fonte: Produzido pela autora.

negócios **RN2** e **RN3**.

O desenvolvimento do *frontend* foi baseado em bibliotecas e repositórios existentes que facilitaram a criação e configuração do sistema de *chatbot*. Especificamente, foi utilizado o repositório *react-chatbot-kit* de FredrikOseberg[4] como estrutura principal para a criação do *chatbot*, combinado com o repositório *Chatbot-using-React-Chatbot-Kit* de Subid Das[5], que forneceu recursos adicionais para personalização da interface e integração do *chatbot* com o fluxo de mensagens interativo. Esses repositórios foram adaptados para atender às necessidades específicas da aplicação, garantindo que o *frontend* fosse capaz de processar interações dinâmicas e personalizadas, conforme os requisitos estabelecidos.

Pelo Código 1 é possível observar que, para a construção do compontente `Chat`, foi usado o componente `Chatbot` da biblioteca `react-chatbot-kit`. O `Chatbot` recebe o `MessageProvider`, responsável por processar as respostas do usuário e acionar as funções do `ActionProvider`, que gerencia o envio das requisições ao backend.

Código 1 – Componente Chat

```
import { useState } from "react";
```

[4] https://github.com/FredrikOseberg/react-chatbot-kit

[5] https://github.com/devsubid/Chatbot-using-React-Chatbot-Kit

```
import Chatbot from "react-chatbot-kit";
import "react-chatbot-kit/build/main.css";
import { styled } from "styled-components";
import config from "../../bot/config";
import ActionProvider from "../../bot/ActionProvider";
import MessageParser from "../../bot/MessageParser";


const Chat = () => {
    const [isOpened, setIsOpened] = useState<boolean>(false)
    return (
    <StyledChat>
        <div className="intro">
        <h2>Aqui você pode tirar dúvidas sobre algumas questões do Beecrowd!</h2>
        <button onClick={() => setIsOpened((prev) => !prev)}>
            Comece agora!
        </button>
        </div>
        {isOpened && (
        <Chatbot
            config={config}
            messageParser={MessageParser}
            actionProvider={ActionProvider}
        />
        )}
    </StyledChat>
    );
};

export default Chat;
```

Para atender ao Requisito Funcional **RF5**, descrito na seção 4.2.2, foi criado um componente `Details` para a rota details, a fim de mostrar as descrições dos possíveis resultados do juíz online do Beecrowd (Código 2).

Código 2 – Rotas do frontend

```
<Routes location={location} key={location.pathname}>
    <Route path="/" element={<Chat />} />
    <Route path="/details" element={<Details />} />
    <Route path="*" element={<NotFound />} />
</Routes>
```

4.2.6 Comunicação Frontend-Backend

O Código 3 representa como o arquivo *ActionProvider.tsx*, do *frontend*, informa para o *backend* o número da questão. Ele define duas funções assíncronas: `fetchData` e `handleFirstMessage`, usadas para interagir com um *backend* via requisições HTTP.

A função `fetchData` recebe uma URL e um corpo de dados (`URLSearchParams`) para fazer uma requisição `POST`. Ela define cabeçalhos para enviar os dados como `application/x-www-form-urlencoded`, garantindo que o *backend* receba os dados em um formato compatível e de fácil manipulação, especialmente em linguagens que interpretam parâmetros de URL de forma nativa. Além disso, a função inclui cookies nas requisições com `credentials: 'include'` para permitir que a sessão seja mantida entre as requisições, o que é crucial para a continuidade da comunicação com o *backend* e possibilita cumprir o Requisito Não Funcional **RNF4** descrito na seção 4.2.2. Caso a resposta não tenha o status 200, a função trata o erro, retornando uma mensagem de erro em vez do JSON da resposta.

A função `handleFirstMessage` constrói um corpo de dados contendo `questionNumber` e chama `fetchData` para enviar esses dados ao *backend* no *endpoint* `"endereço/server"`. Em seguida, cria uma estrutura de mensagens adicionais vazia e invoca `handleMessageResponse` para manipular a resposta, preenchendo essa estrutura conforme a resposta recebida.

Código 3 – Funções assíncronas para requisição HTTP

```
    const fetchData = async (url: string, body: URLSearchParams) => {
        try {
            const response = await fetch(url, {
            method: "POST",
            headers: { 'Content-Type':
    'application/x-www-form-urlencoded;charset=UTF-8' },
            credentials: 'include',
            body,
            });

            if (response.status !== 200) throw new Error("Erro ao consultar o
    backend.");

            return await response.json();
        } catch (error) {
            return { error: "Erro ao consultar o backend." };
        }
    };

const handleFirstMessage = async (questionNumber?: string) => {
```

```
    const body = new URLSearchParams({ questionNumber: questionNumber?.toString()
    || '' });
    const data = await fetchData("{endereço}/server", body);
    let additionalMessages: {
        messages: {
        message: string;
        type: string;
        id: number;
        loading?: boolean;
        widget?: string | undefined;
        delay?: number | undefined;
        payload?: any;
        }[];
    }[] = [];

    await handleMessageResponse(data, additionalMessages);
};
```

4.2.7 Elementos do Backend

O *backend* foi desenvolvido em Prolog, utilizando o SWI-Prolog para possibilitar o tratamento de requisições HTTP, conforme especificado no Requisito Não Funcional **RNF6**, na seção 4.2.2.

Além disso, conforme ilustrado na Figura 28, o *backend* é composto pelo arquivo principal `server.pl` e por arquivos adicionais, cada um representando um sistema especialista específico para cada questão, seguindo, também, o Requisito Não Funcional **RNF6**. Esses arquivos são nomeados no formato `questao_{numeroDaQuestao}.pl` e contêm dicas específicas para as respectivas questões. Por exemplo, o arquivo `questao_1234.pl` armazena as orientações para a questão 1234 do Beecrowd.

4.2.7.1 Arquivos no formato questao_{numeroDaQuestao}.pl

O Código 4 apresenta o arquivo *questao_1181.pl* como um exemplo de estrutura para arquivos do tipo *questao_{numeroDaQuestao}.pl*.

Código 4 – Arquivo *questao_1181.pl*

```
:- module(questao_1181, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1181, [], "Você já desenvolveu sua solução, mas não está saíndo a
resposta correta?").
questao(1181, ["sim"], "Tome cuidado que a questão quer todos os elementos da
```

```
linha, e não da coluna! Verifique \ccomo você está iterando sobre a matriz: se
está pegando todos os elementos de uma linha \c ou todos os elementos de uma
coluna. Mais uma dica sobre isso?").
questao(1181, ["sim", "sim"], "Se sua matriz é feita de forma que é iterada
assim: matriz[linha][coluna], verifique \cse o 'for' está correto. Quer saber
como o for fica?").
questao(1181, ["sim", "sim", "sim"], "considerando que l é a entrada do
usuário que indica a linha que será a \cconsiderada para operação: \nfor i in
range(0, 12): soma += matriz[l][i]. \n\cPróxima dica?").
questao(1181, ["sim", "sim", "sim", "sim"], "Verifique se o número pelo qual
você está tentando dividir para \cpara conseguir a média está correto. Próxima
pergunta?").

questao(1181, Respostas, "Deu Presentation Error?") :-
    Respostas = ["não"];
    Respostas = ["sim", "não"];
    Respostas = ["sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim", "sim"].

questao(1181, Respostas, "Perceba que a formatação que a questão pede é:
Imprima o resultado solicitado \c(a soma ou média), com 1 casa após o ponto
decimal. Ou seja, '%.1f'. Deu certo?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "não", "sim"];
    Respostas = ["sim", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim"].

% Diagnóstico com base nas respostas completas
diagnostico(1181, Respostas, "Legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "não", "sim", "sim"].

diagnostico(1181, _, "Atente-se bem às dicas já enviadas ou pergunte
novamente! Além disso, você pode \cusar o udebug para verificar a diferença
entre as saídas do seu código, e as \csaídas esperadas pela questão! Também é
uma boa ideia debugar seu código no Thonny, \crevisando linha por linha, ou
usando prints para entender o que ele está executando!").
```

Esse arquivo Prolog define um módulo especializado para auxiliar usuários na resolução da questão 1181 por meio de um sistema especialista baseado em regras. O módulo *questao_1181* implementa dois predicados principais: *questao/3* e *diagnostico/3*, que são responsáveis por operacionalizar a lógica de suporte do sistema.

- **Predicado *questao/3*:** Este predicado orienta o usuário de maneira sequencial, adaptando as instruções com base nas respostas fornecidas. Ele recebe três parâmetros: o identificador da questão, uma lista de respostas (que mantém o histórico de interações), e uma mensagem de orientação. A matriz de decisões, implementada neste predicado, direciona o fluxo de perguntas e respostas com base no histórico de interações do usuário, promovendo uma progressão lógica que facilita a resolução da questão. Essa matriz está estruturada de forma que cada combinação de respostas leva a uma nova instrução, que oferece dicas cada vez mais específicas até que o usuário obtenha a solução correta.

  A tabela 2 apresenta a matriz de decisões para o predicado *questao/3*, em formato de tabela, representando as combinações de respostas possíveis e as respectivas instruções emitidas:

Tabela 2 – Matriz de Decisões para o Predicado *questao/3* na Questão 1181

Questão	**Respostas**	**Instrução**
1181	[]	Você já desenvolveu sua solução, mas não está saindo a resposta correta?
1181	[sim]	Tome cuidado que a questão quer todos os elementos da linha, e não da coluna! Verifique como você está iterando sobre a matriz: se está pegando todos os elementos de uma linha ou todos os elementos de uma coluna. Mais uma dica sobre isso?
1181	[sim, sim]	Se sua matriz é feita de forma que é iterada assim: matriz[linha][coluna], verifique se o 'for' está correto. Quer saber como o for fica?
1181	[sim, sim, sim]	Considerando que l é a entrada do usuário que indica a linha que será considerada para operação: `for i in range(0, 12): soma += matriz[l][i]` Próxima dica?

Continuação da Tabela		
Questão	**Respostas**	**Instrução**
1181	[sim, sim, sim, sim]	Verifique se o número pelo qual você está tentando dividir para calcular a média está correto. Próxima pergunta?
1181	[não], [sim, não], [sim, sim, não], [sim, sim, sim, não], [sim, sim, sim, sim, sim]	Deu Presentation Error?
1181	[não, sim], [sim, não, sim], [sim, sim, não, sim], [sim, sim, sim, não, sim], [sim, sim, sim, sim, sim, sim]	Perceba que a formatação que a questão pede é: Imprima o resultado solicitado (a soma ou média), com 1 casa após o ponto decimal. Ou seja, %.1f. Deu certo?

- **Predicado *diagnostico/3*:** Este predicado fornece um diagnóstico final com base nas respostas acumuladas ao longo da interação. Atuando como *feedback* conclusivo, ele emite uma mensagem de sucesso caso as respostas indiquem que o usuário compreendeu e solucionou a questão. Caso contrário, sugere ferramentas como o *udebug* e o depurador *Thonny*, de modo que o usuário possa inspecionar o comportamento do seu código em detalhes.

Dessa forma, arquivos do formato *questao_{numeroDaQuestao}.pl* devem seguir essa estrutura, onde cada questão contém um predicado *questao/3* para fornecer orientações sequenciais e um predicado *diagnostico/3* para avaliação final. A matriz de decisões implementada no predicado *questao/3* permite um processo dinâmico e adaptativo de interação, guiando o usuário de forma direcionada e progressiva na resolução da questão específica.

Assim, para a inclusão de novas questões, basta criar um arquivo no formato especificado, seguindo a mesma estrutura. Essa abordagem facilita a expansão do sistema, conforme estabelecido no requisito não funcional **RNF5**, descrito na seção 4.2.2.

4.2.7.2 Arquivo server.pl

4.2.7.2.1 Configurando servidor HTTP

O arquivo *server.pl* configura e inicializa um servidor HTTP, definindo rotas para processar requisições e habilitando o CORS, permitindo conexões da porta do *frontend*. Ele recebe requisições *POST* para iniciar ou continuar uma questão, associando cada sessão ao usuário correspondente, e também aceita requisições *GET* para fornecer o diagnóstico final com base nas respostas acumuladas durante a interação.

O Código 5 exemplifica a configuração de um servidor HTTP em Prolog utilizando o módulo `http_dispatch`, no qual as rotas são definidas por meio de `http_handler` para manipular requisições *GET*, *POST* e *OPTIONS*. Especificamente, a requisição *POST* é tratada pela rota `server`, sendo manipulada pela função `handle_post_request`.

Código 5 – Arquivo *server.pl* - *Handlers* e Manipuladores de requisições

```
% Define o handler para receber as requisições HTTP POST
:- http_handler(root(server), handle_post_request, [method(post)]).
:- http_handler(root(.), handle_options, [method(options)]).

% Manipulador de requisições POST
handle_post_request(Request) :-
    member(method(post), Request), !,

    % Garante que a sessão está ativa para o cliente
    http_session_id(_SessionID),

    % Lê os dados da requisição
    http_read_data(Request, Data, [encoding(utf8)]),

    % Definindo os headers para permitir CORS
    format('Access-Control-Allow-Origin: {endereço}~n'),
    format('Access-Control-Allow-Credentials: true~n'),
    format('Content-Type: application/json; charset=UTF-8'),
    cors_enable(Request, [methods([post])]),

    (   % Caso seja a primeira requisição da questão
        member(questionNumber=QN, Data)
    ->  % Inicializa a sessão e a questão
        http_session_retractall(questionNumber(_)),
        http_session_retractall(answers("Answers", _)),
        http_session_assert(answers("Answers", [])),

        http_session_assert(questionNumber(QN)),
        start_question(QN, FirstQuestion),

        reply_json_dict(FirstQuestion)

    ;   % Caso receba respostas subsequentes
        member(answer=Answer, Data),

        % Garante que a sessão tenha a variável questionNumber
```

```
        ( http_session_data(questionNumber(QN)) ->
            continue_question(QN, Answer, NextQuestionOrResult),
            reply_json_dict(NextQuestionOrResult)
        ;   % Erro caso a sessão não tenha um questionNumber
            reply_json_dict(_{error: "Número da questão não encontrado na sessão"})
        )
    ).
```

O **manipulador de requisições POST** (`handle_post_request`) tem como função processar requisições *POST* enviadas ao servidor, gerenciando o fluxo de perguntas e respostas entre o cliente e o servidor. Inicialmente, ele habilita o CORS, seguindo o Requisito Não Funcional **RNF6**, descrito na seção 4.2.2. Isso permite requisições provenientes do *frontend* no endereço `endereço`, e configura os cabeçalhos necessários para o envio de credenciais, como cookies de sessão. A função verifica se a sessão está ativa, garantindo a associação com um ID de sessão único, cumprindo os Requisitos Não Funcionais **RNF3** e **RNF4**.

Dependendo do tipo de requisição, a função processa a primeira pergunta ou continua a sequência de perguntas baseadas nas respostas anteriores. Se for a primeira requisição, o número da questão é extraído, e a sessão é inicializada com os dados necessários. Além disso, o arquivo correspondente à questão é carregado dinamicamente.

Caso seja uma resposta subsequente, a função valida a presença do número da questão na sessão, atualiza as respostas e retorna a próxima pergunta ou o resultado final. Se os dados da sessão estiverem faltando, um erro é enviado indicando que o número da questão não foi encontrado.

4.2.7.2.2 Carregando os arquivos das questões

Os arquivos de questões são carregados dinamicamente no *server.pl* com base no número da questão recebido nas requisições *POST*. Se o usuário alterna para uma nova questão, por exemplo, a questão Y, o *backend* carrega o arquivo *questao_Y.pl* e passa a conduzir a interação de acordo com suas informações.

No Código 6, é mostrado o trecho do `server.pl` que realiza esse carregamento dinâmico. O predicado `start_question/2` tem como objetivo inicializar uma nova questão dentro de um sistema interativo, baseado no número da questão fornecido. O primeiro passo do código é construir o caminho para o arquivo que contém a definição da questão, concatenando o prefixo `questao_` com o número da questão (`QuestionNumber`) e adicionando a extensão `.pl` para formar o nome completo do arquivo. Isso é realizado através do predicado `atom_concat3`, que concatena átomos (strings) para formar o caminho completo do arquivo a ser carregado.

Código 6 – Arquivo *server.pl* - *Handlers* e Manipuladores de requisições

```
start_question(QuestionNumber, FirstQuestion) :-
    atom_concat('questao_', QuestionNumber, FileName),
    atom_concat(FileName, '.pl', FilePath),

    % Verifica se o arquivo existe antes de tentar carregá-lo
    (   exists_file(FilePath)
    ->  % Carrega dinamicamente o arquivo da questão
        abolish(questao/3),
        abolish(diagnostico/3),

        ensure_loaded(FilePath)
    ).
```

Depois, o código verifica se o arquivo gerado existe no sistema de arquivos utilizando o predicado `exists_file1`. Caso o arquivo exista, o sistema prossegue para carregá-lo dinamicamente com o predicado `ensure_loaded1`. Antes de carregar o arquivo, o predicado `abolish1` é utilizado para garantir que qualquer definição anterior dos predicados `questao3` e `diagnostico3` seja removida da memória, evitando conflitos ou sobrecarga de definições antigas.

4.2.7.2.3 Recebe respostas

O arquivo *server.pl* também armazena na sessão do usuário as informações fornecidas, como número da questão e respostas, para garantir a continuidade da interação. O Código 7 apresenta a função **continue_question**, responsável por processar respostas subsequentes enviadas pelo usuário para uma questão em andamento, armazenando a resposta na sessão e determinando o próximo passo com base nas respostas acumuladas. Inicialmente, a função adiciona a nova resposta à lista de respostas da sessão, atualizando a variável de sessão `answers("Answers")`.

Código 7 – Arquivo *server.pl* - *Handlers* e Manipuladores de requisições

```
continue_question(QuestionNumber, Answer, Response) :-
    (   string(Answer) -> AW = Answer
    ;   atom(Answer) -> atom_string(Answer, AW)
    ;   term_string(Answer, AW)
    ),
    atom_number(QuestionNumber, QN),

    % Salva a resposta na sessão
    http_session_data(answers("Answers", AnswerList)),
    append(AnswerList, [AW], UpdatedAnswers),
```

```
    http_session_retractall(answers("Answers", _)),
    http_session_assert(answers("Answers", UpdatedAnswers)),

    % Chama a questão correspondente com a lista atualizada de respostas
    (   questao(QN, UpdatedAnswers, NextQuestion)
    ->  Response = _{question: NextQuestion}
    ;   (   diagnostico(QN, UpdatedAnswers, Result)
        ->  true
        ;   Result = "Sem conclusão final."
        ),
        Response = _{result: Result}
    ).
```

Em seguida, tenta gerar a próxima pergunta por meio da chamada ao predicado `questao/3`. Se `questao/3` retornar uma nova pergunta (`NextQuestion`), esta é enviada ao cliente para dar continuidade à interação. Caso contrário, a função verifica se uma resposta final pode ser gerada utilizando o predicado `diagnostico/3`. O resultado final, seja uma nova pergunta ou uma conclusão, é então enviado ao cliente no formato JSON, permitindo que a interação prossiga ou seja concluída de acordo com o progresso da sessão, como consta no Requisito Não Funcional **RNF6**, descrito na seção 4.2.2.

4.2.8 Expert Bee: Sistema Especialista de Apoio à Resolução de Dúvidas nos Exercícios do Beecrowd

A aplicação foi desenvolvida para ser acessada via navegador e tem como objetivo ajudar os usuários com dúvidas na resolução dos exercícios do Beecrowd. Por meio de um *chat* interativo, o usuário pode informar qual questão precisa de orientação. A partir disso, o sistema especialista realiza perguntas específicas para compreender melhor as necessidades do usuário e, com base nas respostas, fornece dicas e orientações personalizadas. Ou seja, à medida que o usuário responde "sim"ou "não"a cada pergunta, o sistema ajusta as questões subsequentes e as sugestões, guiando o usuário de forma progressiva até fornecer uma solução ajustada ao contexto das respostas acumuladas.

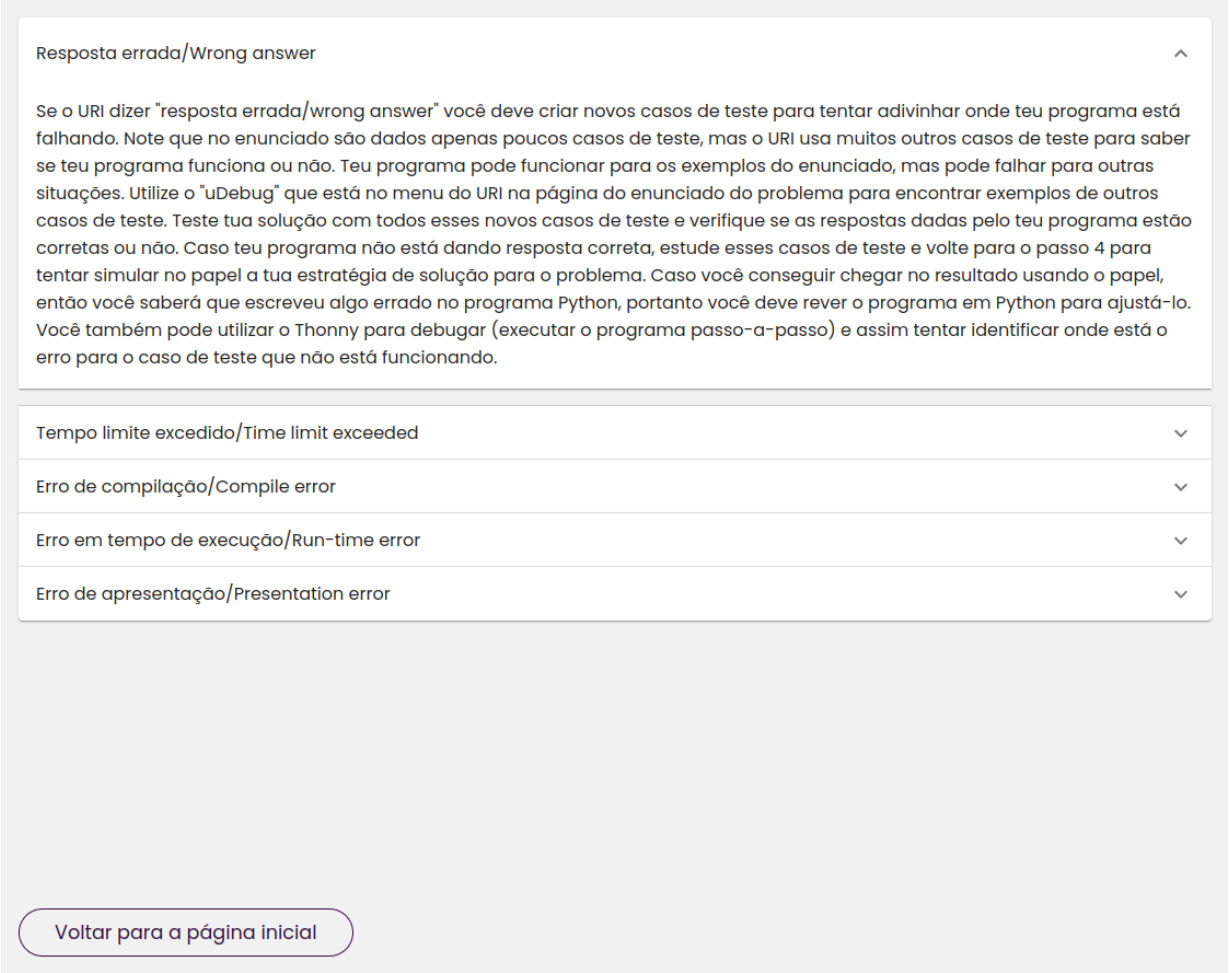
Nesta seção, serão apresentadas imagens que ilustram o resultado final da aplicação.

Figura 29 – Expert Bee: Tela Inicial

Fonte: Produzido pela autora.

A Figura 29 exibe a tela inicial da aplicação Web, que contém dois botões. Ao clicar no botão "Clique aqui para ler sobre os tipos de resposta do Beecrowd!", o usuário é redirecionado para a tela mostrada na Figura 30. Nessa tela, são apresentados detalhes sobre cada tipo de resposta do Beecrowd, atendendo ao Requisito Funcional **RF5**, descrito na seção 4.2.2. Além disso, caso o usuário clique no botão "Voltar para a página inicial", ele retorna à tela inicial.

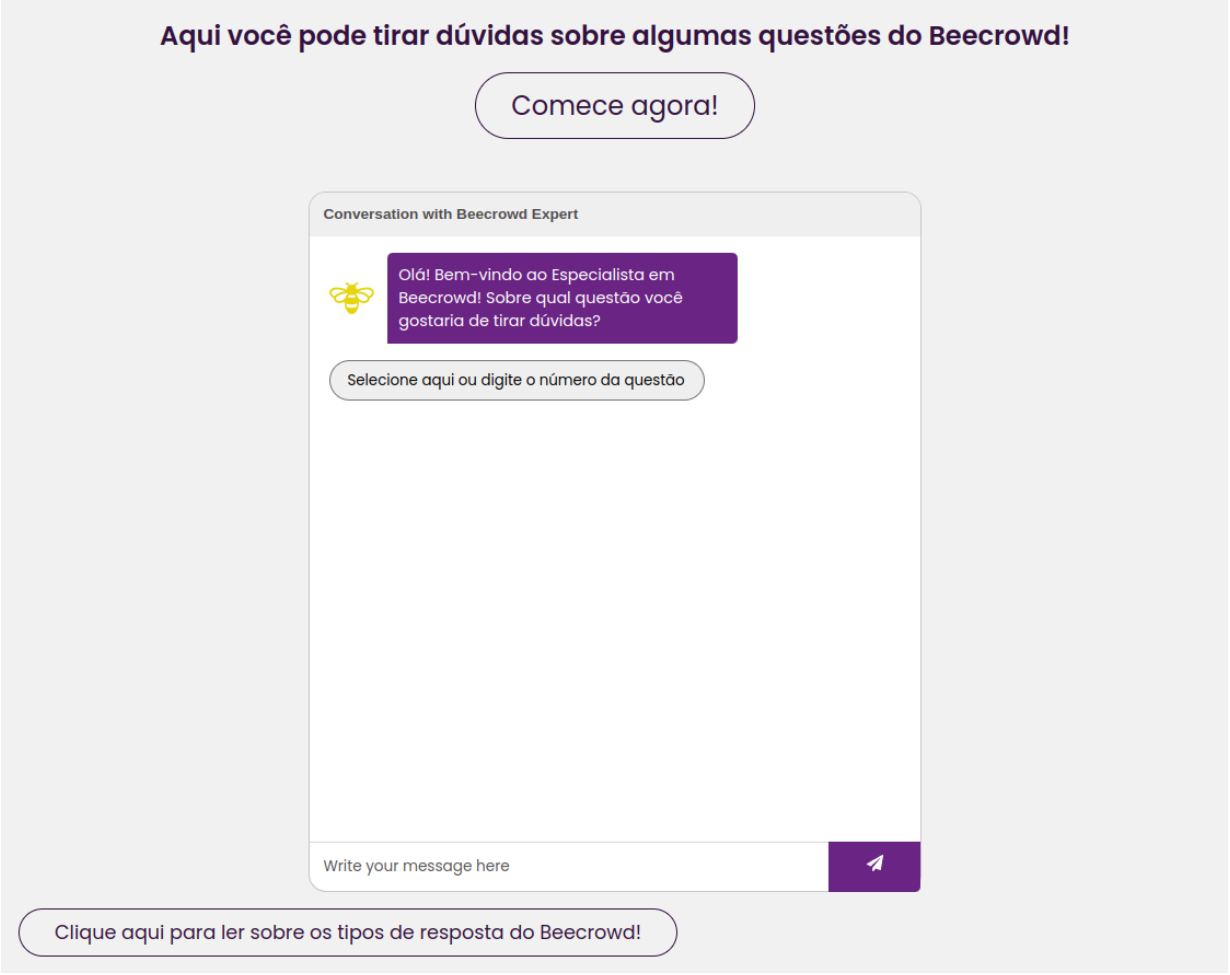
Figura 30 – Expert Bee: Detalhes das Respostas do Juíz Online do Beecrowd



Fonte: Produzido pela autora.

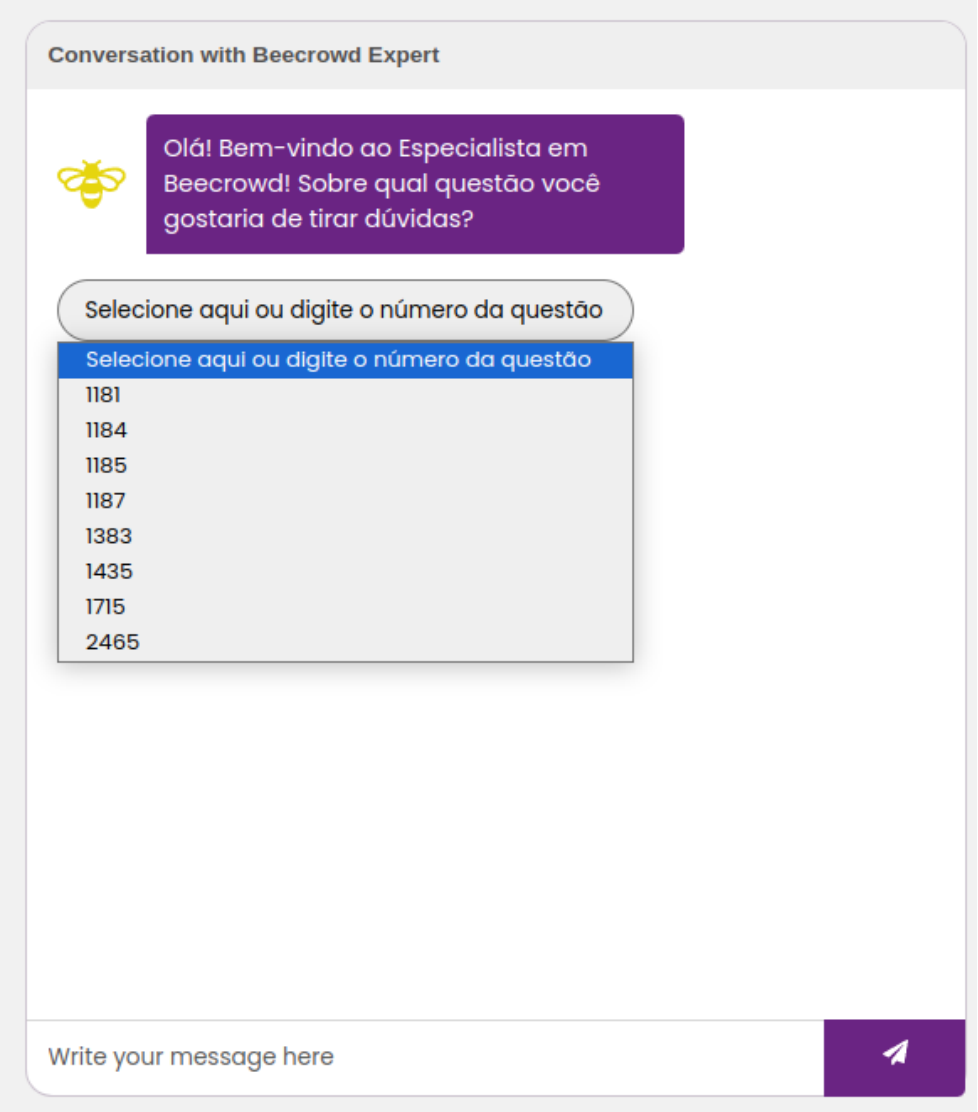
Na página inicial, ao clicar no botão "Comece agora!", é aberto o *chat*, conforme o Requisito Funcional **RF1**, ilustrado na Figura 31. Através deste *chat*, o usuário deve inicialmente informar o número da questão sobre a qual deseja tirar dúvidas, atendendo ao Requisito Funcional **RF2**. O usuário pode selecionar o número da questão a partir da lista apresentada (Figura 32) ou optar por digitá-lo manualmente (Figura 33). Caso o número informado não corresponda a uma questão existente no sistema, uma mensagem informando que a questão não foi encontrada será exibida (Figura 34).

Figura 31 – Expert Bee: Chat



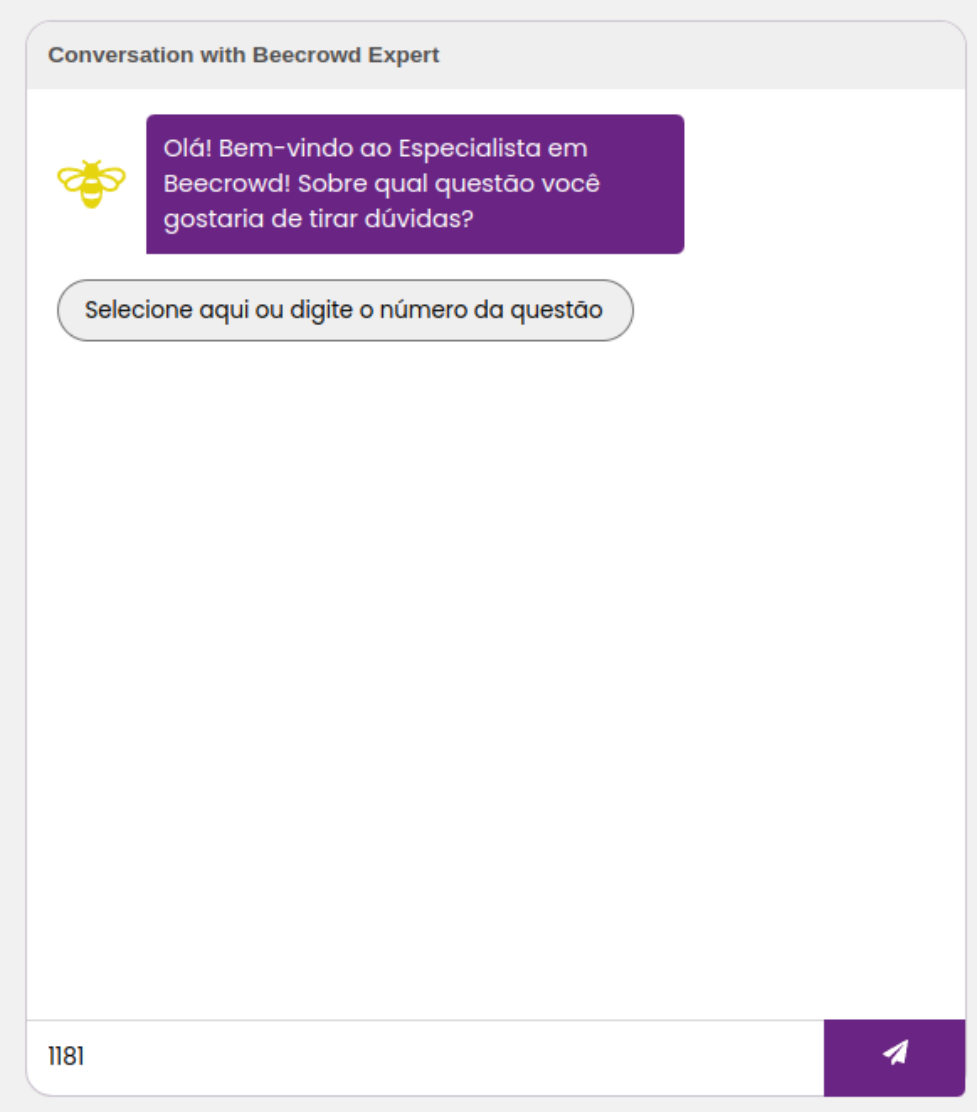
Fonte: Produzido pela autora.

Figura 32 – Expert Bee: Escolhendo a Questão



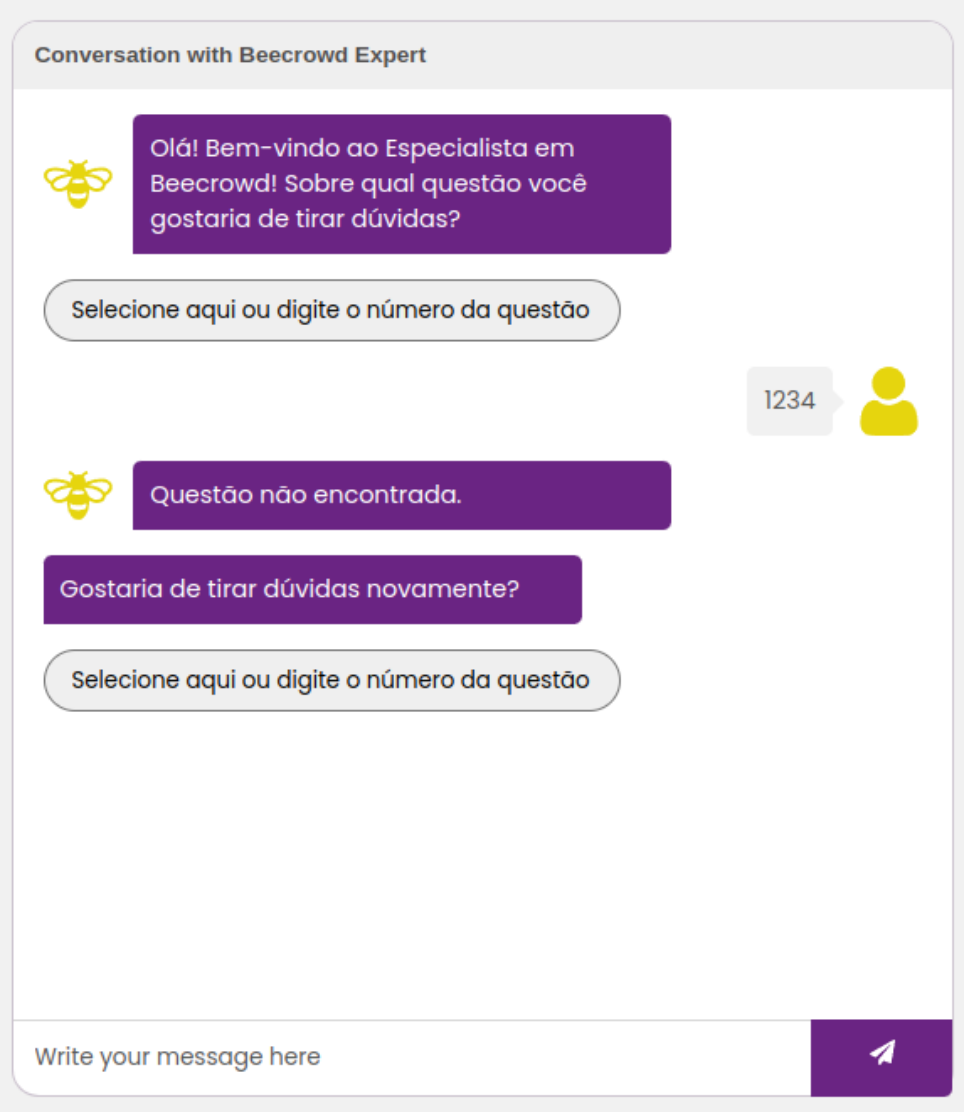
Fonte: Produzido pela autora.

Figura 33 – Expert Bee: Digitando o Número da Questão



Fonte: Produzido pela autora.

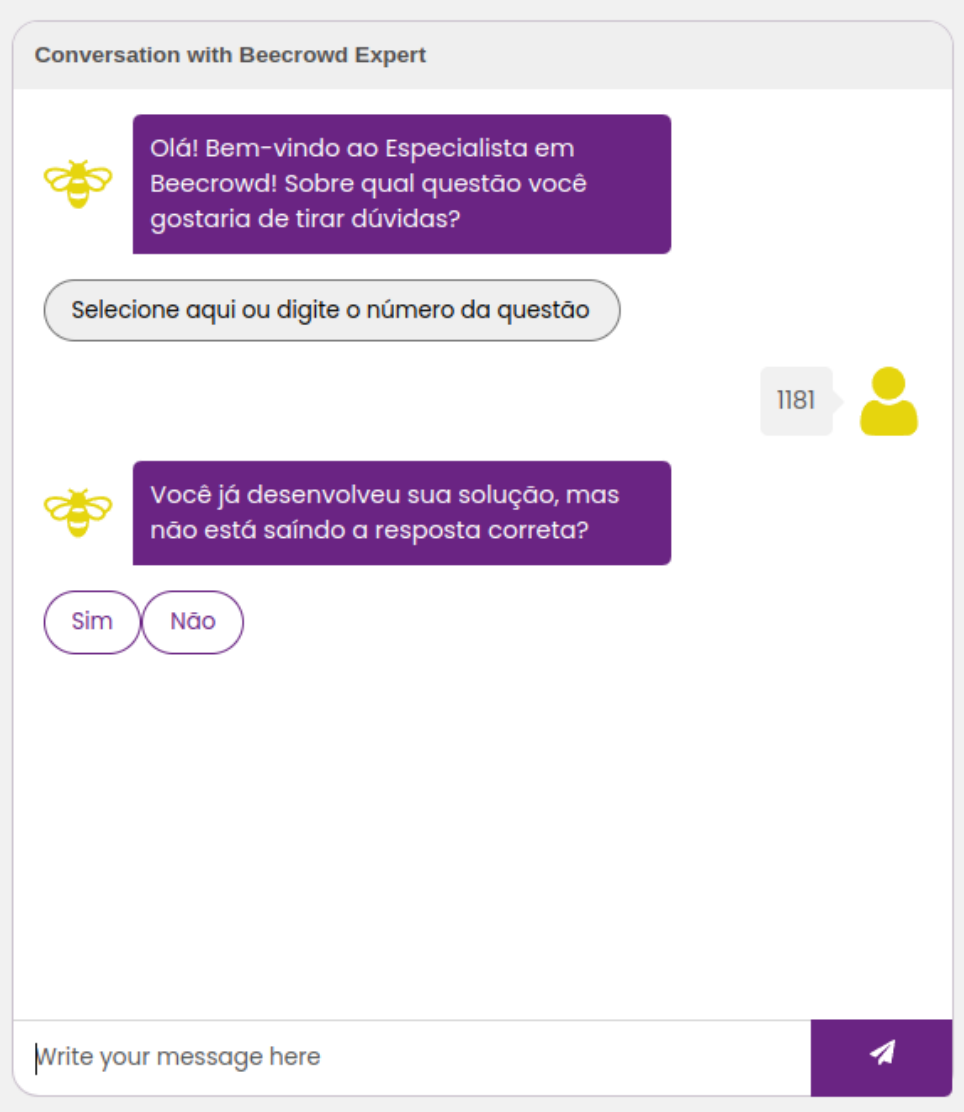
Figura 34 – Expert Bee: Questão não encontrada



Fonte: Produzido pela autora.

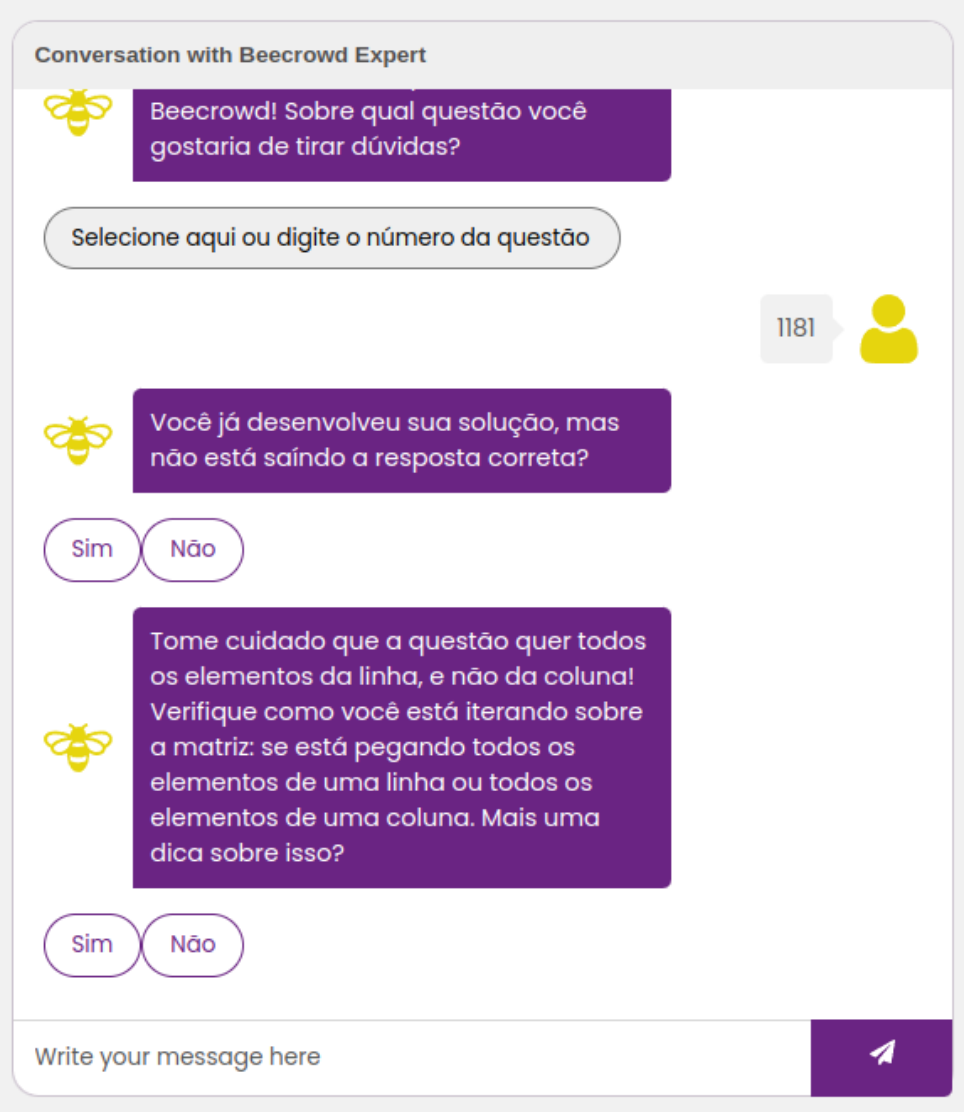
Se a questão for localizada, o *chat* começa a enviar as perguntas do sistema especialista (Figura 35), com opções de resposta "sim"ou "não", conforme o Requisito Funcional **RF3**. O usuário deve responder clicando no botão "sim"ou "não". Se a resposta for "sim", a resposta adequada para aquela pergunta é exibida junto com a próxima pergunta (Figura 36). Se a resposta for "não", uma nova pergunta será apresentada (Figura 37). Assim, o Requisito Funcional **RF4** é atendido, pois o sistema adapta as perguntas conforme as respostas fornecidas pelo usuário.

Figura 35 – Expert Bee: Primeira Pergunta



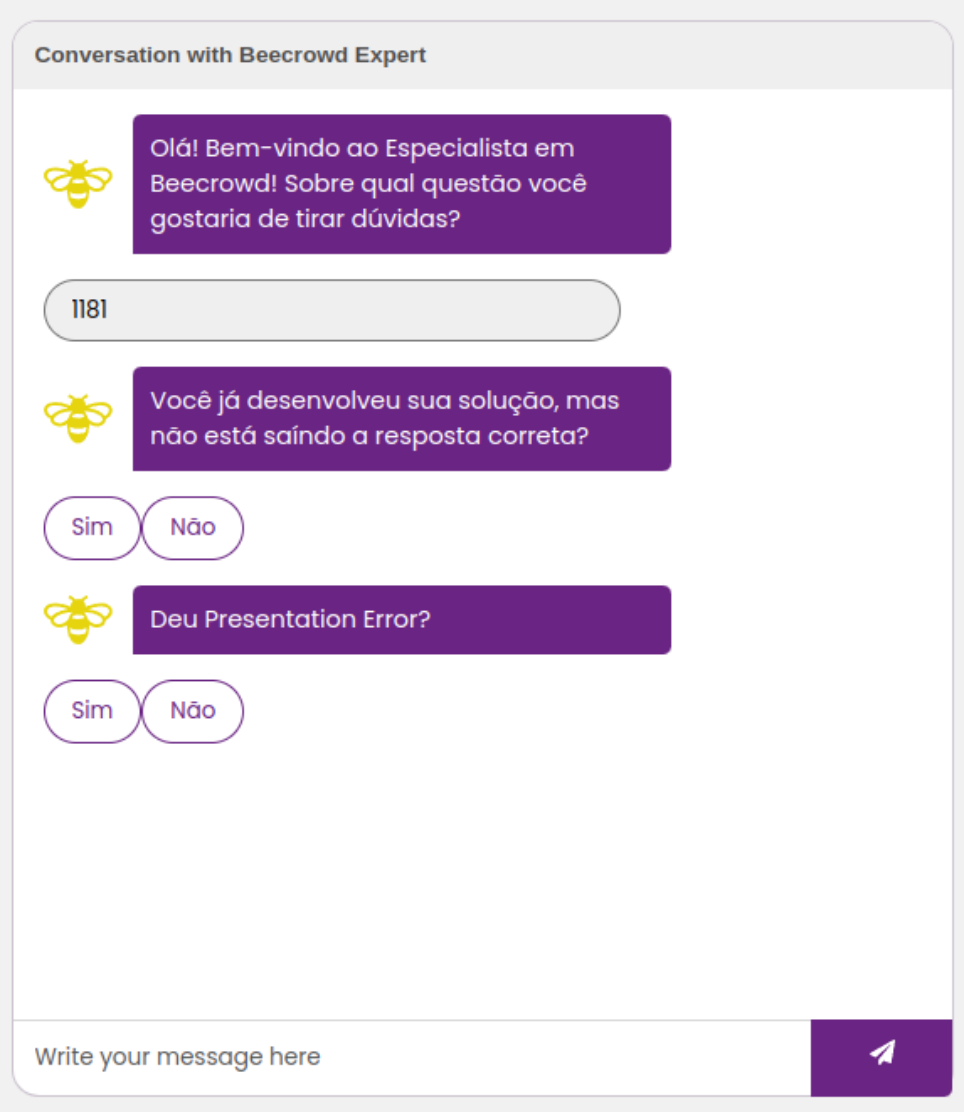
Fonte: Produzido pela autora.

Figura 36 – Expert Bee: Sim para a Primeira Pergunta



Fonte: Produzido pela autora.

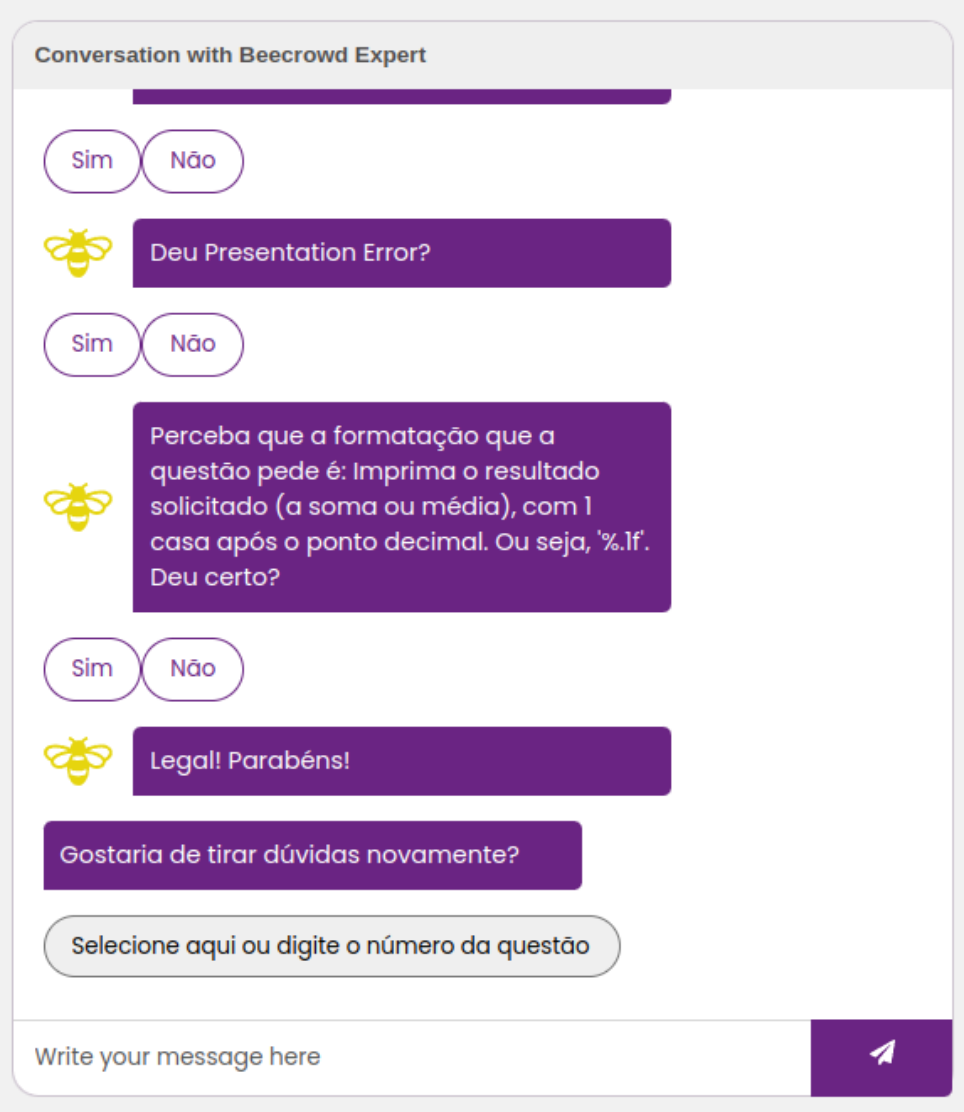
Figura 37 – Expert Bee: Não para a Primeira Pergunta



Fonte: Produzido pela autora.

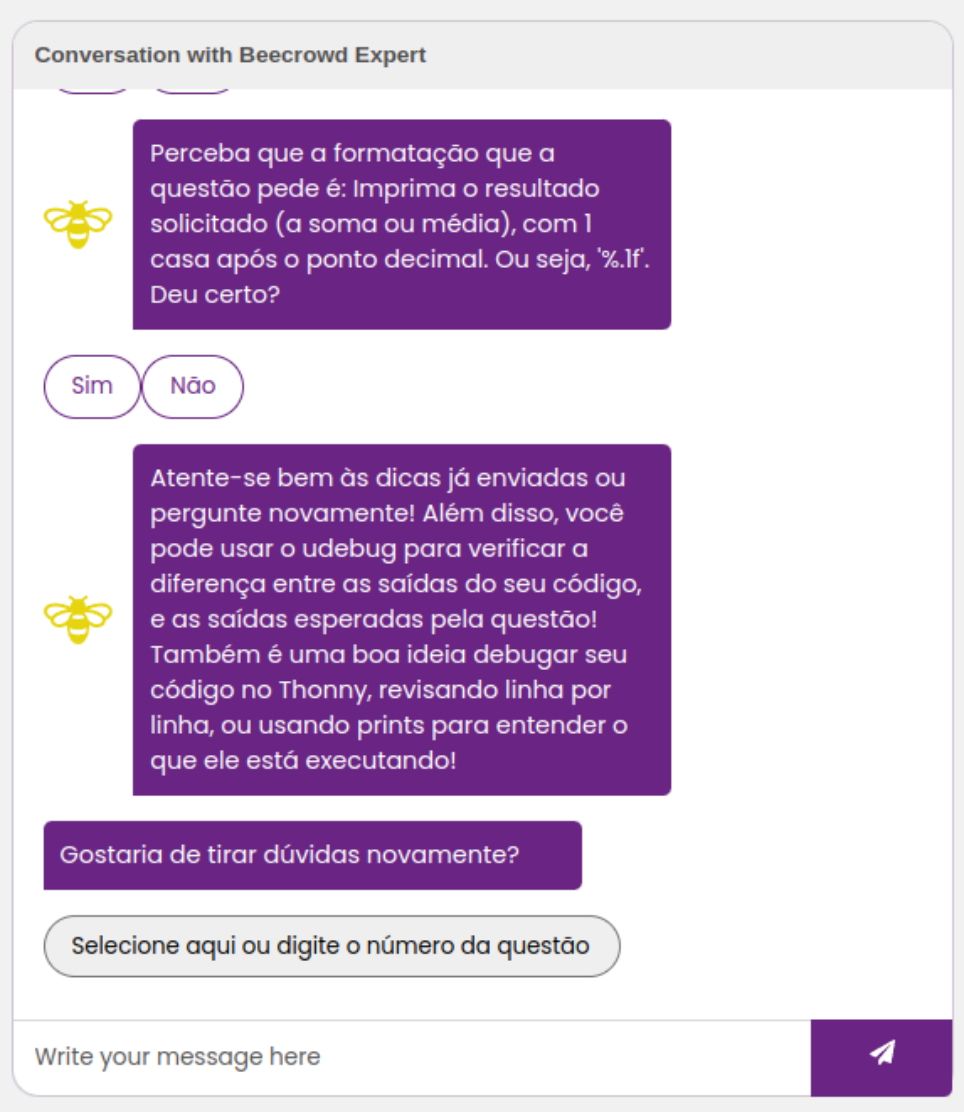
A Figura 38 mostra a resposta final, caso o usuário indique que conseguiu resolver a questão. Já a Figura 39 exibe a resposta com uma dica final, caso o usuário informe que não conseguiu resolver o problema. Em ambos os casos, o usuário tem a opção de inserir um novo número de questão, para iniciar uma nova sessão de dúvidas.

Figura 38 – Expert Bee: Usuário conseguiu



Fonte: Produzido pela autora.

Figura 39 – Expert Bee: Usuário não conseguiu



Fonte: Produzido pela autora.

4.2.9 Adicionar novas questões no Expert Bee

Para adicionar novas questões ao sistema, deve-se criar um arquivo Prolog no diretório `/backend`, nomeado conforme o formato `questao_numeroDaQuestao.pl`. Cada arquivo deve seguir uma estrutura padrão contendo dois predicados principais: `questao/3` e `diagnostico/3`.

O predicado `questao/3` associa uma sequência de respostas do usuário a uma nova pergunta, enquanto o predicado `diagnostico/3` fornece um diagnóstico ou *feedback* final com base nas respostas completas. O sistema utiliza essas definições para iterar sobre as perguntas e fornecer respostas progressivas ou um diagnóstico final, de acordo com as respostas fornecidas até o momento. A consistência das respostas e do número da questão é essencial para o funcionamento adequado do sistema.

Ao adicionar uma nova questão, é fundamental que o arquivo siga corretamente

essa estrutura, permitindo que o sistema integre automaticamente a questão ao fluxo de interações. As respostas parciais devem ser tratadas de forma coerente, e qualquer sequência de respostas não mapeada deve ser considerada com um curinga no predicado `diagnostico/3`.

Com essa estrutura, o sistema será capaz de processar novas questões de forma automática, bastando apenas adicionar o arquivo correspondente ao diretório. Essa abordagem atende e completa o Requisito Não Funcional **RNF5**, conforme descrito na seção 4.2.2.

Além disso, foi criado um manual (Apêndice C) que auxilia o professor, passo a passo, em como adicionar dicas para uma nova questão no Expert Bee. Este manual proporciona uma abordagem prática para garantir que as questões sejam adicionadas de forma correta e eficiente, facilitando a inclusão de novas questões e o fornecimento de orientações adequadas aos usuários do sistema.

5 VALIDAÇÃO

Nesta seção, será realizada uma análise crítica da proposta, abordando as vantagens e desvantagens da integração entre o Moodle e o Beecrowd via LTI (*Learning Tools Interoperability*), bem como da aplicação da aplicação com sistema especialista desenvolvida. Além disso, será apresentada uma avaliação prática sobre o uso da aplicação pelos alunos da disciplina *Programação Orientada a Objetos I* da Universidade Federal de Santa Catarina (UFSC) no segundo semestre de 2024, com o objetivo de verificar se o sistema especialista conseguiu resolver efetivamente as dúvidas dos estudantes.

5.1 INTEGRAÇÃO LTI ENTRE O MOODLE E O BEECROWD

5.1.1 Vantagens

- **Vantagens do Beecrowd:** O Beecrowd oferece uma plataforma robusta para a resolução de problemas de programação, com uma grande variedade de questões que permitem aos alunos praticar e desenvolver habilidades de programação em diversos níveis de dificuldade. A plataforma também facilita o acompanhamento do progresso dos alunos, oferecendo métricas e *feedback* instantâneo sobre o desempenho nas questões.
- **Facilidade de acesso através da integração LTI:** A integração LTI simplifica o acesso ao Beecrowd, pois tanto o professor quanto o aluno podem acessar a plataforma diretamente do Moodle com um único clique.

5.1.2 Desvantagens

- **Desenvolvimento externo e curva de aprendizado:** O desenvolvimento dos códigos e a criação das listas de exercícios ainda ocorre fora do Moodle, no Beecrowd, o que significa que o professor precisa aprender a usar o Beecrowd Academic, embora o processo seja relativamente simples. Isso pode exigir um tempo adicional de adaptação, especialmente para aqueles que não estão familiarizados com a plataforma.

5.2 SISTEMA ESPECIALISTA PARA RESOLUÇÃO DE DÚVIDAS NO BEECROWD

5.2.1 Vantagens

- **Atendimento automatizado:** O sistema especialista pode fornecer respostas rápidas e precisas para questões frequentes ou comuns, permitindo que os alunos resolvam suas dúvidas sem a necessidade de intervenção humana constante.

- **Apoio no aprendizado:** O sistema pode ser configurado para fornecer explicações detalhadas sobre questões específicas do Beecrowd, ajudando os alunos a entender melhor os problemas e as soluções.
- **Escalabilidade:** Como o sistema é automatizado, ele pode lidar com um grande volume de dúvidas de maneira eficiente.
- **Redução da sobrecarga para os professores:** Ao automatizar o atendimento às dúvidas mais comuns, o sistema especialista permite que os professores se concentrem em questões mais complexas e em atividades pedagógicas, sem se sobrecarregar com um grande volume de perguntas repetitivas.
- **Auxílio para professores não familiarizados com o Beecrowd:** Para professores que ainda não estão familiarizados com as questões do Beecrowd, o sistema especialista oferece uma ferramenta valiosa para auxiliá-los a responder dúvidas dos alunos de maneira eficiente, sem a necessidade de um profundo conhecimento prévio sobre a plataforma ou as questões.

5.2.2 Desvantagens

- **Limitação nas respostas para questões complexas:** O sistema especialista é eficiente para lidar com questões simples ou frequentes, mas pode encontrar dificuldades ao lidar com questões mais complexas ou específicas. Para adicionar suporte a novas questões, o professor precisa criar uma matriz de decisões detalhada, o que pode tornar o processo demorado e propenso a erros, especialmente quando se trata de questões que exigem uma análise mais aprofundada. Sem a devida cautela, o sistema pode acabar gerando respostas imprecisas ou inadequadas, comprometendo a qualidade das interações e a efetividade do suporte oferecido.
- **Dependência da qualidade da base de conhecimento:** O desempenho do sistema especialista depende diretamente da qualidade das regras e conhecimentos que foram programados nele. Se a base de dados for limitada ou desatualizada, o sistema pode fornecer respostas inadequadas.
- **Falta de interação humana:** Em casos onde o aluno necessita de uma explicação mais personalizada ou de orientação emocional, o sistema especialista pode não ser suficiente, criando uma experiência impessoal que pode não atender completamente às necessidades do aluno.

5.3 AVALIAÇÃO PRÁTICA

A avaliação prática foi conduzida por meio da utilização do sistema especialista pelos alunos da disciplina *Programação Orientada a Objetos I* da UFSC, durante o segundo semestre de 2024. Para isso, foram realizados os seguintes passos:

1. **Aplicação do sistema:** Os alunos foram incentivados a usar o sistema especialista para resolver dúvidas relacionadas às questões do Beecrowd. Durante esse período, o uso do sistema foi monitorado para avaliar sua eficácia em termos de resolução de dúvidas.
2. **Análise de resultados:** Foram avaliados o número de dúvidas resolvidas, as dúvidas não esclarecidas e os *feedbacks* fornecidos pelos alunos.
3. **Melhorias das questões:** Com base na análise dos resultados, dúvidas não resolvidas pelo Expert Bee foram incluídas no sistema para futuras turmas.

É importante destacar que, até o momento, o Expert Bee tinha apenas dicas de resoluções para essas questões da lista de exercícios fornecida, e não dicas para dúvidas reais coletadas de alunos. Essas dicas de resoluções tinham sido disponibilizadas pelo professor durante a primeira coleta de dados, descrita na seção 4.2.1, para que fosse possível testar a eficácia da ferramenta na hora em que as dúvidas dos alunos surgiriam.

Assim, um dos objetivos da avaliação prática também era identificar dúvidas não tratadas pelo Expert Bee, para que elas pudessem ser adicionadas à ferramenta.

5.3.1 Resultados

Na Avaliação Prática realizada em 18 de novembro de 2024, foi disponibilizada aos alunos uma lista com as seguintes questões do Beecrowd: 1261, 1281, 1430, 1449, 1483, 1763, 1911, 1953, 2091, 2482, 2492, 2654, 2949 e 2987, para que começassem a resolver durante a aula. 12 alunos estiveram presente na sala de aula e, sempre que um aluno apresentava uma dúvida, ele utilizava primeiro o Expert Bee para tentar resolvê-la, e, caso não conseguisse, recorria ao professor.

As dúvidas relatadas pelos alunos foram as seguintes:

- **Questão 1911:** 4 alunos apresentaram dúvidas, e todos conseguiram resolvê-las apenas com o auxílio do Expert Bee.
- **Questão 2654:** 2 alunos tiveram dúvidas, todas solucionadas pelo Expert Bee.
- **Questão 1430:** 1 aluno apresentou dúvida e conseguiu resolvê-la com o Expert Bee.

- **Questão 2949:** 1 aluno apresentou dúvida, solucionada pelo Expert Bee.
- **Questão 1483:** 1 aluno teve dúvida, também solucionada pelo Expert Bee.
- **Questão 1261:** 2 alunos apresentaram dúvidas específicas que não puderam ser resolvidas, pois o Expert Bee, à época, fornecia apenas uma ideia geral de resolução.

No total, de 11 dúvidas relatadas, o Expert Bee conseguiu resolver 9, resultando em uma taxa de sucesso de aproximadamente 81%.

5.3.2 Pontos de Melhoria

- **Questão 1261:** As dúvidas específicas dos alunos foram incorporadas ao sistema, enriquecendo o conteúdo da questão para turmas futuras.
- **Questões 1911 e 2654:** Apesar do sucesso na resolução, foi identificado um ponto de melhoria: o Expert Bee apresentava diretamente a solução completa após a primeira interação. No entanto, os alunos demonstraram interesse em orientações iniciais sobre como resolver as questões utilizando dicionários em Python. Esse ajuste foi implementado, permitindo ao usuário solicitar essas informações antes da solução.
- **Questão 1430:** A dúvida do aluno surgiu por falta de entendimento do método `split()` do Python. Como melhoria, o arquivo dessa questão no Expert Bee agora inclui uma explicação detalhada sobre o uso do método.

6 CONSIDERAÇÕES FINAIS

Em conclusão, este trabalho abordou a relevância da integração de ferramentas tecnológicas no ensino de programação, com foco especial no uso de plataformas como Beecrowd e Moodle. A análise dos juízes online e outras plataformas educacionais, como o VPL, evidenciou desafios na usabilidade e na integração, mas também destacou o Beecrowd como uma solução eficaz para o ensino de algoritmos e programação, com recursos que incentivam o aprendizado ativo, como gamificação, *feedback* em tempo real e uma gama diversificada de questões.

Além disso, este estudo aprofundou o impacto dos juízes online no ensino de programação, enfatizando a relevância da integração entre plataformas como o Beecrowd e o Moodle. A integração do Beecrowd ao Moodle, por meio da tecnologia LTI, facilita o acesso de professores e alunos a uma vasta gama de questões, otimizando o uso da plataforma no ambiente acadêmico. Além disso, oferece uma interface amigável para a criação de listas de exercícios pelos professores e a resolução de problemas de programação pelos alunos. Com o objetivo de apoiar essa integração, foi desenvolvido um manual detalhado sobre como integrar o Moodle ao Beecrowd via LTI, bem como um guia para orientar os professores no uso da plataforma.

Para estimular o uso do Beecrowd no ambiente acadêmico, também foi criada uma ferramenta com um sistema especialista que auxilia os alunos ao esclarecer dúvidas recorrentes nas questões do Beecrowd. Esse sistema, testado em sala de aula, não só oferece suporte aos estudantes, mas também facilita a adaptação dos professores, promovendo um ambiente de aprendizado mais fluido e eficiente para todos os envolvidos.

6.1 TRABALHOS FUTUROS

A seção a seguir apresenta algumas sugestões de melhorias e direções futuras para a aplicação de sistema especialista desenvolvida, a Expert Bee.

- **Adição de dicas de novas questões pela interface web:** Uma melhoria importante seria adaptar o sistema para permitir que os professores possam adicionar dicas para novas questões diretamente pela interface da aplicação web, sem a necessidade de modificar o código-fonte. Essa melhoria tornaria o sistema mais acessível e eficiente, permitindo que os educadores personalizem o conteúdo de maneira prática e intuitiva, sem a dependência de habilidades técnicas.
- **Sistema autoalimentado (aprendizado com as interações dos alunos):** Uma possível evolução seria adaptar o sistema para se autoalimentar com base nas interações dos alunos. O sistema poderia aprender com as dúvidas mais frequentes e se ajustar automaticamente, oferecendo respostas e dicas mais precisas

com o tempo. Essa melhoria contribuiria para uma experiência de aprendizagem mais personalizada, permitindo que os alunos superem suas dificuldades de forma mais eficiente.

- **Integração com ferramentas de processamento de linguagem natural (ChatGPT):** Uma adaptação interessante seria integrar o sistema especialista com ferramentas de processamento de linguagem natural, como o ChatGPT. Isso permitiria uma interação mais fluida e natural entre os alunos e o sistema, além de melhorar a precisão, contextualização e dinamismo das respostas. Com essa integração, seria possível oferecer um nível maior de personalização nas interações, aprimorando a qualidade do suporte oferecido.
- **Adição de dicas de novas questões de outras plataformas:** O Expert Bee também pode ser utilizado para fornecer dicas de questões de outras plataformas além do Beecrowd. Um possível aprimoramento seria adaptar o Expert Bee para identificar de qual plataforma as questões estão vindo e oferecer dicas específicas para essas questões. Dessa forma, seria possível ampliar a utilidade da ferramenta para diferentes plataformas de ensino, tornando-a mais flexível e adaptada a diferentes contextos de aprendizagem.

REFERÊNCIAS

ACADEMIC, Beecrowd. **Beecrowd Academic**. [*S.l.*: *s.n.*], 2024. Disponível em: <https://www.beecrowd.com.br/academic/pt>. Acesso em: 9 nov. 2023. Citado na p. 29.

BARTH, A. **HTTP State Management Mechanism**. [*S.l.*: *s.n.*], 2011. Disponível em: <https://datatracker.ietf.org/doc/html/rfc6265>. Acesso em: 13 nov. 2024. Citado na p. 67.

BEECROWD. **Beecrowd**. [*S.l.*: *s.n.*], 2024. Disponível em: <https://www.beecrowd.com.br/judge/pt>. Acesso em: 9 nov. 2023. Citado nas pp. 24–30.

BERSSANETTE, João Henrique; FRANCISCO, Antonio Carlos de. Uma Proposta de Ensino de Programação de Computadores com base na PBL utilizando o portal URI Online Judge. *In:* II SIMPÓSIO IBERO-AMERICANO DE TECNOLOGIAS EDUCACIONAIS. Araranguá: [*s.n.*], 2018. P. 348–354. Disponível em: <https://www.academia.edu/download/83230343/428-25-1253-1-10-20180622.pdf>. Acesso em: 20 jul. 2023. Citado nas pp. 17, 24.

BEZ, Jean Luca; TONIN, Neilor A. URI Online Judge e a Internacionalização da Universidade. **Revista Eletrônica de Extensão da Uri**, Erechim, v. 10, n. 18, p. 237–249, 2014. Disponível em: <https://docplayer.com.br/11099240-Uri-online-judge-e-a-internacionalizacao-da-universidade-uri-online-judge-and-the-university-internationalization.html>. Acesso em: 20 jul. 2023. Citado nas pp. 16–18, 24–27.

BEZ, Jean Luca; TONIN, Neilor A. URI Online Judge: A New Classroom Tool for Interactive Learning. *In:* THE Steering Committee Of The World Congress In Computer Science, Computer Engineering And Applied Computing (Worldcomp). Athens: [*s.n.*], 2012. P. 1–5. Disponível em: <https://search.proquest.com/openview/33325d7946128d0f1096abbe8b0b3664/1?pq-origsite=gscholar&cbl=1976352>. Acesso em: 20 jul. 2023. Citado nas pp. 17, 18, 24, 26, 29–31.

BOHANEC, Marko; BRATKO, I.; RAJKOVIC, V. Expert system for decision making. **Sistemica**, v. 1, n. 1, p. 145–157, 1990. Disponível em: <http://www-ai.ijs.si/MarkoBohanec/pub/IFIP1983.pdf>. Acesso em: 9 nov. 2024. Citado na p. 38.

BRAY, T. **The JavaScript Object Notation (JSON) Data Interchange Format**. [*S.l.*: *s.n.*], 2017. Disponível em: <https://www.rfc-editor.org/rfc/rfc8259>. Acesso em: 13 nov. 2024. Citado na p. 66.

CAMPOS, Cassio P. de; FERREIRA, Carlos E. BOCA: um sistema de apoio a competições de programação. *In:* SOCIEDADE Brasileira de Computação. São Paulo: [*s.n.*], 2004. Disponível em: <https://research.tue.nl/en/publications/boca-um-sistema-de-apoio-a-competi%C3%A7%C3%B5es-de-programa%C3%A7%C3%A3o>. Acesso em: 8 nov. 2023. Citado nas pp. 22, 23, 30, 31.

CHAVES, José Osvaldo *et al.* Uma Ferramenta Baseada em Juízes Online para Apoio às Atividades de Programação de Computadores no Moodle. **Revista Novas Tecnologias na Educação**, v. 11, n. 3, 2013. Disponível em: <https://www.researchgate.net/profile/Angelica-Castro-4/publication/332584946_Uma_Ferramenta_Baseada_em_Juizes_Online_para_Apoio_as_Atividades_de_Programacao_de_Computadores_no_Moodle/links/61bb3d081d88475981f2cc8b/Uma-Ferramenta-Baseada-em-Juizes-Online-para-Apoio-as-Atividades-de-Programacao-de-Computadores-no-Moodle.pdf>. Acesso em: 10 nov. 2024. Citado nas pp. 51, 57.

CLARK, Keith L.; MCCABE, Frank; MCCABE, F. G. **PROLOG: a language for implementing expert systems**. 1980. Diss. (Mestrado) – Imperial College of Science e Technology. Department of Computing. Disponível em: <https://d1wqtxts1xzle7.cloudfront.net/101473629/classics_210D58B0-libre.pdf?1682420871=&response-content-disposition=inline%3B+filename%3DProlog_a_language_for_implementing_exper.pdf&Expires=1731202575&Signature=fuXhBL~FdUeD8lPTRewUWnxVLkcskz28DYkkhpqSjflp4oDoPpBriAYMvFMZ-LXnkV69Ff3DdM13ofVGfLxqVdEGwBXiD3wYZzMp-hx633ZZ7tJobEHWbXQRkZeY0Y2COAZW~maaTEBmlU4NoVfp9-iVNTNSaDrtV~jm-LrqNhgLKXHYQVBP0zRMqtM0RgfaXVacxeLieUzCeVNli3wP78USukC1scfa7T-RoZSMR9YJstlyeoXgka53UQXG4EZLpT0VCdOLFiuBbF7Jj5BZPdfSOwm7WKAeKX-JyT-x3ZxxLQPAOKnqL0d9gEvo9PjiUjQ6hHJwnB-bumwc7melNw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>. Acesso em: 6 nov. 2023. Citado nas pp. 38, 39.

CODERUNNER. **CodeRunner**. [*S.l.*: *s.n.*], 2023. Disponível em: <https://coderunner.org.nz/>. Acesso em: 13 nov. 2023. Citado nas pp. 32, 33.

CRUZ, Allan Kássio Beckman Soares da *et al.* Utilização da Plataforma Beecrowd de Maratona de Programação como Estratégia para o Ensino de Algoritmos. *In:* ANAIS Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital (Sbgames

Estendido 2022). [*S.l.*]: Sociedade Brasileira de Computação, 2022. P. 754–764. Disponível em: <https://doi.org/10.5753/sbgames_estendido.2022.225898>. Acesso em: 20 jul. 2023. Citado nas pp. 16, 17, 22, 41, 42, 55, 57.

DUNSTAN, Neil. An interactive webbased expert system degree planner. **The Second International Conference on Informatics Engineering Information Science (ICIEIS2013). The Society of Digital Information and Wireless Communication**, p. 302–308, 2013. Disponível em: <https://www.researchgate.net/profile/Angelica-Castro-4/publication/332584946_Uma_Ferramenta_Baseada_em_Juizes_Online_para_Apoio_as_Atividades_de_Programacao_de_Computadores_no_Moodle/links/61bb3d081d88475981f2cc8b/Uma-Ferramenta-Baseada-em-Juizes-Online-para-Apoio-as-Atividades-de-Programacao-de-Computadores-no-Moodle.pdf>. Acesso em: 11 nov. 2024. Citado nas pp. 53, 54, 57, 58.

FERREIRA, Rafael Makaha Gomes. **BROMS: Brazilian Online Marathon Scoreboard**. 2022. Diss. (Mestrado) – Faculdade Unb Gama, Universidade de Brasília, Brasília. Disponível em: <https://bdm.unb.br/handle/10483/30740>. Acesso em: 6 nov. 2023. Citado nas pp. 17, 18.

FIELDING, R. **Hypertext Transfer Protocol – HTTP/1.1**. [*S.l.*: *s.n.*], 1999. Disponível em: <https://datatracker.ietf.org/doc/html/rfc2616>. Acesso em: 13 nov. 2024. Citado na p. 67.

FRANÇA, Allyson Bonetti; SOARES, José Marques. Sistema de apoio a atividades de laboratório de programação via Moodle com suporte ao balanceamento de carga. *In:* ANAIS do XXII SBIE - XVII WIE. [*S.l.*]: SBIE, 2011. P. 710–719. Disponível em: <https://www.researchgate.net/profile/Jose-Soares-14/publication/268337343_Sistema_de_apoio_a_atividades_de_laboratorio_de_programacao_via_Moodle_com_suporte_ao_balanceamento_de_carga/links/564309dd08aeacfd8938a73a/Sistema-de-apoio-a-atividades-de-laboratorio-de-programacao-via-Moodle-com-suporte-ao-balanceamento-de-carga.pdf>. Acesso em: 19 jul. 2023. Citado na p. 16.

FRANCISCO, Rodrigo; JÚNIOR, Cleon Pereira; AMBRÓSIO, Ana Paula. Juiz Online no ensino de Programação Introdutória - Uma Revisao Sistemática da Literatura. *In:* ANAIS do XXVII Simpósio Brasileiro de Informática na Educação (SBIE 2016). [*S.l.*: *s.n.*], 2016. P. 11–20. Disponível em: <https://www.researchgate.net/profile/Cleon-Pereira-Junior/publication/309778045_Juiz_Online_no_Ensino_de_Programacao_Introdutoria_-_Uma_Revisao_Sistematica_da_Literatura/links/582318d608aeb45b58893e6d/Juiz-

Online-no-Ensino-de-Programacao-Introdutoria-Uma-Revisao-Sistematica-da-Literatura.pdf>. Acesso em: 8 nov. 2023. Citado nas pp. 16, 22, 23.

FREITAS, Larissa Mage de. **Análise de Usabilidade do Módulo Laboratório Virtual de Programação do Moodle**. 2016. Diss. (Mestrado) – Universidade Federal de Santa Catarina, Araranguá. Disponível em: <https://repositorio.ufsc.br/handle/123456789/165178>. Acesso em: 19 jul. 2023. Citado na p. 16.

GALASSO, Rafael Hernandez; MOREIRA, Benjamin Grando. Integração do ambiente BOCA com o ambiente Moodle para avaliação automática de algoritmos. *In:* COMPUTER ON THE BEACH. Ponta Grossa: [*s.n.*], 2014. P. 22–31. Disponível em: <https://periodicos.univali.br/index.php/acotb/article/view/5292>. Acesso em: 9 nov. 2023. Citado nas pp. 29, 30, 34, 43–45, 56.

GALVÃO, Leandro; FERNANDES, David; GADELHA, Bruno. Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. *In:* ANAIS do XXVII Simpósio Brasileiro de Informática na Educação (SBIE 2016). [*S.l.*: *s.n.*], 2016. P. 140–149. Disponível em: <https://www.researchgate.net/profile/Leandro-Carvalho-3/publication/309892315_Juiz_online_como_ferramenta_de_apoio_a_uma_metodologia_de_ensino_hibrido_em_programacao/links/5da882a792851caa1babdcc5/Juiz-online-como-ferramenta-de-apoio-a-uma-metodologia-de-ensino-hibrido-em-programacao.pdf>. Acesso em: 8 nov. 2023. Citado na p. 23.

KESTEREN, A. van. **Cross-Origin Resource Sharing, W3C Working Draft WD-cors-20100727**. [*S.l.*: *s.n.*], 2010. Disponível em: <http://www.w3.org/TR/2010/WD-cors-20100727/>. Acesso em: 13 nov. 2024. Citado na p. 67.

LIMA, Gustavo Marques. **Gamma Online Judge: projeto de criação de uma plataforma para o armazenamento das questões das maratonas UnB de programação**. 2022. Diss. (Mestrado) – Faculdade Unb Gama, Universidade de Brasília, Brasília. Disponível em: <https://bdm.unb.br/handle/10483/34516>. Acesso em: 6 nov. 2023. Citado nas pp. 17, 18, 23.

LIMA, José Maria Maciel. Plataforma Moodle: a educação por mediação tecnológica. **Revista Científica Multidisciplinar Núcleo Do Conhecimento**, v. 7, n. 6, p. 17–37, 2021. Disponível em:

<https://www.nucleodoconhecimento.com.br/educacao/plataforma-moodle>. Acesso em: 19 jul. 2023. Citado na p. 16.

LOBB, Richard; HARLOW, Jenny. Coderunner: a tool for assessing computer programming skills. *In:* 1. ACM Inroads. Ponta Grossa: [*s.n.*], 2016. P. 47–51. Disponível em: <https://dl.acm.org/doi/abs/10.1145/2810041?casa_token=cV5554XMVswAAAAA:8SVdEhKBneuW43kadkaSv8ELNdN41F376mTCSo7a98yPV9CJFAp-8vADXTAbD9nURnOLmzne-W_f>. Acesso em: 16 nov. 2023. Citado nas pp. 17, 32, 33, 45–47, 55, 56.

MASSE, Mark. **REST API design rulebook**. [*S.l.*: *s.n.*], 2011. Disponível em: <https://www.swi-prolog.org/pldoc/doc_for?object=section(%27packages/http.html%27>. Acesso em: 13 nov. 2024. Citado na p. 66.

MASSÉ, Mark. **REST API Design Rulebook**. [*S.l.*]: Sebastopol: O'reilly Media, Inc., 2011. Disponível em: <https://books.google.com.br/books?hl=pt-PT&lr=&id=eABpzyTcJNIC&oi=fnd&pg=PR3&dq=REST+API+Design+Rulebook&ots=vBOvZ_jdMD&sig=tvXanZ39ViurHJ7fWrRIovBwlgI>. Acesso em: 13 nov. 2023. Citado na p. 40.

MOODLE. **Moodle Developer Resource centre**. [*S.l.*: *s.n.*], 2023. Disponível em: <https://moodledev.io/>. Acesso em: 10 nov. 2023. Citado nas pp. 32–36.

NASR, Osman A.; TALAB, Ahmed; AL-GAHTANI, Said S. **Building of a Rule-Based Expert System for Academic Advising via Web Expert System Tools**. 2018. Diss. (Mestrado) – FKing Khalid University - College of Business - Dept. of MIS. Disponível em: <https://d1wqtxts1xzle7.cloudfront.net/57393851/17-44-2-11-libre.pdf?1537170728=&response-content-disposition=inline%3B+filename%3DBuilding_of_a_Rule_Based_Expert_System_f.pdf&Expires=1731285754&Signature=C37Cih002J7H1n9uza0UAvGlGagiZrKvuAZEfy~W62aqwIJ16r7EE2p~dfpFUYSdCeB4nuDA7T5-Q2cnEZQ3kcuHanae2lk2emCT0KW6IsfNmCmr15uEMMnL3LYIc~54H3R4cC7dehTcCUSV-LqCJXBXnGawDWj-VZHGWsqUu5pqUlNDIxO59ZZvEWLr5l2Le8rKIF~XTIKW7x1V8vC-e8KYjtacSSt4f8hZID~a5AC61al2hGuv2DWqbt2I1hy7gZLGqu5iIBuDh11I~Zvw6gco~rGB42xA3lG1LCDRzqJ93jA7KExsDFuVXglwPSLb5qIEd4~cj~TMnIUNq2~aZg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>. Acesso em: 11 nov. 2024. Citado nas pp. 52, 57, 58.

PIEKARSKI, Ana Elisa Tozetto *et al.* PROGRAMAÇÃO COMPETITIVA EM UM PROJETO DE EXTENSÃO PARA O ENSINO TÉCNICO EM INFORMÁTICA. **Revista Conexão UEPG**, Ponta Grossa, v. 19, n. 1, p. 1–15, 2023. Disponível em:

<https://revistas.uepg.br/index.php/conexao/article/view/21239>. Acesso em: 9 nov. 2023. Citado na p. 24.

RIBEIRO, Ralph Breno *et al.* Gamificação de um Sistema de Juiz Online para Motivar Alunos em Disciplina de Programação Introdutória. *In:* ANAIS do XXIX Simpósio Brasileiro de Informática na Educação (SBIE 2018). [*S.l.*: *s.n.*], 2018. P. 805–814. Disponível em: <http://milanesa.ime.usp.br/rbie/index.php/sbie/article/view/8040>. Acesso em: 8 nov. 2023. Citado na p. 22.

RICHARDSON, Leonard; AMUNDSEN, Mike; RUBY, Sam. **RESTful Web APIs**. [*S.l.*]: Sebastopol: O'reilly Media, Inc., 2011. Disponível em: <https://books.google.com.br/books?hl=pt-PT&lr=&id=wWnGAAAAQBAJ&oi=fnd&pg=PR2&dq=restful+web+apis&ots=Fi9itG-75e&sig=q64KUvXJzi4zFGQhnJnIzxjL2LM>. Acesso em: 13 nov. 2023. Citado na p. 40.

RODRÍGUEZ-DEL-PINO, Juan Carlos. **Moodle plugins directory: Virtual Programming Lab**. [*S.l.*: *s.n.*], 2023. Disponível em: <https://moodle.org/plugins/mod_vpl>. Acesso em: 19 jul. 2023. Citado na p. 16.

RODRÍGUEZ-DEL-PINO, Juan Carlos; ROYO, Enrique Rubio; FIGUEROA, Zenón Hernández. A virtual programming lab for Moodle with automatic assessment and anti-plagiarism features. *In:* THE 2012 INTERNATIONAL CONFERENCE ON E-LEARNING, E-BUSINESS, ENTERPRISE INFORMATION SYSTEMS, E-GOVERNMENT. Hong Kong: [*s.n.*], 2012. Disponível em: <http://hdl.handle.net/10553/9773>. Acesso em: 16 nov. 2023. Citado nas pp. 31, 48–50, 56, 57.

RUSSELL, Stuart; NORVIG, Peter. Inteligência Artificial: tradução da terceira edição. *In:* Disponível em: <https://www.kufunda.net/publicdocs/Intelig%C3%AAncia%20Artificial%20(Peter%20Norvig,%20Stuart%20Russell).pdf>. Acesso em: 10 nov. 2024. Citado na p. 39.

SALES, André Barros de; JUNIOR, Edson Costa; SALES, Márcia Barros de. Utilização de Problemas da Maratona de Competição de Programação e Juízes Eletrônicos como Estratégia de Ensino em um Curso de Graduação em Engenharia de Software. *In:* ANAIS do XXVII Simpósio Brasileiro de Informática na Educação (SBIE 2016). [*S.l.*: *s.n.*], 2016. P. 210–219. Disponível em: <http://milanesa.ime.usp.br/rbie/index.php/sbie/article/view/6701>. Acesso em: 8 nov. 2023. Citado na p. 23.

SANTOS, Joanna C. S.; RIBEIRO, Admilson R. L. JOnline: proposta preliminar de um juiz online didático para o ensino de programação. *In:* ANAIS do Simpósio Brasileiro de Informática na Educação (SBIE 2011). [*S.l.*: *s.n.*], 2011. P. 964–967. Disponível em: <http://milanesa.ime.usp.br/rbie/index.php/sbie/article/view/1863>. Acesso em: 8 nov. 2023. Citado nas pp. 22, 23.

SINGLA, Jimmy. The diagnosis of some lung diseases in a prolog expert system. **International Journal of Computer Applications**, v. 78, n. 15, p. 37–40, 2013. Disponível em: <https://d1wqtxts1xzle7.cloudfront.net/47387099/pxc3891435-libre.pdf?1469050904=&response-content-disposition=inline%3B+filename%3DThe_Diagnosis_of_Some_Lung_Diseases_in_a.pdf&Expires=1731202594&Signature=K2HWohOs96t-TOsee9D56TwDTQ8bnsejjvZ~htc5V5ip0dTMBYdFB2976WEyKqDFfbNMOLVpMd9BW-3mEROPqGwLFr361QYC5WyCXxNtMOOY8s6kJUzHO5PlULctIg9EVVorrTMyyRCyB65Azm-pA3eWtwKtfFzwx7x5RSnSiuhYqFYbpF0BqOGmUZ8DhkGZt8~yO7J14W9xsTcMgtdYYaiajRbPcxgixepiWlOBbYZRwmDgN0W03qBrhXKLqy69tFcNzPNWs1PU~NZh5uDgb7lzBTQvi9UYU3R61g6zwokJF1m7Am6GfL6vk~MMje6dtXQL0Jq3HzDN3iUH~jQUmA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>. Acesso em: 9 nov. 2024. Citado nas pp. 38, 39.

VERDAGUER, Júlia. **O que é LTI e como ele pode melhorar seu ecossistema de aprendizagem**. [*S.l.*: *s.n.*], 2021. Disponível em: <https://moodle.com/pt-br/news/o-que-e-e-como-pode-melhorar-seu-ecossistema-de-aprendizagem/>. Acesso em: 14 nov. 2023. Citado nas pp. 19, 37.

VICKERS, Stephen P; BOOTH, Simon. Creating Environments For Learning Through Instigating A Community Of Developers: A JISC-funded Project. **Learning Tools Interoperability (LTI): A Best Practice Guide**, 2014. Disponível em: <https://ltiapps.net/guide/LTI_Best_Practice.pdf>. Acesso em: 14 nov. 2023. Citado na p. 37.

VPL. **Virtual Programming Lab for Moodle (VPL)**. [*S.l.*: *s.n.*], 2021. Disponível em: <https://vpl.dis.ulpgc.es/>. Acesso em: 13 nov. 2023. Citado nas pp. 31, 32.

WASIK, Szymon *et al.* A Survey on Online Judge Systems and Their Applications. **Acm Computing Surveys**, v. 51, n. 1, p. 1–34, 2018. Disponível em: <https://dl.acm.org/doi/abs/10.1145/3143560>. Acesso em: 8 nov. 2023. Citado na p. 22.

WIELEMAKER, J.; HUANG, Z.; MEIJ, L. VAN DER. SWI-Prolog and the web. *In:* THEORY and Practice of Logic Programming. [*S.l.*: *s.n.*], 2008. P. 363–392. Disponível

em: <https://doi.org/10.1017/S1471068407003237>. Acesso em: 10 nov. 2024. Citado nas pp. 39, 40.

WIELEMAKER, J.; SCHRIJVERS, T. *et al.* SWI-Prolog. *In:* THEORY and Practice of Logic Programming. [*S.l.*: *s.n.*], 2012. P. 67–96. Disponível em: <https://doi.org/10.1017/S1471068411000494>. Acesso em: 10 nov. 2024. Citado na p. 39.

WIELEMAKER, Jan. *In:* SWI-PROLOG 6.1: Reference Manual. [*S.l.*: *s.n.*], 2012. Disponível em: <https://rabbit.eng.miami.edu/class/een594/SWI-Prolog-6.1.2.pdf>. Acesso em: 10 nov. 2024. Citado na p. 39.

WIELEMAKER, Jan. **SWI-Prolog HTTP support**. [*S.l.*: *s.n.*], 2024. Disponível em: <https://www.swi-prolog.org/pldoc/doc_for?object=section(%27packages/http.html%27)>. Acesso em: 13 nov. 2024. Citado na p. 66.

WIELEMAKER, Jan. SWI-Prolog version 7 extensions. *In:* WORKSHOP on Implementation of Constraint and Logic Programming Systems and Logic-based Methods in Programming Environments. [*S.l.*: *s.n.*], 2014. Disponível em: <https://www.swi-prolog.org/download/publications/swi7.pdf>. Acesso em: 10 nov. 2024. Citado na p. 39.

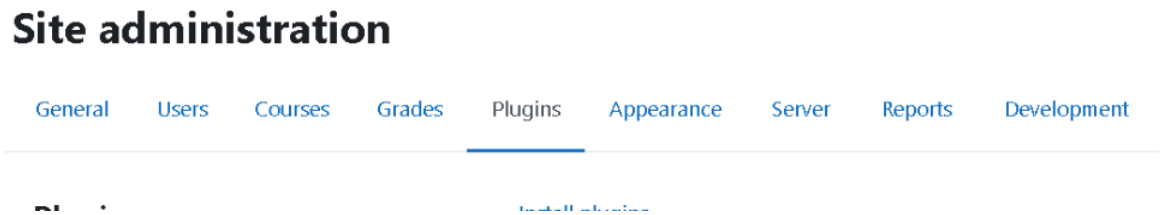
WIELEMAKER, Jan; LAGER, Torbjorn; RIGUZZI, Fabrizio. SWISH: SWI-Prolog for Sharing. *In:* Disponível em: <https://arxiv.org/pdf/1511.00915>. Acesso em: 10 nov. 2024. Citado na p. 40.

Apêndices

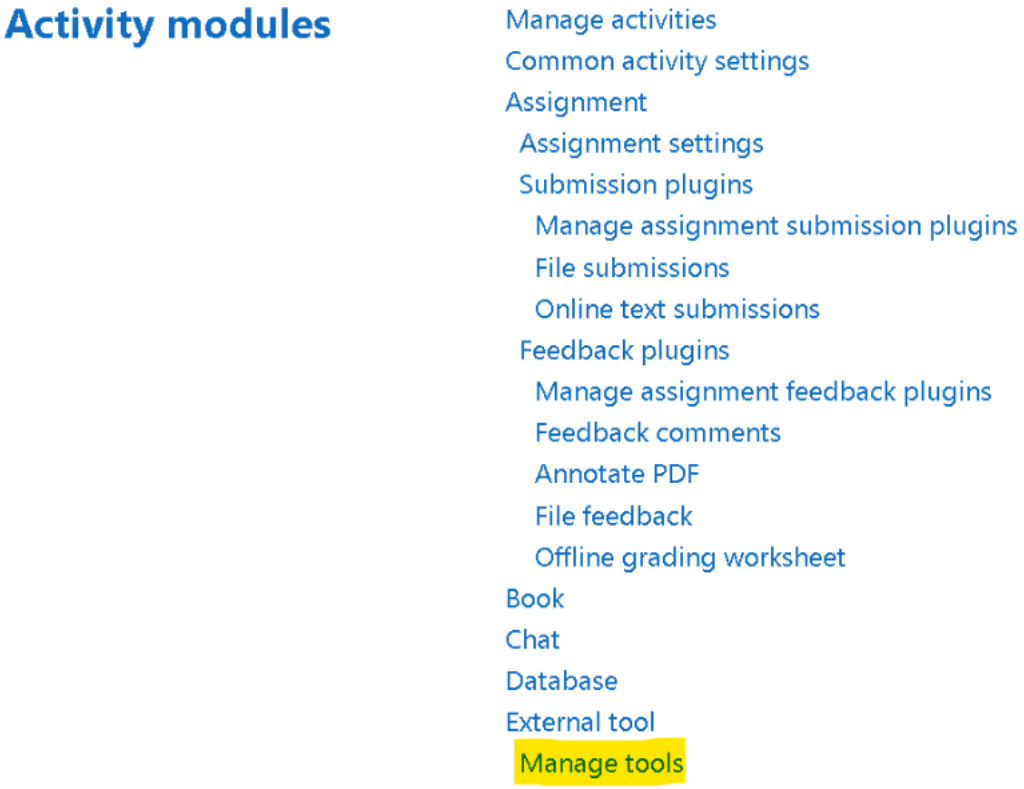
APÊNDICE A – MANUAL DE CONFIGURAÇÃO DA LTI BEECROWD NO MOODLE

O administrador do Moodle deve configurar a LTI do Beecrowd.

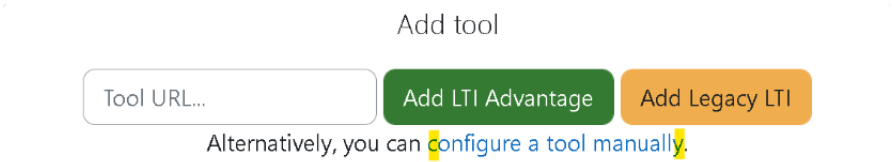
1. Acessar o moodle com uma conta de administrador
2. Selecionar a opção: **Plugins**



3. Em Activity Modules, selecionar **External tool** -> **Manage tools**



4. Selecionar a opção **configure a tool manually**



5. Na página a seguir, preencher os campos conforme abaixo:

   Tool name: *beecrowd*
   Tool URL: *https://api.beecrowd.com/*
   Tool description: *beecrowd LTI1.3 moodle integration*
   LTI version: *LTI 1.3*
   Public key type: *Keyset URL*
   Public keyset: *https://api.beecrowd.com/lti/jwks*
   Initiate login URL: *https://api.beecrowd.com/lti/login*
   Redirection URI(s): *https://api.beecrowd.com/lti/launch*
   Custom parameters: “Manter vazio”
   Tool configuration usage: *Show in activity chooser and as a preconfigured tool*
   Default launch container: *New window*
   Supports Deep Linking (Content-Item Message): *“Assinalar o check box”*
   Content Selection URL: *https://api.beecrowd.com*

   Icon URL: *https://moodle.beecrowd.io/pluginfile.php/15/course/summary/lgo.png*
   Secure icon URL: *https://moodle.beecrowd.io/pluginfile.php/15/course/summary/lgo.png*

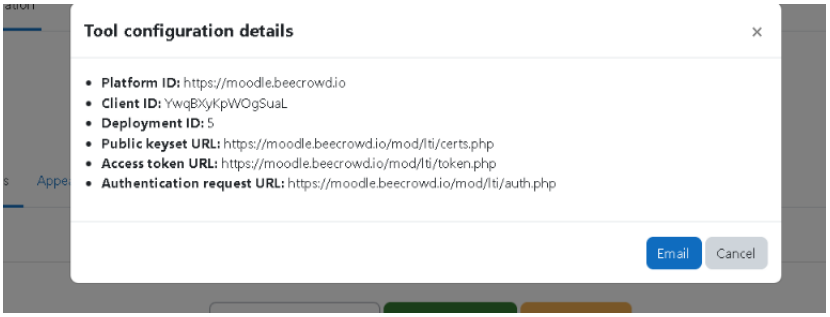
   Services
   IMS LTI Assignment and Grade Services: *Use this service for grade sync and column management*
   IMS LTI Names and Role Provisioning: *Use this service to retrieve members' information as per privacy settings*
   Tool Settings: *Use this service*

   Privacy
   Share launcher's name with tool: *Always*
   Share launcher's email with tool: *Always*
   Accept grades from the tool: *Always*
   Force SSL: “Não assinalar”

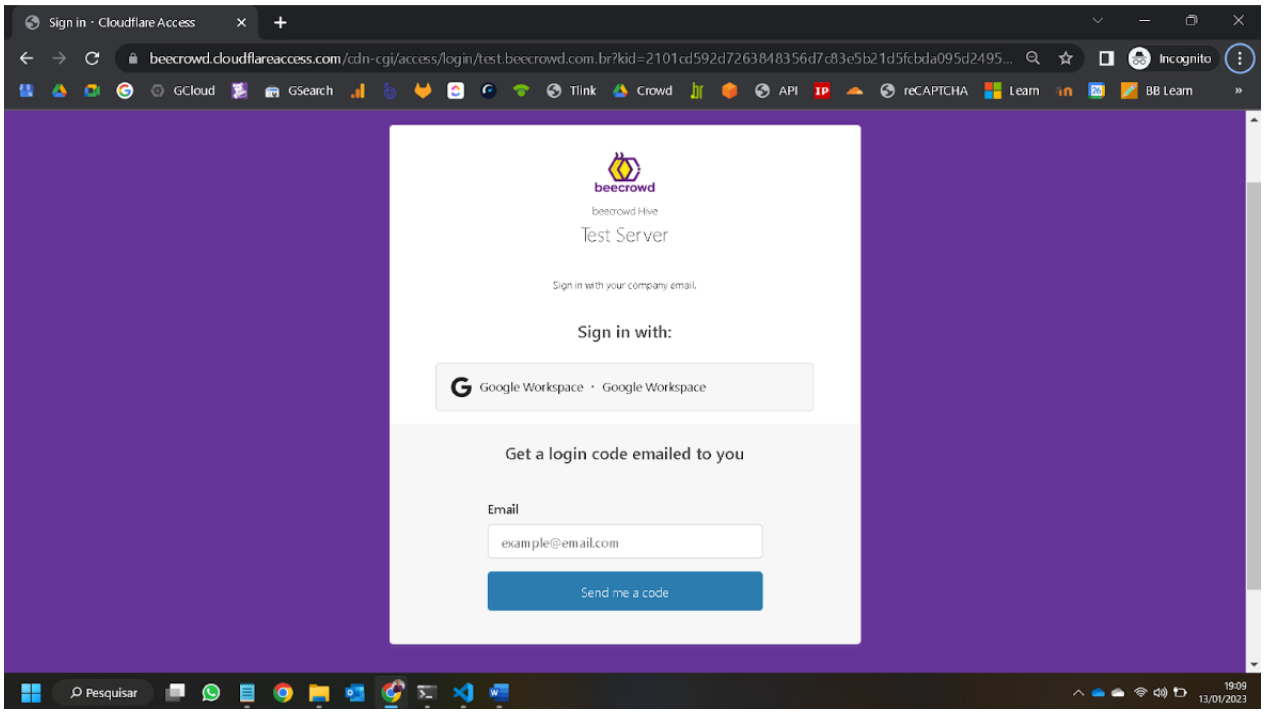
   Miscellaneous
   Default organisation ID: *Site hostname*
   Organisation ID: “Manter Vazio”
   Organisation URL: *https://beecrowd.com/*

6. Após configurar as informações acima, você deve enviar para o Beecrowd a

captura de tela da seguinte tela (exemplo na foto abaixo) através do botão de e-mail com os dados que eles devem inserir na parte deles para estabelecer a comunicação:



7. Será exibida uma página de autorização. Basta preencher o email e aguardar a aprovação. Uma vez feito, é necessário recarregar a sessão e iniciar os testes novamente.



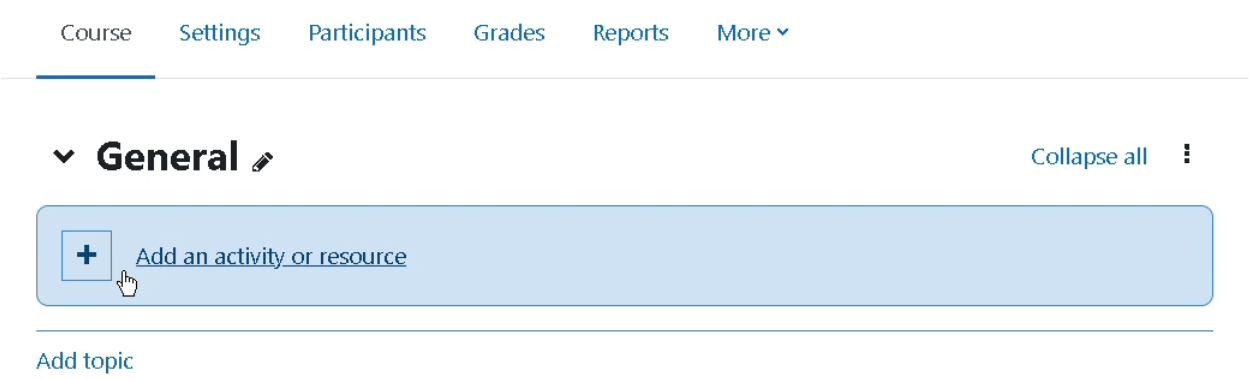
APÊNDICE B – MANUAL DE USO LTI BEECROWD NO MOODLE

B.1 UTILIZAR LTI DO BEECROWD JÁ CONFIGURADA NO MOODLE

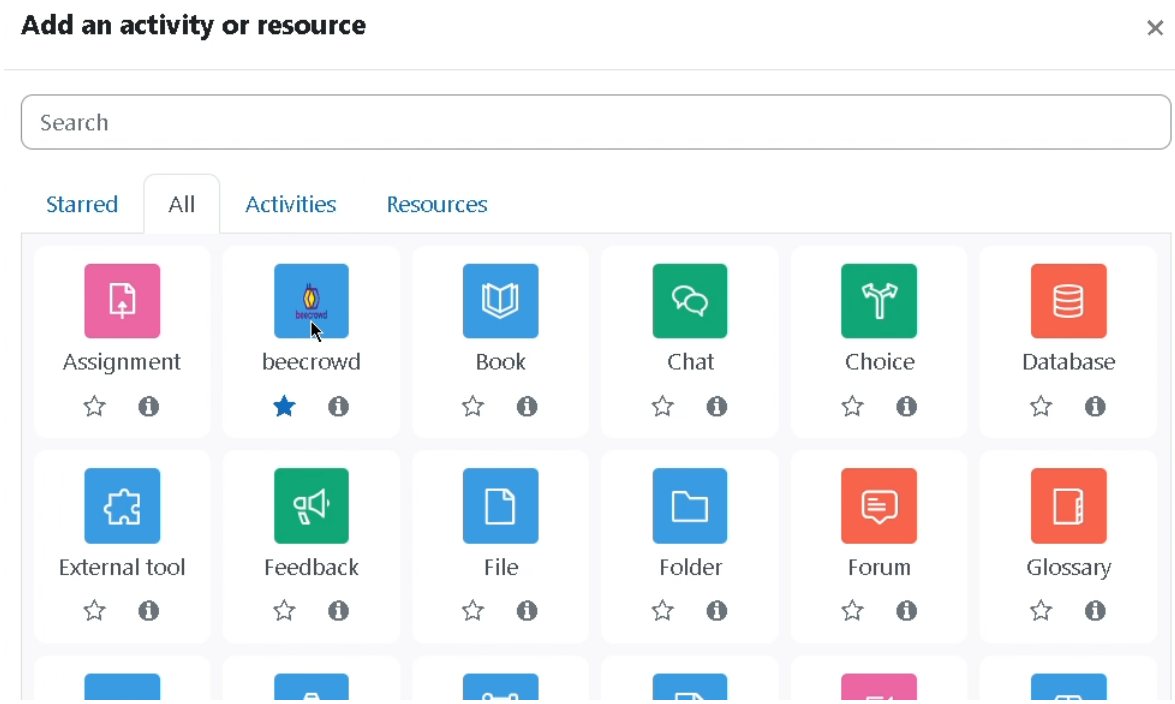
Este Manual mostra como o professor cria uma atividade Beecrowd no Moodle, como um aluno acessa essa atividade, e como o professor pode transferir as notas do Beecrowd para o Moodle.

B.1.1 Criando a atividade

1. Entre em modo edição
2. Vá em adicionar uma atividade ou recurso:



3. Selecione a atividade “beecrowd”



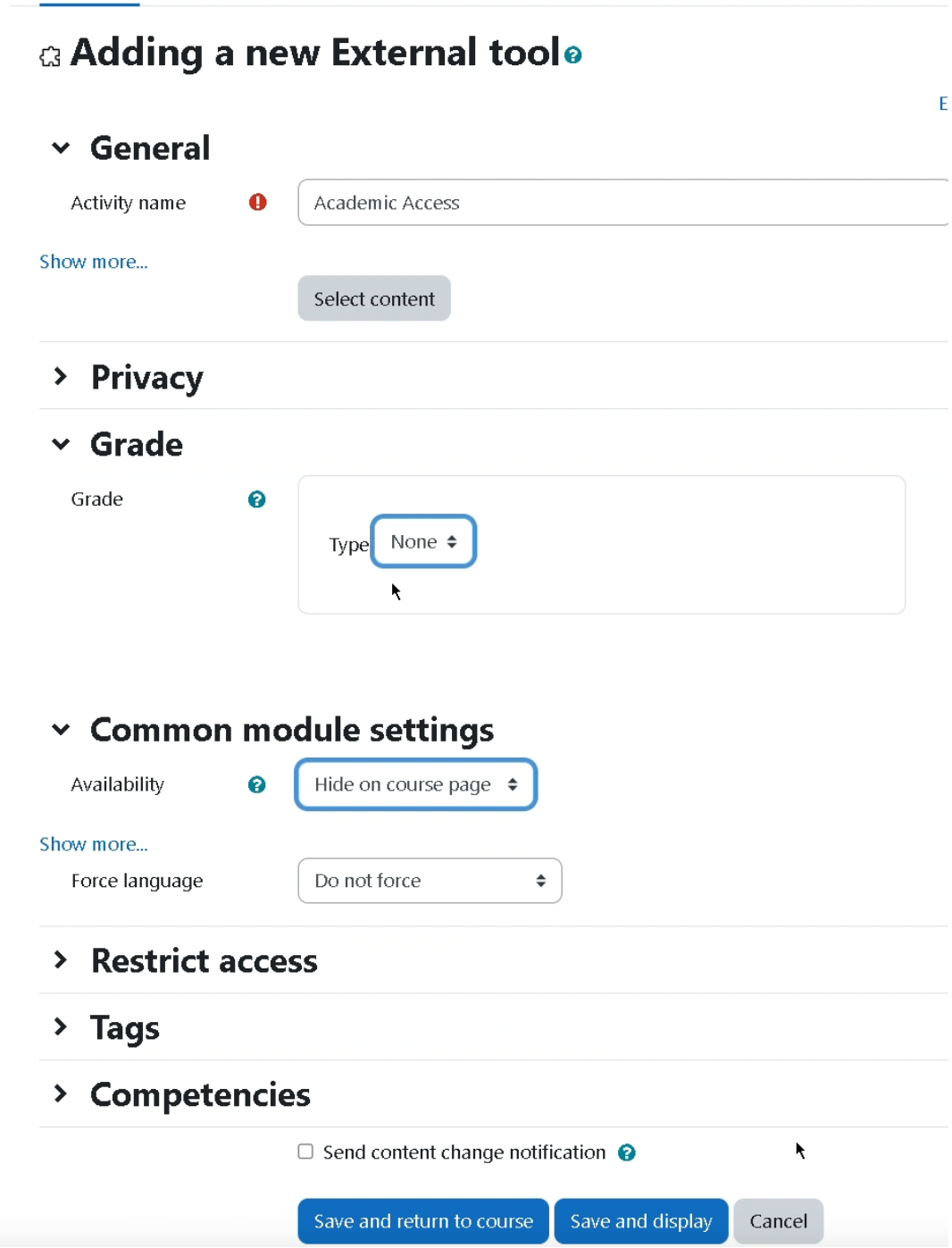
4. Na página que abrir:

   a) Em **General** > **Activity Name**, escreva o nome da atividade - como essa primeira vai ser para uso do professor, para entrar no Beecrowd Academic, você pode escolher um nome tipo *"Acesso ao Beecrowd Acadêmico"*;

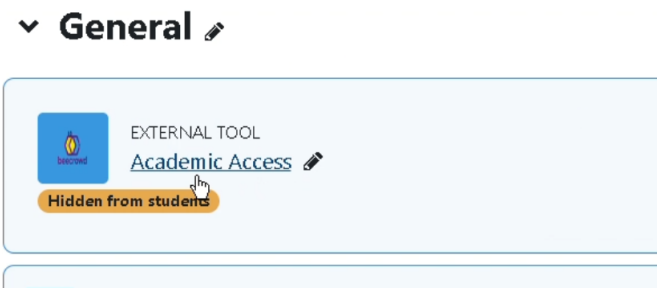
   b) Em **Grade** > **Type**, selecione *None*;

   c) Em **Common module settings** > **Availability**, selecione *"Hidden on Course Page"*;

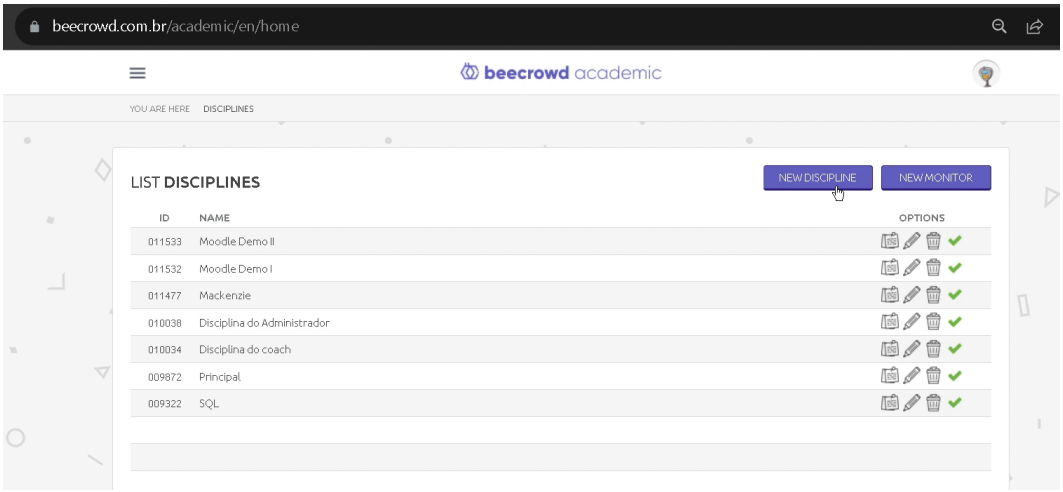
   d) Clique em **Save and return to course**.



5. Agora, a atividade que você criou vai aparecer na página do curso. Clique nela.

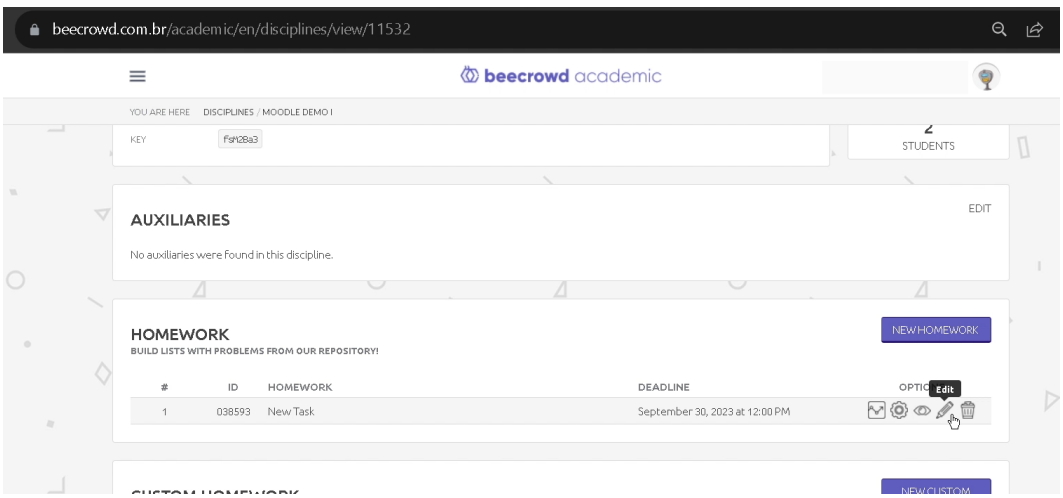


6. Você vai ser redirecionado para a página do Beecrowd Academic. Caso você já esteja logado, vai aparecer a lista de suas disciplinas.

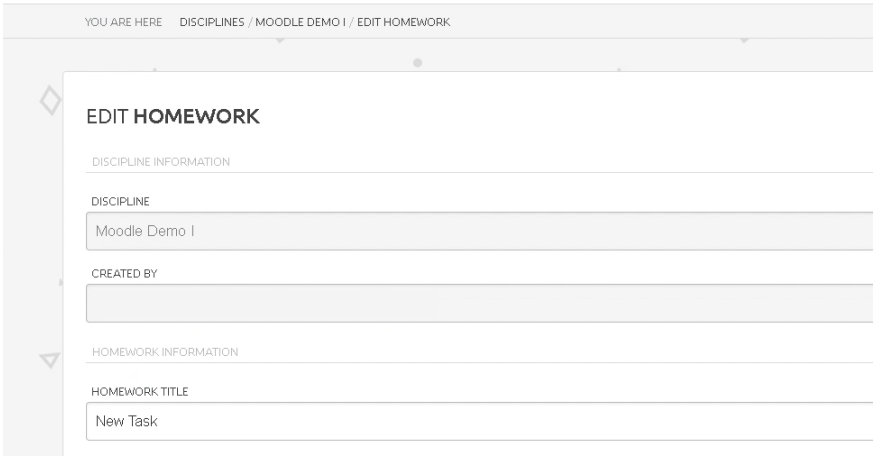


Você pode criar uma disciplina, ou selecionar uma disciplina já existente. Para cada disciplina, você pode criar tarefas, e, para cada tarefa, você pode selecionar diferentes questões, e arrumar outras configurações, como data de início, de fim, etc (Veja as imagens da próxima página).

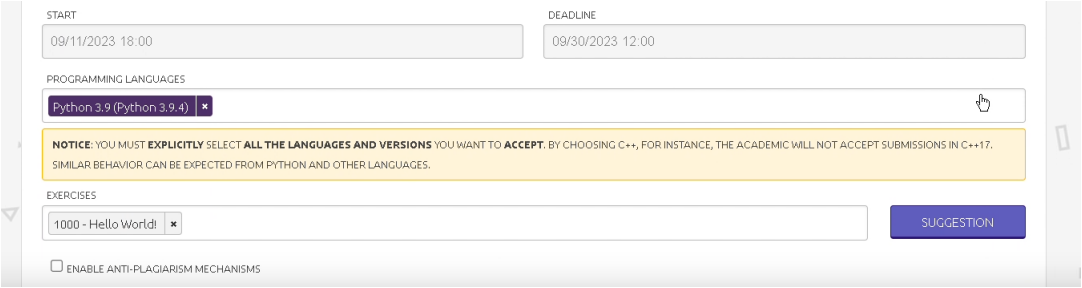
Criar ou editar tarefa:



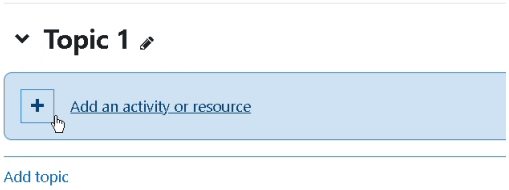
Editando a tarefa:



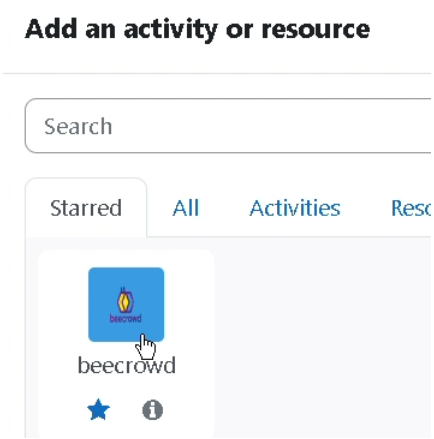
Selecionando questões:



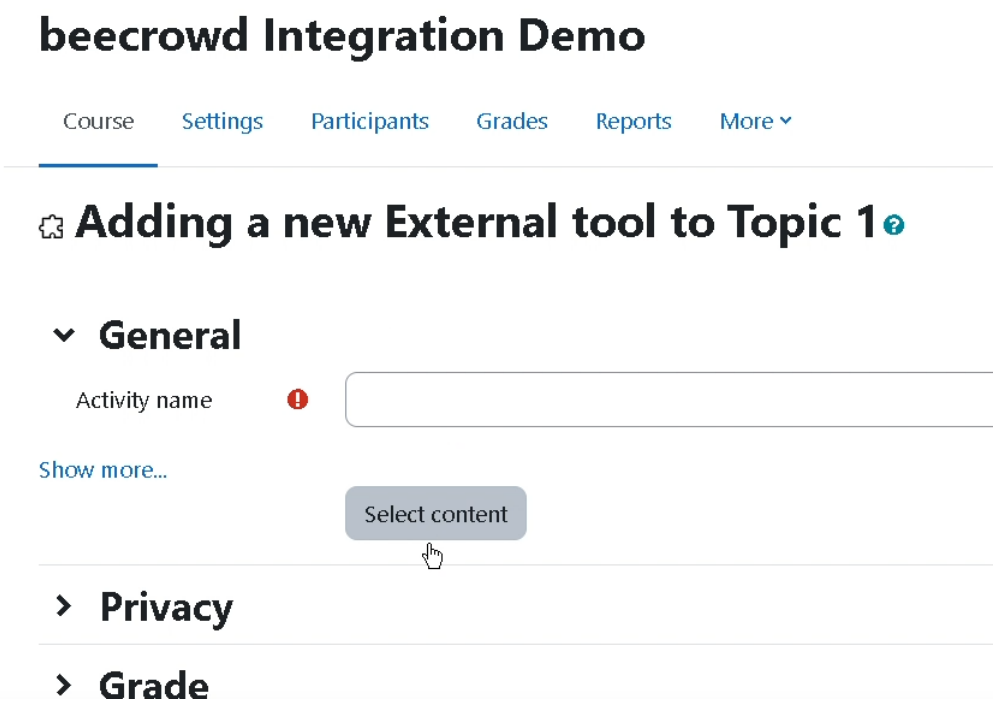
7. Agora que você tem a disciplina com a tarefa desejada, volte ao Moodle e clique para adicionar uma atividade ou recurso:



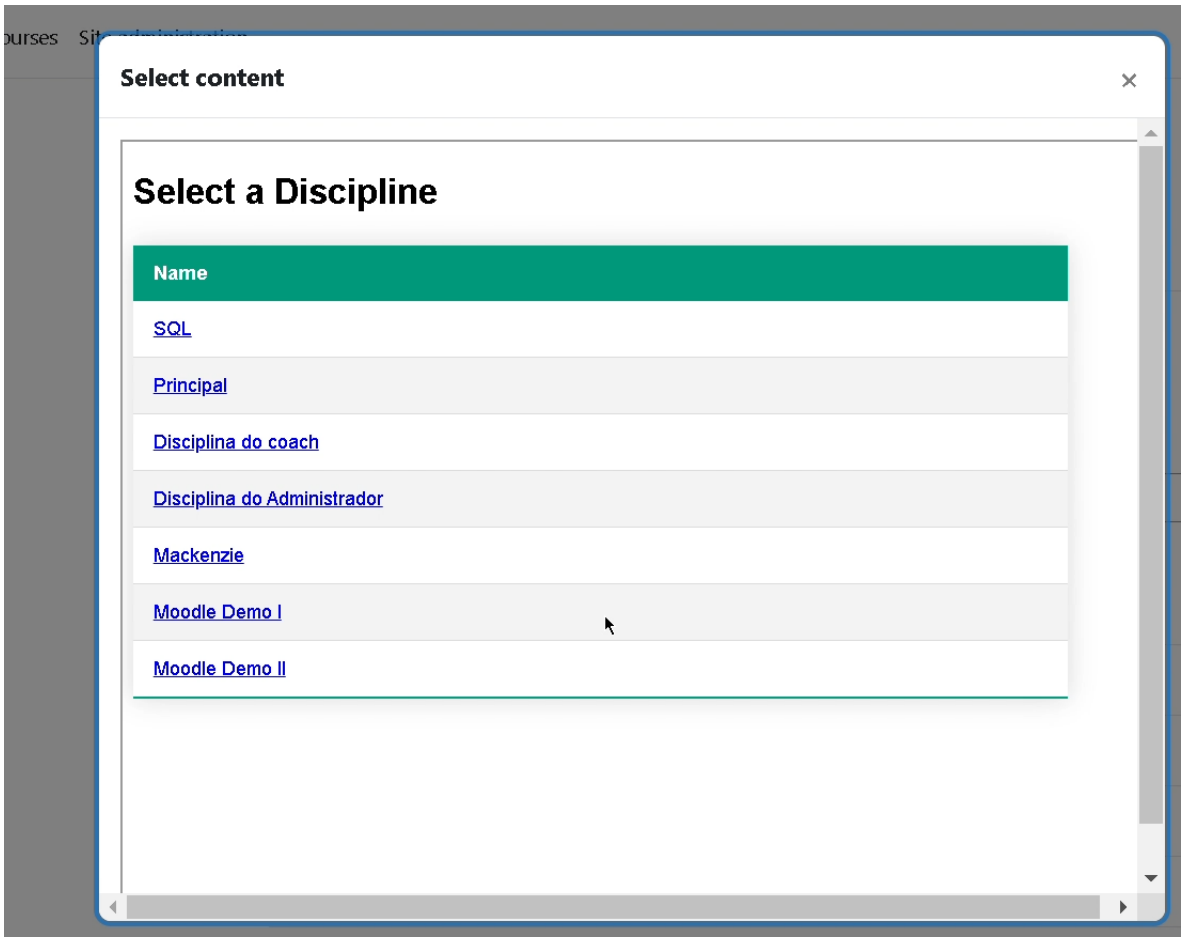
8. Selecione a atividade Beecrowd:



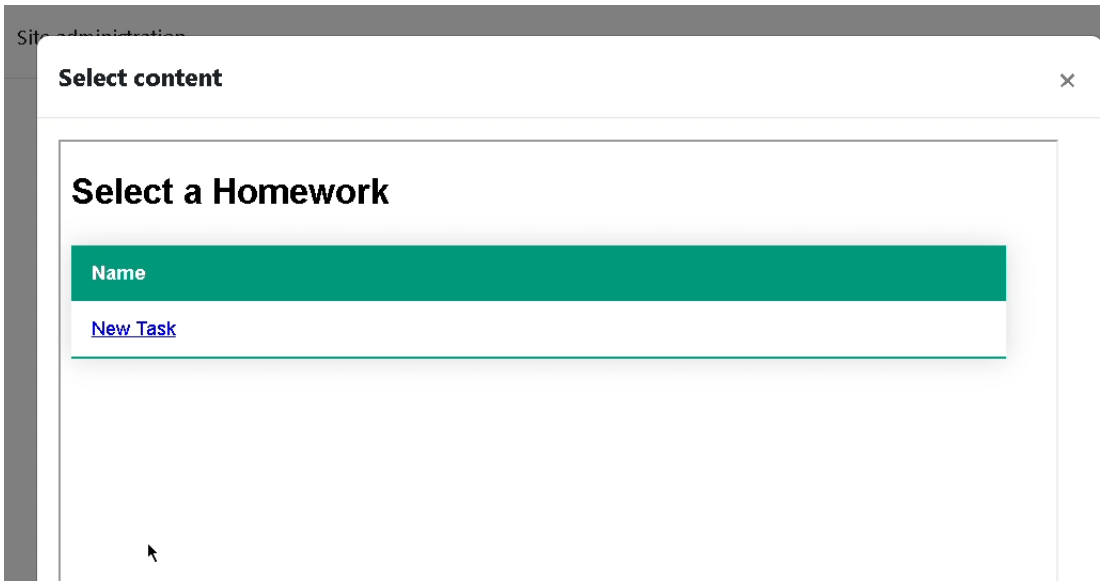
9. Clique em “Selecionar conteúdo”:



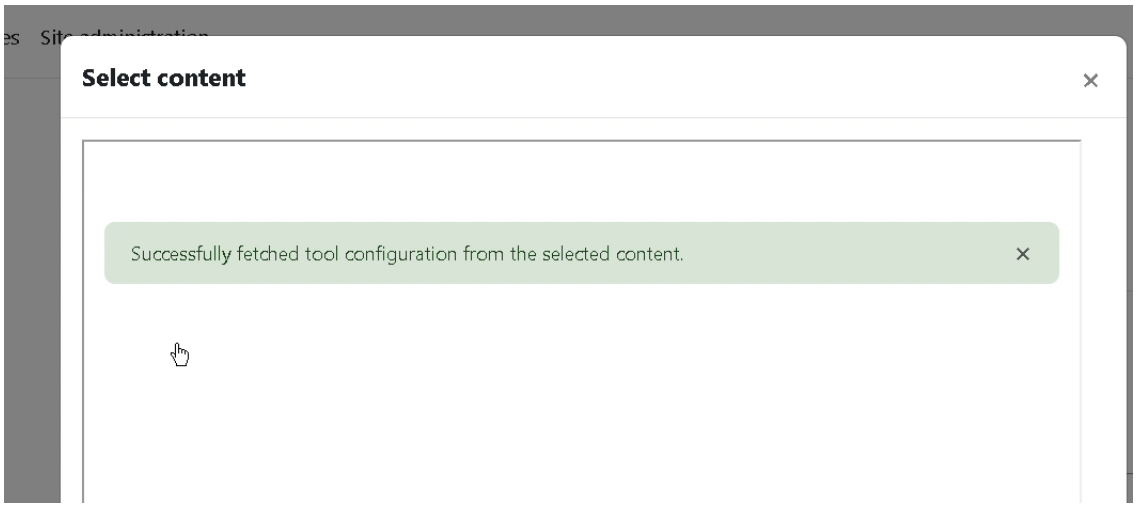
10. Selecione a Disciplina criada no Beecrowd Academic:



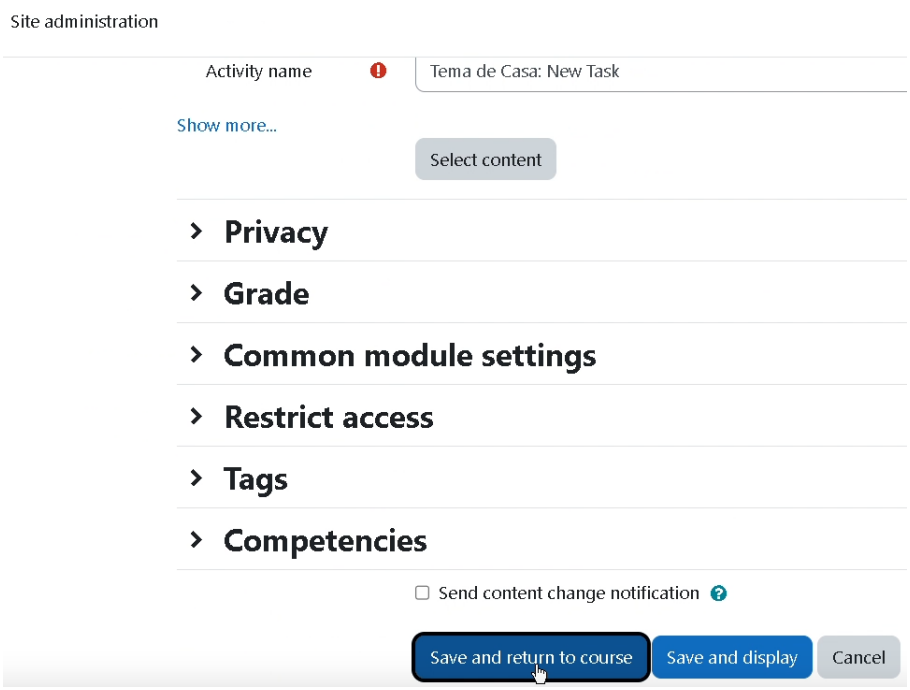
11. Selecione a tarefa criada no Beecrowd Academic:



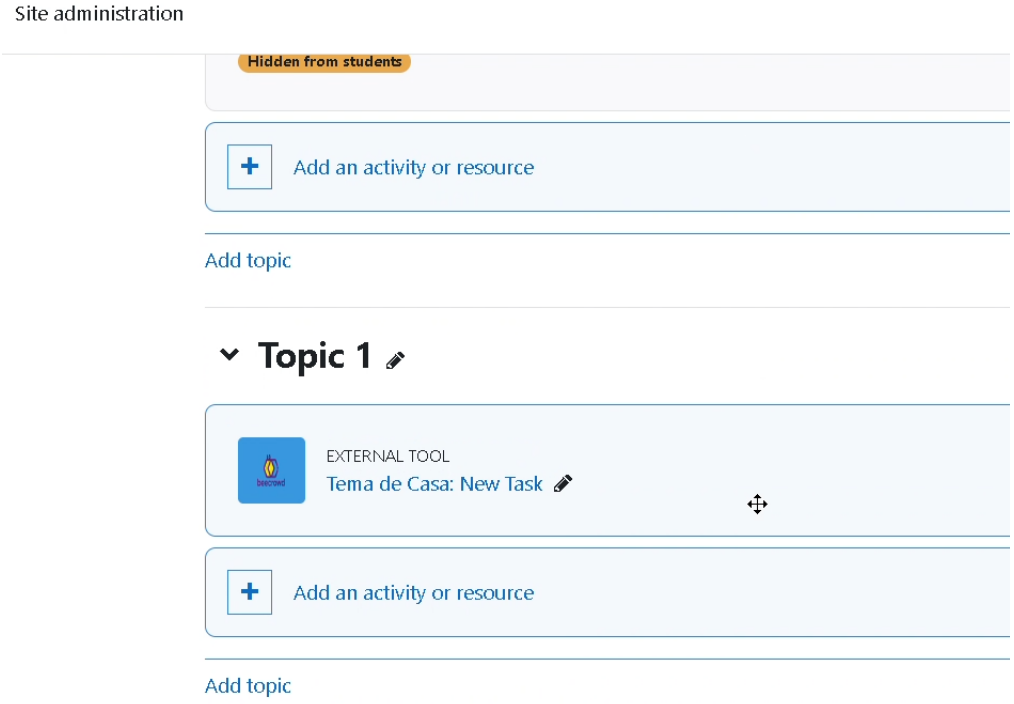
Vai aparecer uma mensagem de confirmação que a tarefa foi selecionada com sucesso:



12. Você vai ser redirecionado de volta para a tela anterior, e agora é só clicar em “Save and return do course”:

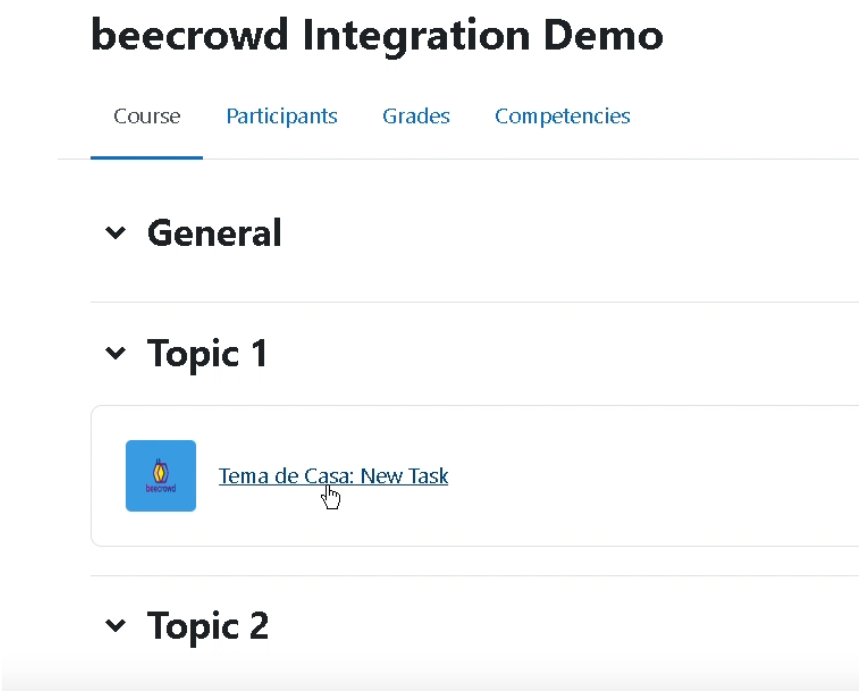


Agora a tarefa aparecerá dessa forma na página do curso:

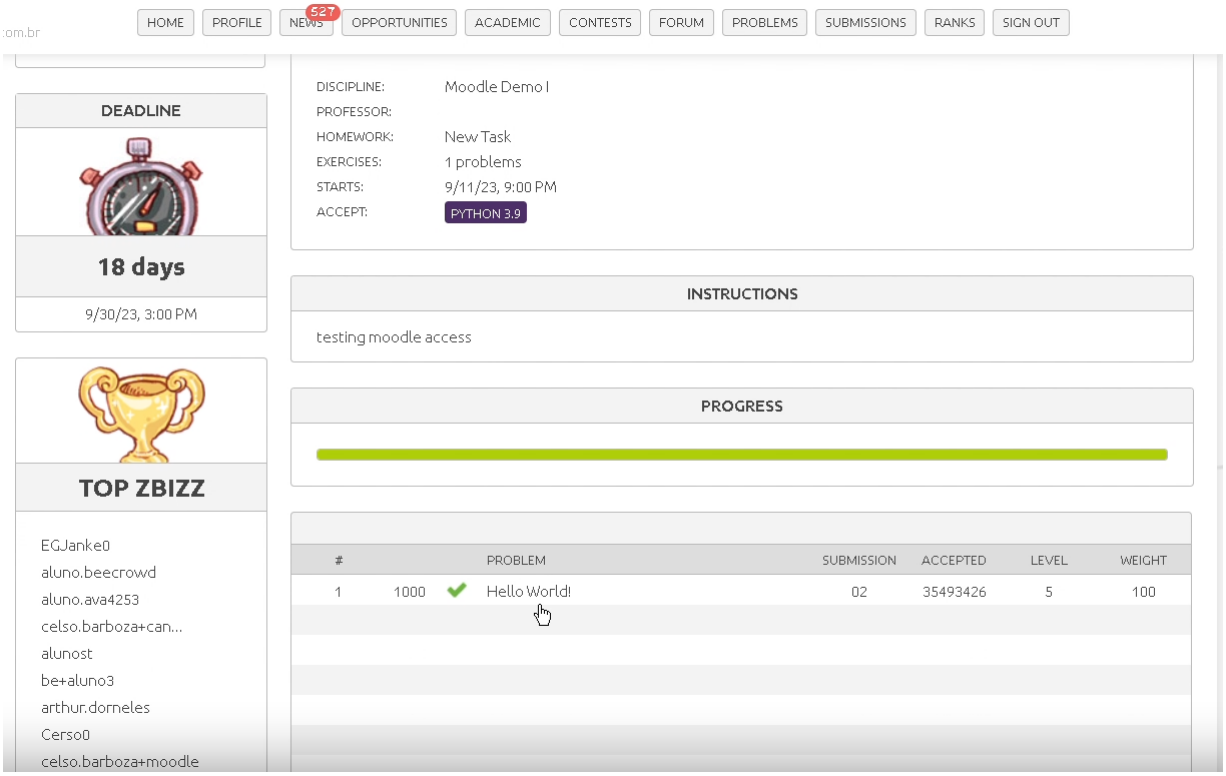


B.1.2 Entrando como aluno

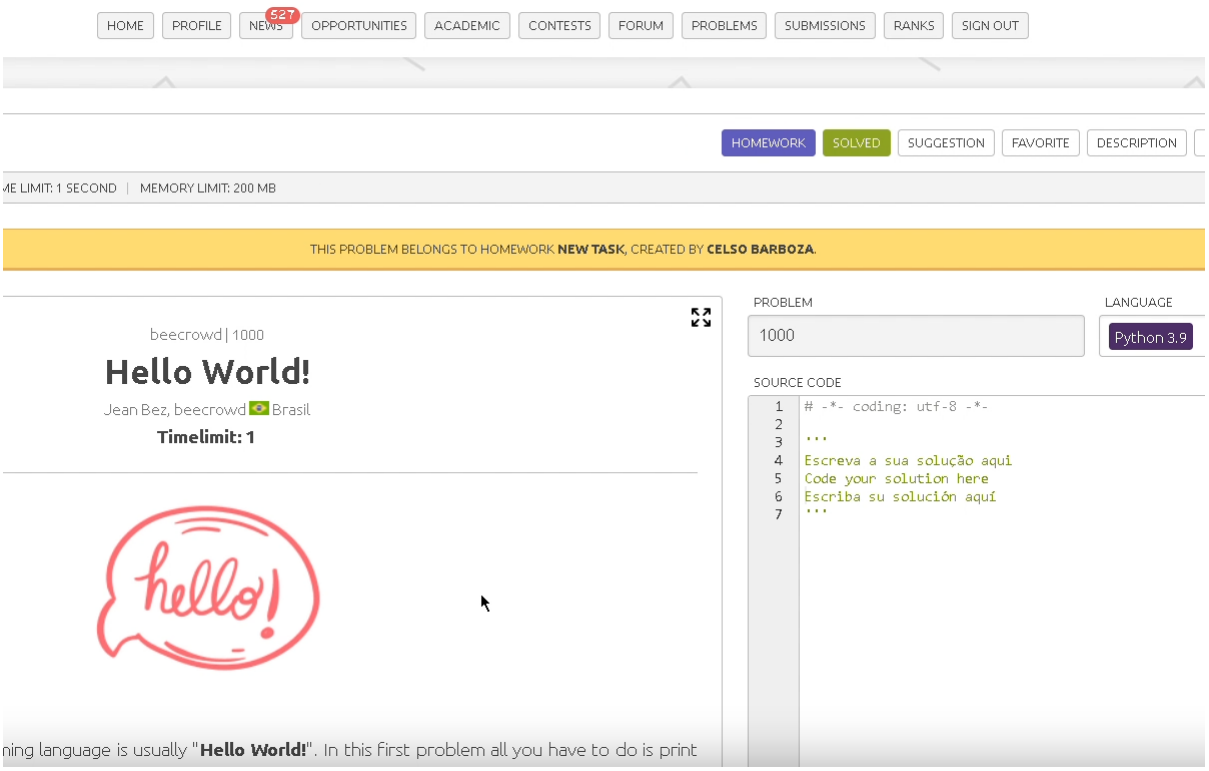
1. O aluno clica na tarefa do Beecrowd, criada pelo professor:



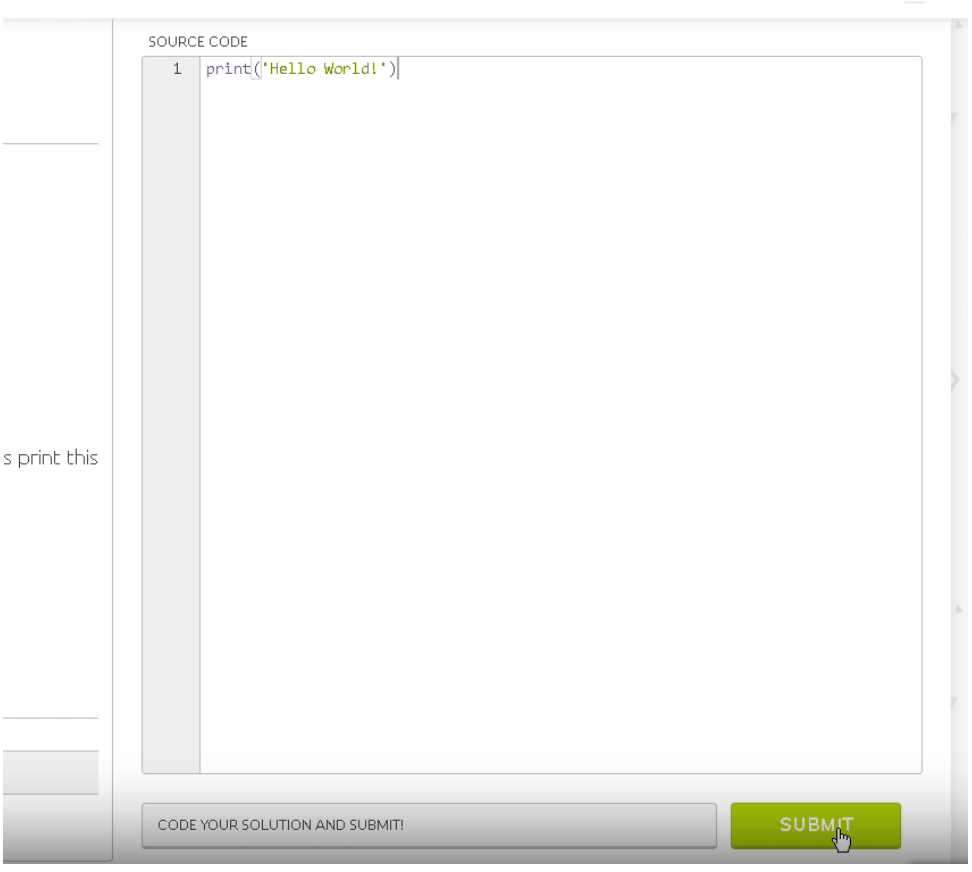
2. O aluno é direcionado à tarefa do Beecrowd, sem precisar criar um usuário, e já pode começar a resolver os exercícios da tarefa:



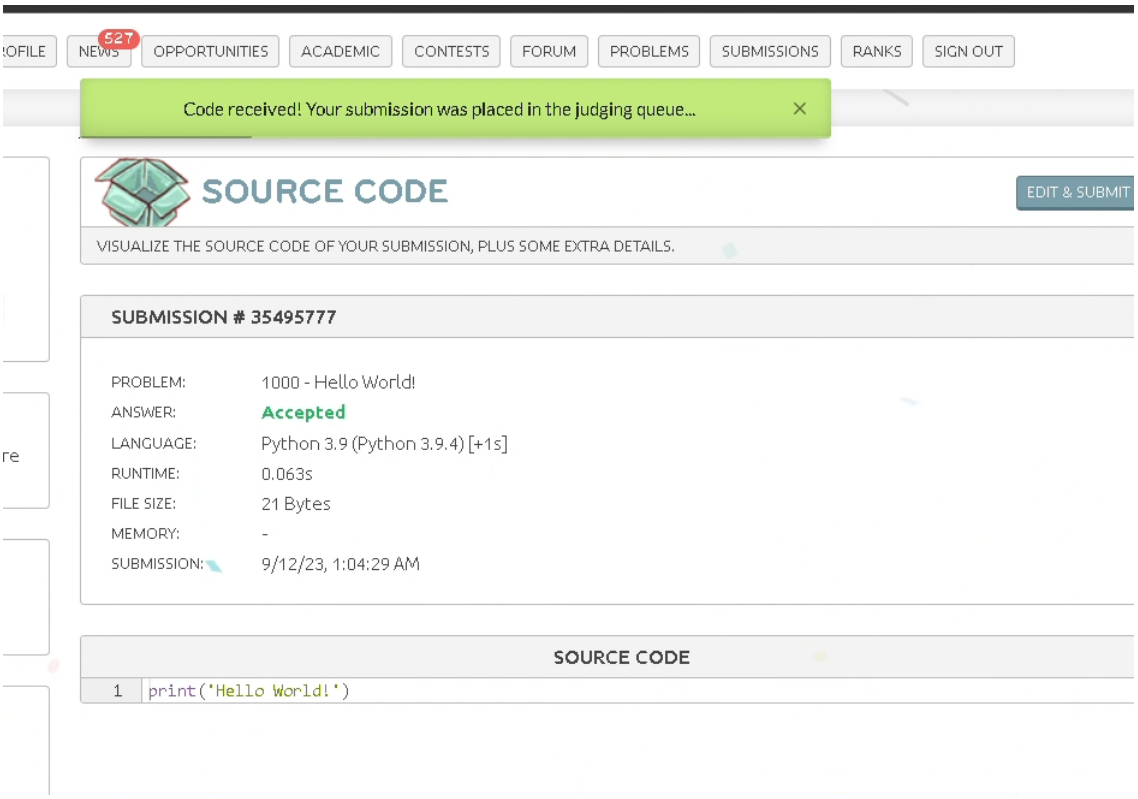
Resolvendo um exercício:



Digita o código e clica em Submit:

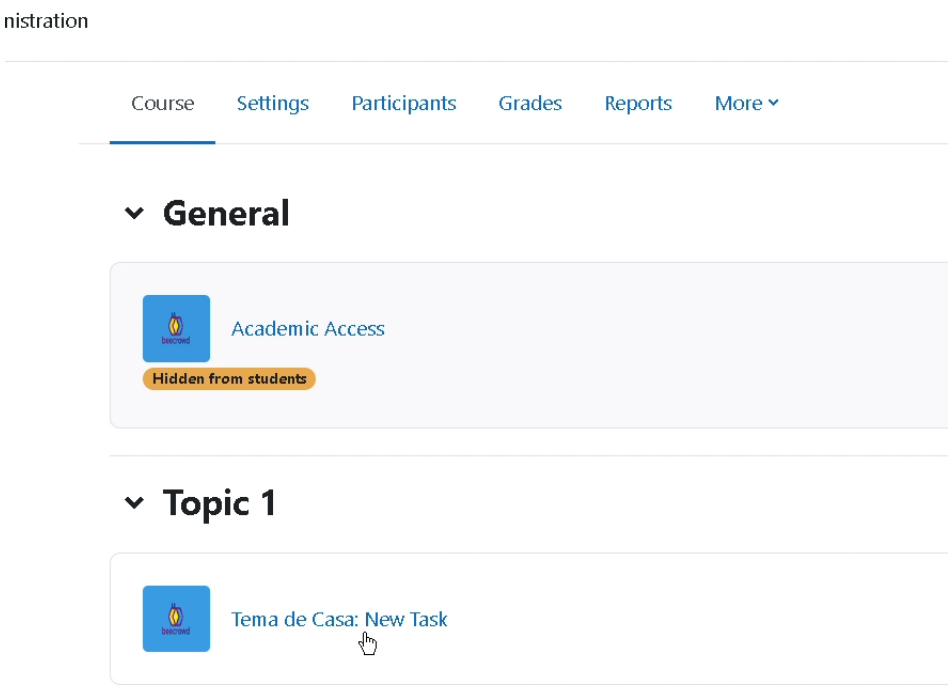


O código é avaliado pelo juíz online da plataforma:



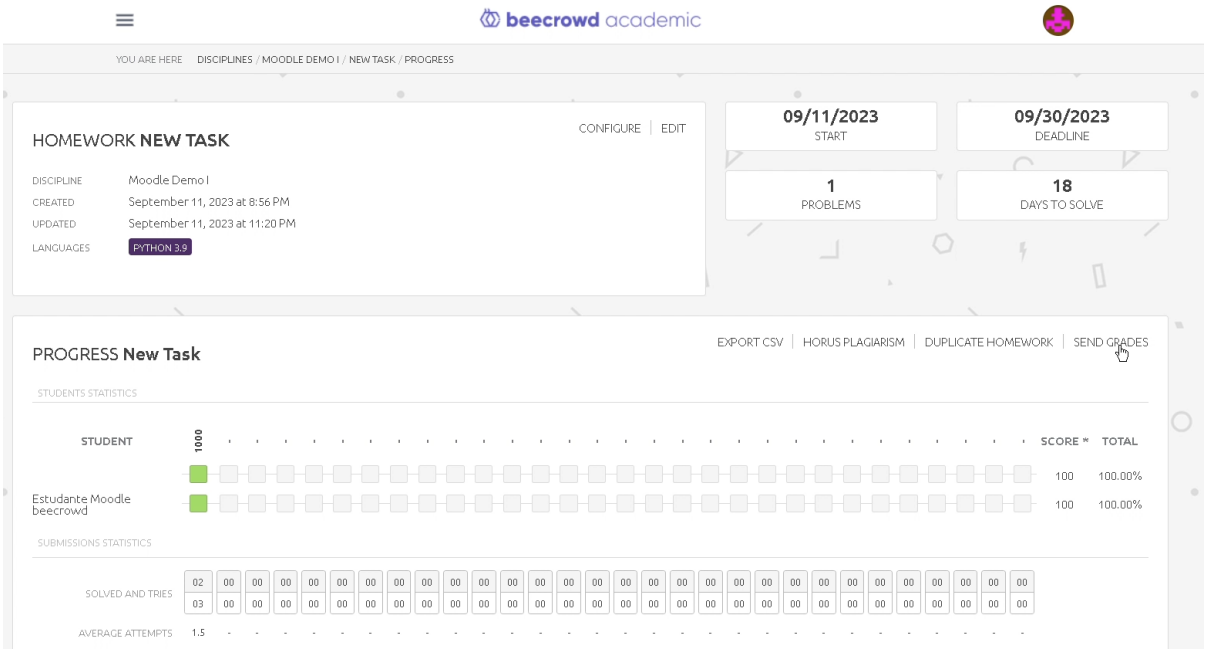
B.1.3 Notas

1. Após seguir os passos anteriores descritos nesse documento, o professor pode clicar na tarefa do Beecrowd que ele criou para os alunos:

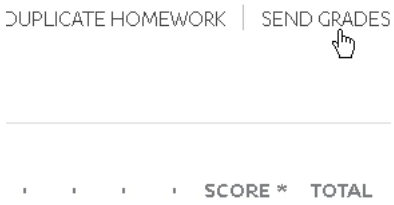


2. O professor pode visualizar quem respondeu as questões, e, ao clicar em **"Send Grades"**, as notas são enviadas para o Moodle, e o professor consegue visualizar as notas lá:

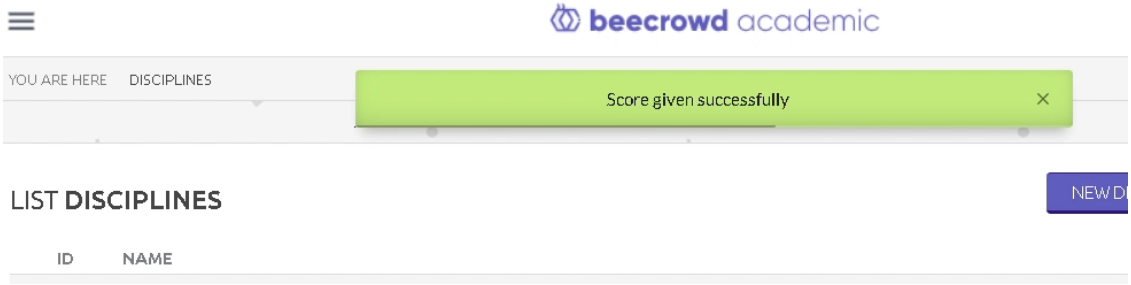
   Página da tarefa no Beecrowd Academic:



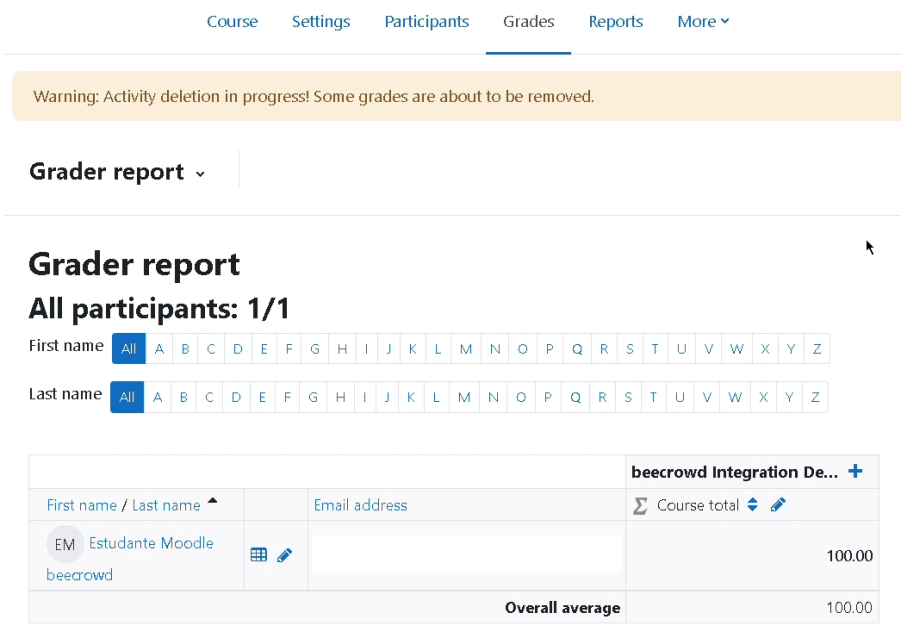
Clica em “Send Grades”:



Mensagem de sucesso que as notas foram enviadas:



Notas agora estão moodle:



APÊNDICE C – COMO ADICIONAR RESPOSTAS PARA NOVAS QUESTÕES NO EXPERT BEE

Basta adicionar um arquivo dentro do diretório `backend`, com o nome no formato `questao_numeroDaQuestao.pl`. Cada arquivo deve seguir o seguinte padrão:

C.1 ESTRUTURA DO ARQUIVO

C.1.1 Definição do módulo

O arquivo deve começar com a declaração do módulo. O nome do módulo deve ser igual ao nome do arquivo (sem a extensão `.pl`). Além disso, o módulo deve exportar dois predicados principais:

- `questao/3`
- `diagnostico/3`

Exemplo:

```
:- module(questao_2465, [questao/3, diagnostico/3]).
```

C.1.2 Predicado `questao/3`

O predicado `questao/3` deve ser usado para mapear uma sequência de respostas a uma nova pergunta. Ele segue o formato:

```
questao(NumeroDaQuestao, RespostasAteAgora, Pergunta).
```

- `NumeroDaQuestao`: Número da questão a que o arquivo se refere.
- `RespostasAteAgora`: Lista de respostas fornecidas pelo usuário até o momento.
- `Pergunta`: Pergunta a ser exibida com base nas respostas recebidas.

Exemplo:

```
questao(2465, [], "Você quer uma dica de qual solução adotar pra esse problema?").
questao(2465, ["sim"], "Você pode usar um while True para verificar, a partir da posição
```

C.1.3 Predicado `diagnostico/3`

O predicado `diagnostico/3` deve ser usado para fornecer um diagnóstico final ou mensagem de feedback com base nas respostas completas do usuário. Ele segue o formato:

```
diagnostico(NumeroDaQuestao, RespostasCompletas, Diagnostico).
```

- `NumeroDaQuestao`: Número da questão.
- `RespostasCompletas`: Lista de respostas fornecidas.
- `Diagnostico`: Diagnóstico ou feedback final.

Exemplo:

```
diagnostico(2465, ["sim", "sim", "sim"], "Legal! Parabéns!").
diagnostico(2465, _, "Atente-se bem às dicas já enviadas ou pergunte novamente!").
```

C.2 CONSIDERAÇÕES IMPORTANTES

- Certifique-se de que todas as perguntas e diagnósticos estejam relacionados ao número da questão especificado.
- O sistema utilizará a lista de respostas fornecida até o momento para determinar qual pergunta exibir em seguida. Portanto, as sequências de respostas devem ser consistentes.
- Para mensagens genéricas ou casos em que as respostas não correspondam a nenhuma sequência específica, use _ como curinga no campo `RespostasCompletas` no predicado `diagnostico/3`.

Com essa estrutura, o sistema poderá processar novas questões automaticamente, bastando que o arquivo correspondente seja adicionado ao diretório.

APÊNDICE D – ARTIGO

Expert Bee: Um Sistema Especialista de Apoio à Aprendizagem Integrada ao Moodle e Beecrowd

Raquel Cristina Schaly Behrens[1], **Maicon Rafael Zatelli**[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)

Abstract. *The use of technology in education has the potential to strengthen the teaching and learning relationship between educators and students, allowing access to materials, timetables and forums through virtual environments. Moodle is an example of a virtual platform that offers pedagogical support to educational institutions, and is widely used among universities. As well as simulating an online classroom, it has tools to support Information Technology and Computing courses, such as VPL. VPL is a Moodle-integrated module and programming task manager, but it has some limitations related to interface and usability. On the other hand, there are platforms not linked to Moodle that have an online judge for programming problems, such as Beecrowd. Beecrowd, by having ready-made programming questions, each with test batteries, which provide real-time feedback, allows users to develop programming and problem-solving skills. In addition, this platform is attracting growing interest in educational environments. This is partly due to its vast collection of questions and also because it is available in Brazilian Portuguese. This makes the platform accessible to a wider audience, especially those who are taking their first steps in the field and are not yet proficient in reading English. Therefore, the aim of this study is to favor the use of the Beecrowd platform among educators, leveraging the existing LTI integration between Moodle and Beecrowd, developed by the platform's team. Additionally, it aims to provide a tool based on an expert system to assist both teachers and students in clarifying recurring doubts about Beecrowd problems, facilitating their adaptation to the platform.*

Resumo. *A utilização da tecnologia na educação tem o potencial de fortalecer a relação de ensino e aprendizagem entre educadores e alunos, permitindo o acesso a materiais, cronogramas e fóruns por meio de ambientes virtuais. O Moodle é um exemplo de plataforma virtual que oferece suporte pedagógico a instituições de ensino, e é amplamente utilizado dentre as universidades. Ele, além de simular uma sala de aula online, possui ferramentas de suporte a cursos da área de Tecnologia da Informação e da Computação, como o VPL. O VPL é um módulo integrado ao Moodle e gerenciador de tarefas de programação, mas que apresenta algumas limitações relacionadas à interface e à usabilidade. Em contrapartida, existem plataformas não vinculadas ao Moodle que possuem um juiz online para problemas de programação, como o Beecrowd. O Beecrowd, ao possuir questões prontas de programação, cada uma com baterias de testes, que fornecem feedback em tempo real, permite aos usuários desenvolver habilidades de programação e resolução de problemas. Além disso, essa plataforma desperta um interesse crescente nos ambientes pedagógicos. Isso ocorre, em parte, devido à sua vasta coleção de questões e também por estar disponível*

em português brasileiro. Isso torna a plataforma acessível a um público mais amplo, especialmente àqueles que estão dando os primeiros passos na área e ainda não possuem proficiência em inglês. Dessa forma, o objetivo deste trabalho consiste em favorecer o uso da plataforma Beecrowd entre docentes, aproveitando a integração LTI já existente entre o Moodle e o Beecrowd, desenvolvida pela equipe da plataforma. Além disso, busca-se disponibilizar uma ferramenta baseada em sistema especialista para auxiliar professores e alunos no esclarecimento de dúvidas recorrentes sobre questões do Beecrowd, facilitando a adaptação de ambos ao uso da plataforma.

1. Introdução

A utilização da tecnologia no âmbito educacional aumenta o potencial de um vínculo proveitoso entre educadores e alunos, possibilitando a disponibilização de materiais, cronogramas, fóruns, entre outras atividades, por ambientes virtuais, e motivando, assim, os educandos [Francisco et al. 2016]. Isto posto, há ambientes virtuais de aprendizado que servem de apoio às instituições de ensino no gerenciamento pedagógico dos cursos, como o software livre Moodle – (*Modular Object-Oriented Dynamic Learning Environment*) – Ambiente de Aprendizagem Dinâmico Orientado a Objetos Modulares. O Moodle disponibiliza diversas ferramentas computacionais que proporcionam acesso a materiais didáticos, tarefas interativas e integração entre os membros do curso em que estão matriculados, reproduzindo, dessa forma, uma sala de aula [Lima 2021].

Por conseguinte, a fim de proporcionar uma maior assistência à cursos da área de tecnologia da informação, algumas iniciativas de integração de recursos de suporte à disciplinas de programação no ambiente Moodle foram elaboradas, como o VPL (*Virtual Programming Lab*) [França and Soares 2011].

O VPL é um módulo integrável ao Moodle que possibilita o desenvolvimento remoto de programas, permitindo realizar as seguintes atividades no navegador: editar o código-fonte dos programas, executá-los interativamente, realizar testes para revisá-los, definir restrições de edição e evitar colagem de texto externo [Rodríguez-Del-Pino 2023].

Contudo, [de Freitas 2016] constata que, em sua pesquisa de avaliação do uso do VPL em disciplinas de programação no curso de graduação de Tecnologias de Informação e Comunicação da UFSC, apesar da ferramenta cumprir seu papel de auxílio virtual à prática de programação, ela possui diversas limitações relacionadas à interface gráfica e à usabilidade, como a falta de mensagens de ajuda para erros, a necessidade de inserção manual de casos de teste e botões pouco intuitivos.

A plataforma Beecrowd destaca-se como uma ferramenta que auxilia alunos na resolução de problemas e desenvolvimento de algoritmos, além de automatizar a correção para educadores e incentivar maratonas de programação [da Cruz et al. 2022]. [Bez and Tonin 2014] ressaltam que sua construção foi voltada às necessidades pedagógicas, oferecendo um módulo acadêmico para acompanhar o desempenho dos alunos e recursos como listas temáticas e gamificação, com *badges* e *rankings*. [Ferreira 2022] enfatiza a base de problemas organizada por categorias e níveis, além de soluções modelo que ajudam na correção, enquanto [Lima 2022] destaca a acessibilidade do Beecrowd com questões em português, tornando-a uma escolha preferida por professores.

Alternativas como Neps Academy e CodeBench oferecem menos recursos, sendo limitadas quanto à gestão de turmas e acesso a problemas, conforme [Lima 2022]. Juízes online como BOCA e UVa apresentam limitações técnicas e operacionais que restringem sua eficiência em sala de aula [Bez and Tonin 2012].

Com sua popularidade crescente nacional e internacionalmente, o Beecrowd atende às demandas de ensino, promovendo aprendizado ativo e acessível. Dessa forma, sua adoção em ambientes acadêmicos é uma estratégia eficaz para aprimorar o ensino de programação e a resolução de problemas [Bez and Tonin 2012], [Ferreira 2022].

1.1. Objetivos

Favorecer o uso da plataforma Beecrowd entre os docentes, para facilitar e aprimorar o ensino da prática de programação. Será aproveitada a integração LTI do Beecrowd com o ambiente Moodle já implementada, a qual facilita o acesso à plataforma e otimiza o aproveitamento de seu extenso banco de questões, oferecendo uma ampla variedade de exercícios de forma mais eficiente. Além disso, será disponibilizada uma ferramenta baseada em sistema especialista para auxiliar tanto professores quanto alunos no esclarecimento de dúvidas recorrentes sobre as questões do Beecrowd, facilitando a adaptação de ambos ao uso da plataforma.

2. Fundamentação Teórica

A fundamentação teórica apresenta os conceitos essenciais para compreender a proposta, destacando tecnologias e ferramentas relacionadas ao tema. Examina suas funcionalidades, limitações e aplicações, além de explorar integrações e interoperabilidade entre sistemas. Por fim, aborda fundamentos técnicos necessários para o entendimento do trabalho.

2.1. Juízes online

Os autores [Wasik et al. 2018] definem os juízes online como "sistemas projetados para a avaliação confiável do código-fonte do algoritmo enviado pelos usuários, que é compilado e testado em seguida em um ambiente homogêneo". Da mesma forma, [Santos and Ribeiro 2011] explicam a principal função dos juízes online: "avaliar códigos fonte que foram enviados em uma determinada linguagem de programação". Essa avaliação automática é feita a partir de casos de teste, ou seja, cada caso de teste possui um conjunto de entradas e saídas, e verifica-se as respostas aos casos de teste da solução do usuário, com as respostas cadastradas para aquela questão no *site* [Francisco et al. 2016].

Diversos juízes online podem ser encontrados na Internet, dentre eles estão o Beecrowd [da Cruz et al. 2022] e o Codebench [Ribeiro et al. 2018], plataformas online com problemas de programação competitiva, e que aceitam soluções para esses desafios, avaliando-as quanto à sua eficácia e eficiência. Também há o BOCA *Online Contest Administrator*, um sistema que visa apoiar as competições de programação na correção de soluções apresentadas pelos competidores [de Campos and Ferreira 2004].

Semelhantemente, [Francisco et al. 2016] ressaltam os benefícios da utilização de um juiz online no ensino de matérias iniciais de programação: "[...] Aprendizagem no ritmo do aluno, auto-aprendizagem e redução da carga de trabalho do professor, são alguns dos benefícios apontados que contribuem não só em ambientes tradicionais de ensino, mas em ambientes de Educação a Distância (EAD) e em MOOC's. A liberdade de

definir listas de exercícios e a disponibilidade de instrumentos para acompanhar os alunos são questões importantes para o professor".

2.2. Beecrowd

O Beecrowd, anteriormente conhecido como URI Online Judge de janeiro de 2013 a outubro de 2021 [Piekarski et al. 2023], é um portal online criado para tornar a prática de programação mais dinâmica, interessante e acessível aos iniciantes na área [Bez and Tonin 2012]. Desenvolvido pela Universidade Regional Integrada (URI) - Campus de Erechim, o portal foi inicialmente concebido com a intenção de oferecer um juiz automático que atendesse às necessidades tanto de professores quanto de alunos, com foco em interatividade, flexibilidade e usabilidade [Bez and Tonin 2012]. O ambiente foi estruturado para ser didaticamente organizado e está disponível 24 horas por dia [Bez and Tonin 2014].

A construção do Beecrowd foi orientada pelas necessidades dos educadores e, principalmente, pelos alunos [Bez and Tonin 2014]. Integrando recursos de renomados portais internacionais de programação, a plataforma oferece funcionalidades exclusivas, como problemas para iniciantes, um sistema de recompensas com *badges*, e um módulo acadêmico completo para o acompanhamento do desempenho dos estudantes, incluindo o *ranking* por universidade [Bez and Tonin 2014].

Além disso, o Beecrowd disponibiliza problemas no estilo do ICPC (International Collegiate Programming Contest), com um juiz online que avalia as submissões dos usuários em tempo real [Berssanette and de Francisco 2018]. Os problemas são organizados por assunto e nível de dificuldade, e há flexibilidade quanto à escolha da linguagem de programação para resolver os exercícios [Bez and Tonin 2012]. Entre seus diferenciais, destacam-se materiais de estudo, tutoriais sobre algoritmos e programação, e um fórum colaborativo [Bez and Tonin 2012].

A plataforma oferece também recursos importantes para professores de Tecnologia da Informação, permitindo a criação de disciplinas com alunos, listas de exercícios organizadas por assunto e prazos, além do acompanhamento da evolução dos estudantes [Bez and Tonin 2014]. Por fim, o Beecrowd suporta diversas linguagens de programação, oferecendo ao usuário a liberdade de escolher a mais adequada para o desenvolvimento das suas soluções [Bez and Tonin 2014].

2.3. BOCA Online Contest Administrator

O BOCA *Online Contest Administrator* é um sistema desenvolvido para gerenciar competições de programação, como a Maratona de Programação da Sociedade Brasileira de Computação [de Campos and Ferreira 2004]. Por ser um juiz online [Bez and Tonin 2012], avalia automaticamente as soluções enviadas pelos participantes, classificando-as como corretas ou incorretas, com feedback imediato para correção [Galasso and Moreira 2014]. Sua interface web permite a organização de competições distribuídas, com equipes localizadas remotamente, e possibilita competições com múltiplas sedes, garantindo a centralização da correção e a igualdade de condições entre os participantes [de Campos and Ferreira 2004].

Embora tenha sido utilizado em sala de aula para apoiar disciplinas de programação, o BOCA não é ideal para esse contexto, devido à necessidade de limpeza

constante do sistema e registro de novos problemas [Bez and Tonin 2012]. Isso se deve ao fato de o sistema não ter sido projetado para esse uso, o que acaba fazendo com que se perca a classificação dos problemas e dos usuários. Apesar disso, é uma excelente opção para competições de programação, incentivando os alunos a estudarem conceitos fundamentais da computação, como estruturas de dados e análise de algoritmos [de Campos and Ferreira 2004].

2.4. VPL

O *Virtual Programming Lab* (VPL) é um módulo do Moodle para gerenciar tarefas de programação, permitindo a edição e execução de código diretamente no navegador. Ele oferece recursos como testes interativos, detecção de plágio e restrições de edição para um ambiente acadêmico controlado [VPL 2021].

O VPL funciona como uma ferramenta de avaliação automática para alunos, e como sistema de gerenciamento para instrutores, facilitando a criação de tarefas, verificação de plágio e condução de avaliações com flexibilidade e segurança [Rodríguez-Del-Pino et al. 2012]. Ele é independente da linguagem de programação e permite a personalização de cada tarefa com informações como prazos, tamanho de arquivos e casos de teste [**?**].

Para criar novas tarefas, o professor deve preencher um formulário de configuração e incluir manualmente os casos de teste. Isso exige a criação de uma nova atividade a cada semestre, garantindo que cada exercício seja adequado às necessidades da turma [**?**].

2.5. CodeRunner

O CodeRunner[1] é um plugin gratuito e de código aberto para o Moodle, projetado para avaliar respostas de programação em várias linguagens. Ele permite que os alunos enviem código, visualizem os resultados dos testes imediatamente e corrijam suas respostas, com avaliação adaptativa baseada nos testes definidos. Ao contrário do VPL, oferece um sistema de penalidades ajustável, permitindo reenvios com penalidades graduais ou até 100% de penalização para tentativas erradas [Lobb and Harlow 2016].

O CodeRunner integra-se facilmente com outros tipos de perguntas do Moodle e é amplamente utilizado em cursos de introdução à programação e em cursos mais avançados de Ciência da Computação, como inteligência artificial e programação na web [Lobb and Harlow 2016]. Assim como no VPL, o criador da tarefa deve configurar os detalhes e casos de teste.

2.6. LTI

LTI *Learning Tools Interoperability* é um padrão que permite a troca segura de dados entre o sistema de gestão de aprendizado (LMS) e ferramentas de aprendizagem externas, centralizando o acesso a conteúdo interno e externo, oferecendo uma experiência de aprendizado consistente, eliminando a necessidade de múltiplos logins para diferentes plataformas [Verdaguer 2021].

A especificação *Learning Tools Interoperability* foi inicialmente introduzida como Basic LTI em maio de 2010 pelo IMS Global Learning Consortium (agora conhecida

[1] `https://coderunner.org.nz/`

como LTI 1.0). Desde então, essa especificação ganhou ampla aceitação como um método simples, mas eficaz, para integrar conteúdos e produtos de terceiros aos Ambientes Virtuais de Aprendizagem (VLEs). Hoje em dia, ela faz parte integral dos principais VLEs, como o Moodle, Blackboard Learn 9, Desire2Learn e o Canvas da Instructure [Vickers and Booth 2014].

2.7. Moodle

O Moodle é uma plataforma de aprendizagem robusta e segura, projetada para educadores, administradores e alunos, oferecendo um sistema integrado para ambientes de aprendizagem personalizados [Moodle 2023]. Como um Sistema de Gerenciamento de Aprendizagem (LMS) altamente extensível, permite acesso remoto por estudantes, tutores e administradores, sendo utilizado por instituições renomadas como o Open Polytechnic da Nova Zelândia [Moodle 2023].

Com foco em segurança, o Moodle oferece controles rigorosos contra acesso não autorizado e perda de dados, podendo ser implantado em nuvem privada ou servidor. Disponível em mais de 60 idiomas, é usado por organizações como London School of Economics e Microsoft, com mais de 213 milhões de usuários [Moodle 2023].

Como software de código aberto, o Moodle oferece ferramentas centradas no aluno, integração com diversas ferramentas colaborativas e escalabilidade para diferentes tamanhos de turmas. Seu design modular permite personalização e adaptação para necessidades específicas, e a comunidade ativa de desenvolvedores garante atualizações constantes e melhorias [Moodle 2023].

Adotando o conceito construcionista social de educação [Galasso and Moreira 2014], o Moodle se destaca como uma escolha popular entre instituições educacionais que buscam modernizar seus sistemas de aprendizagem.

2.7.1. LTI External Tools

As Ferramentas Externas LTI do Moodle permitem integrar aplicativos ao curso, como conteúdos interativos ou avaliações, acessíveis diretamente pelos alunos no Moodle sem necessidade de login em outro sistema. Dependendo da ferramenta, as notas podem ser enviadas de volta ao Moodle [Moodle 2023].

A integração entre Moodle e essas ferramentas é feita pelo padrão Learning Tools Interoperability (LTI), permitindo uma troca segura de informações e criando uma experiência de aprendizado contínua [Moodle 2023].

Administradores podem gerenciar essas ferramentas, adicionando-as aos cursos por meio do botão 'Adicionar ferramenta' e configurando-as nas "Configurações da ferramenta" com informações fornecidas pelo fornecedor, podendo deixar alguns campos em branco se necessário [Moodle 2023].

2.8. Sistema Especialista

De acordo com [Singla 2013], um sistema especialista é um conjunto de programas que manipula conhecimento para resolver problemas em um domínio especializado que exige a *expertise* humana. Já [Bohanec et al. 1990] definem sistemas especialistas como

sistemas inteligentes de informação que imitam, em certo grau, o comportamento de um especialista humano no domínio em questão.

Os principais componentes de um sistema especialista são a base de conhecimento e o motor de inferência. A base de conhecimento armazena informações sobre um domínio específico, geralmente em forma de regras, enquanto o motor de inferência utiliza esse conhecimento para resolver problemas e gerar explicações [Bohanec et al. 1990]. O motor é responsável por gerar conclusões através de raciocínio [Singla 2013].

Embora esses sistemas sejam eficazes, apresentam limitações, como a falta de senso comum, dificuldades em lidar com casos excepcionais e desafios em se adaptar a ambientes em constante mudança [Singla 2013].

2.8.1. Prolog

O Prolog, desenvolvido por Alain Colmerauer e baseado na lógica de predicados de Kowalski, é amplamente utilizado em sistemas especialistas, especialmente em áreas como farmacologia e diagnóstico de falhas [Clark et al. 1980]. Popular para prototipação rápida e análise de linguagem natural, é eficaz em áreas como direito e medicina [Russell and Norvig 2013]. Sua semântica segura e gerenciamento automático de memória o tornam ideal para serviços Web [WIELEMAKER et al. 2008].

2.8.2. SWI-Prolog

O SWI-Prolog, baseado na WAM, é um ambiente de desenvolvimento robusto e versátil, usado principalmente para protótipos de pesquisa e aplicações que requerem integração e lógica sofisticada [WIELEMAKER 2012]. Com bibliotecas que suportam integração com gráficos, bancos de dados e servidores HTTP, é uma excelente ferramenta para o desenvolvimento de sistemas interativos e Web [Wielemaker 2014].

3. Trabalhos Relacionados

Este capítulo apresenta uma revisão de artigos relevantes sobre o uso de plataformas de programação, integração de juízes online com o Moodle e a criação de sistemas especialistas para a web. A seção é estruturada em três partes: um estudo sobre o uso da plataforma Beecrowd em um curso de algoritmos, a análise de ferramentas de integração de juízes online com o Moodle, e a exploração de sistemas especialistas aplicados à educação.

3.1. Uso da Plataforma Beecrowd no Ensino de Algoritmos

O estudo de [da Cruz et al. 2022] investigou o impacto do uso da plataforma Beecrowd em seis semestres da disciplina de algoritmos na Universidade Federal do Maranhão, envolvendo alunos que, em sua maioria, não possuíam conhecimento prévio em programação. A disciplina foi aplicada de forma intercalada: em três semestres com o uso do Beecrowd e em outros três sem a utilização da plataforma. Os resultados demonstraram que, nos semestres em que o Beecrowd foi utilizado, o desempenho dos estudantes foi significativamente superior, sugerindo que o uso de plataformas de programação online pode melhorar os resultados no ensino de algoritmos.

3.2. Integração de Juízes Online com o Moodle

Os autores [Galasso and Moreira 2014], [Lobb and Harlow 2016], e [Rodríguez-Del-Pino et al. 2012] detalharam outras alternativas de juízes online já integrados ao Moodle, como o BOCA, CodeRunner e VPL, respectivamente. mas que possuem desvantagens em relação ao Beecrowd.

Vale destacar que, enquanto o Beecrowd, o CodeRunner e o VPL foram desenvolvidos com foco no apoio ao ensino, o BOCA foi projetado especificamente para avaliar submissões de código em maratonas de programação, embora também possa ser utilizado como ferramenta de suporte no ensino de programação.

Uma característica distintiva do Beecrowd é a disponibilização de questões pré-existentes, eliminando a necessidade de cadastrar manualmente perguntas e testes, algo obrigatório em BOCA, CodeRunner e VPL. Como destacado por [Lobb and Harlow 2016], a criação de questões no CodeRunner é trabalhosa, especialmente em atividades complexas ou turmas grandes, sendo mais apropriado para tarefas simples e bem definidas.

A prevenção contra plágio é um diferencial crucial no Beecrowd e no VPL. Segundo [da Cruz et al. 2022], o Beecrowd emprega análise avançada de código-fonte para identificar padrões suspeitos e similaridades, assegurando a originalidade dos trabalhos. Já o VPL, conforme [Rodríguez-Del-Pino et al. 2012], realiza comparações automáticas de códigos e detecta características indicativas de plágio, preservando a integridade acadêmica.

Outro ponto relevante é que os sistemas BOCA, VPL e CodeRunner adotam servidores independentes para o Moodle e o juiz online, o que assegura maior desempenho, segurança e escalabilidade. Essa arquitetura distribuída facilita a gestão da carga de trabalho, prevenindo sobrecargas e permitindo um dimensionamento mais eficiente de acordo com as demandas específicas de cada sistema. Em relação à criação e execução de código, [Rodríguez-Del-Pino et al. 2012] destacam que o VPL integra um editor Java para a execução e avaliação de códigos, enquanto [Lobb and Harlow 2016] e [Galasso and Moreira 2014] apontam que tanto o CodeRunner quanto o BOCA utilizam caixas de texto para a edição e submissão de código.

Outro trabalho relacionado foi o de [Chaves et al. 2013], onde foi apresentado o Módulo de Integração com Juízes Online (MOJO), ferramenta que integra plataformas com juízes online, como o Beecrowd, ao Moodle, delegando ao Moodle o controle das atividades e centralizando a comunicação com os juízes. Devido à ausência de APIs na época, implementações específicas foram feitas em PHP e JavaScript, com PostgreSQL para armazenamento de dados dos alunos e feedback preciso.

Neste trabalho, contudo, será disponibilizado um manual de uso da integração LTI fornecida pelo Beecrowd, simplificando a interação com a plataforma. Essa abordagem permite que o usuário desenvolva seu código diretamente na plataforma, enquanto ela realiza a correção das resoluções automaticamente.

3.3. Sistemas Especialistas Desenvolvidos para o Ambiente Acadêmico

Por fim, os projetos de [NASR et al. 2018] e [Dunstan 2013] estão relacionados ao desenvolvimento de sistemas especialistas para aconselhamento e planejamento acadêmico,

porém, com abordagens distintas.

[NASR et al. 2018] desenvolveram um sistema baseado em regras para replicar decisões de conselheiros acadêmicos, usando árvores de decisão no ES Builder, com lógica ”se-então” e interface web simples. Já [Dunstan 2013] utiliza XML e Prolog em um planejador acadêmico híbrido, estruturando regras em XML e convertendo-as para Prolog. O sistema distribui funções entre cliente e servidor, permitindo consultas personalizadas e uso em dispositivos móveis.

Este artigo adota uma abordagem similar à de [Dunstan 2013], utilizando a linguagem de programação Prolog para construir o sistema especialista. O Prolog será responsável por gerenciar requisições HTTP, armazenar dados de sessões, possibilitar uma comunicação contínua com o front-end através dessas requisições, permitindo perguntas dinâmicas e respostas personalizadas, além de configurar permissões CORS para integração com a interface web.

4. Manuais de Configuração e Uso da LTI Beecrowd

Com a disponibilização de uma ferramenta LTI pelo Beecrowd, visando facilitar sua integração com o Moodle, foi elaborado um manual detalhado para orientar o administrador do Moodle na configuração dessa LTI.

Para configurar uma LTI externa, o administrador deve acessar ”Administração do *site*”, selecionar ”Plugins” e, em ”Módulos de atividade”, escolher ”Ferramenta externa” e clicar em ”Gerenciar ferramentas”. Em seguida, deve clicar em ”Configurar uma ferramenta manualmente” e preencher os campos conforme as instruções especificadas no Manual de Configuração da LTI Beecrowd no Moodle. Após a configuração, será possível visualizar os detalhes da ferramenta, os quais deverão ser enviados ao Beecrowd para estabelecer a comunicação entre as plataformas.

Além disso, foi desenvolvido um manual de uso da LTI Beecrowd, direcionado aos professores que desejam utilizar o Beecrowd como suporte educacional em sala de aula. Esse manual instrui os docentes sobre como criar uma atividade do Beecrowd no Moodle, orientar o acesso dos estudantes e transferir as notas obtidas no Beecrowd para o Moodle.

Após seguir o guia do Manual de Uso da LTI Beecrowd, duas atividades ficam disponíveis na página do curso, apresentadas como ”botões” visuais. A primeira atividade, que é ocultada para os estudantes, permite ao professor acessar o Beecrowd Academic para configurar disciplinas, criar listas de exercícios para os alunos e enviar as notas para o Moodle. A segunda atividade, acessível aos estudantes, redireciona-os diretamente para a página no Beecrowd da lista de exercícios criada pelo professor, sem necessidade de cadastro ou login. Assim, os estudantes podem resolver as atividades diretamente a partir dessa página. Esse botão também permite ao professor acessar a lista de exercícios criada, podendo verificar o progresso dos alunos, as tentativas realizadas e, quando desejado, enviar as notas para o Moodle.

5. Expert Bee: Sistema Especialista Web

O objetivo do sistema especialista desenvolvido, o Expert Bee, é oferecer suporte na resolução de dúvidas recorrentes dos estudantes em relação a questões do Beecrowd. A

proposta é uma aplicação acessível aos alunos via navegador, com um sistema de *chat* onde o aluno pode informar a questão sobre a qual necessita orientação. O sistema, então, conduz o aluno por meio de perguntas binárias (sim ou não) e, conforme as respostas, fornece dicas relevantes relacionadas ao exercício em questão.

Para viabilizar essa funcionalidade, foi estruturado com um conjunto de dados contendo perguntas direcionadas aos alunos e respostas predefinidas que orientam o aluno na resolução das atividades específicas da disciplina. Em disciplinas como Programação Orientada a Objetos I, onde as listas de exercícios frequentemente incluem questões já utilizadas em semestres anteriores, observa-se uma repetição de problemas e, consequentemente, de dúvidas recorrentes dos alunos.

A primeira etapa deste trabalho foi dedicada à coleta de dados sobre as questões, com o objetivo de identificar e catalogar as dúvidas mais recorrentes. Na segunda etapa, foram definidos os requisitos funcionais, os requisitos não funcionais e as regras de negócio da aplicação. Em seguida, na terceira etapa, realizou-se uma pesquisa sobre as tecnologias que seriam utilizadas no desenvolvimento. Na quarta etapa, foram elaborados os diagramas que definiram a estrutura e o funcionamento da aplicação. Por fim, a partir da quinta etapa, deu-se início ao desenvolvimento propriamente dito.

5.1. Coleta de Dados

Nesta fase, foram realizadas duas coletas de dados para integrar a base de conhecimento do sistema especialista. Na primeira, o professor de Programação Orientada a Objetos I da UFSC forneceu dicas de resolução para duas listas de exercícios do Beecrowd que ainda não haviam sido entregues aos alunos, com o objetivo de avaliar a eficácia da aplicação posteriormente. Na segunda coleta, foram registradas as dúvidas dos alunos enquanto resolviam exercícios sobre o uso de listas em Python, em sala de aula.

5.2. Requisitos da Aplicação

Esta seção descreve os requisitos essenciais para o desenvolvimento da aplicação, divididos em Regras de Negócio, Requisitos Funcionais e Não Funcionais.

5.2.1. Regras de Negócio

As Regras de Negócio definem os processos centrais e orientam o comportamento do sistema. São elas:

- **RN1 – Identificação da Questão**: O usuário deve informar o código da questão do Beecrowd que está tentando resolver para que o sistema possa fornecer dicas específicas.
- **RN2 – Perguntas Binárias**: O sistema deve formular perguntas binárias (sim/não) para guiar o usuário, baseando-se em possíveis erros ou dificuldades comuns para cada questão.
- **RN3 – Personalização de Orientações**: Com base nas respostas do usuário, o sistema deve ajustar as perguntas para fornecer a dica mais apropriada.
- **RN4 – Encerramento do Fluxo**: Ao atingir um número máximo de perguntas, o sistema deve fornecer uma dica final ao usuário.

5.2.2. Requisitos Funcionais

Os Requisitos Funcionais descrevem as funcionalidades que o sistema deve possuir para cumprir os objetivos propostos. São eles:

- **RF1 – Comunicação por Chat**: O usuário deve interagir com o sistema por uma interface de *chat* intuitiva, que apresente respostas sequenciais.
- **RF2 – Busca e Seleção de Questão**: O sistema deve permitir que o usuário informe o código da questão do Beecrowd que deseja resolver.
- **RF3 – Fluxo de Perguntas Binárias**: O sistema apresenta perguntas binárias adaptadas conforme as respostas do usuário.
- **RF4 – Sistema de Dicas**: Após a análise das respostas, o sistema deve oferecer dicas relevantes para o problema informado.
- **RF5 – Instruções sobre Erros Comuns do Beecrowd**: O sistema deve ter uma seção que informa ao usuário descrições dos principais erros retornados pelo Beecrowd, contidos na seção **??**.

5.2.3. Requisitos Não Funcionais

Os Requisitos Não Funcionais especificam características de qualidade e restrições que o sistema deve atender. Eles incluem aspectos como desempenho, usabilidade, segurança e arquitetura. São eles:

- **RNF1 – Desempenho**:
  - O sistema deve responder em tempo real a cada interação do usuário, mantendo a fluidez do *chat*.
- **RNF2 – Usabilidade**:
  - O sistema deve ser intuitivo e fácil de usar, com uma interface de *chat* amigável.
  - Deve ser compatível com os principais navegadores modernos.
- **RNF3 – Escalabilidade**:
  - O sistema deve suportar o uso simultâneo de múltiplos usuários.
- **RNF4 – Segurança**:
  - As sessões de *chat* devem ser isoladas, para evitar vazamento de informações entre os usuários.
- **RNF5 – Manutenibilidade**:
  - O sistema deve ter um código modular, facilitando a inclusão de novas perguntas e dicas para futuras questões do Beecrowd.
- **RNF6 – Arquitetura Backend e Frontend**:
  - O sistema deve ter um *backend* desenvolvido em Prolog, utilizando SWI-Prolog para lidar com requisições HTTP.
  - O *backend* deve definir configurações de CORS apropriadas para permitir a comunicação entre cliente e servidor.
  - O *backend* deve seguir o padrão de API REST e devolver respostas no formato JSON.
  - O *backend* deve conter o sistema especialista, que processa as perguntas e respostas para oferecer suporte aos estudantes.
  - O *frontend* deve ser desenvolvido em React e TypeScript, proporcionando uma interface interativa e amigável para os alunos.

5.3. Tecnologias Usadas

5.3.1. Prolog e SWI-Prolog

Para o desenvolvimento do *backend* da aplicação, utilizou-se a linguagem Prolog, com o auxílio da implementação SWI-Prolog. O SWI-Prolog é uma versão amplamente utilizada do Prolog, que inclui uma série de bibliotecas essenciais para a manipulação de requisições HTTP e dados JSON, como http/thread_httpd, http/http_dispatch, http/json, http/http_session, entre outras. Essas bibliotecas permitem a configuração de um servidor HTTP para o processamento de requisições web e a manipulação de dados JSON. Além disso, elas possibilitam o gerenciamento de sessões, o que viabiliza a integração com aplicações externas e permite o uso simultâneo da aplicação por diferentes usuários [Wielemaker 2024].

5.3.2. API REST

O servidor do backend desse trabalho definiu os *endpoints* HTTP (/server e /questions) para o recebimento de requisições POST e GET, permitindo a interação com o *backend* por meio de uma arquitetura baseada em REST.

O termo REST (Representational State Transfer) é um estilo de comunicação baseado em padrões e protocolos da web, como HTTP, que permite a troca de dados entre sistemas de forma escalável e eficiente. Nesse modelo, as requisições são realizadas utilizando os métodos HTTP padrão (como GET, POST, PUT e DELETE), facilitando a comunicação entre sistemas de maneira eficiente e escalável [Masse 2011].

5.3.3. JSON

JSON (JavaScript Object Notation) é um formato leve e baseado em texto para intercâmbio de dados, independente de linguagem de programação. Derivado do padrão ECMAScript, o JSON define um conjunto simples de regras de formatação para a representação portátil de dados estruturados. Ele busca remover inconsistências com outras especificações e oferece orientações para melhorar a interoperabilidade entre sistemas [Bray 2017].

A biblioteca http/http_json do SWI-Prolog é utilizada para formatar as respostas no formato JSON, possibilitando que o *backend* envie dados estruturados para o *frontend* em um formato amplamente compatível com diversos frameworks web, incluindo o React. Essa abordagem facilita a troca de informações entre o servidor e a interface do usuário, garantindo a interoperabilidade e a eficiência na comunicação entre as camadas da aplicação.

5.3.4. HTTP e CORS

CORS (Cross-Origin Resource Sharing) é um mecanismo que permite que requisições do lado do cliente acessem recursos de origens diferentes. Ele define algoritmos que possibilitam que uma API faça requisições a recursos de outra origem, controlando

o acesso por meio de cabeçalhos de resposta, como o *Access-Control-Allow-Origin* [van Kesteren 2010].

HTTP (Hypertext Transfer Protocol) é um protocolo de aplicação para sistemas distribuídos e colaborativos de informações hipermídia. É um protocolo genérico e sem estado, utilizado em diversas tarefas além do hipertexto, como servidores de nomes e sistemas de gerenciamento de objetos distribuídos. O HTTP permite a negociação e tipificação da representação dos dados, possibilitando a construção de sistemas independentes dos dados transferidos [Fielding 1999].

As bibliotecas http/thread_httpd, http/http_dispatch e http/http_cors, do SWI-Prolog, configuram um servidor HTTP básico, sendo responsáveis pelo tratamento de requisições e respostas. Elas também gerenciam as permissões de CORS, permitindo que o *frontend* (React) acesse o *backend* Prolog, mesmo quando este está hospedado em uma origem diferente.

5.3.5. Sessões HTTP

Uma sessão HTTP é mantida por meio do uso de cookies, que são pequenos arquivos de dados armazenados nos agentes de usuário (como navegadores) pelos servidores HTTP. Os cabeçalhos Cookie e Set-Cookie permitem que os servidores mantenham o estado de uma sessão, mesmo em um protocolo essencialmente sem estado como o HTTP. Isso significa que, através das sessões, os servidores podem armazenar informações sobre o usuário, como preferências ou dados de autenticação, e usá-las em requisições subsequentes, garantindo uma experiência contínua. Embora os cookies tenham questões históricas relacionadas à segurança e privacidade, os cabeçalhos Cookie e Set-Cookie são amplamente utilizados para gerenciar sessões de usuário, permitindo, por exemplo, o login persistente em *sites* ou a manutenção de um carrinho de compras em uma loja online [Barth 2011].

A biblioteca http/http_session do SWI-Prolog possibilita o gerenciamento de sessões de usuário, permitindo o armazenamento de variáveis de sessão, como questionNumber e answers. Esse recurso viabiliza a persistência de informações entre requisições subsequentes de um mesmo usuário, assegurando a continuidade do estado da aplicação ao longo da interação.

5.3.6. React e Typescript (Frontend)

O *frontend* foi feito em React[2] e Typescript[3], interagindo com o *backend* por meio de requisições HTTP, acessando *endpoints* para enviar as respostas do usuário e buscar o resultado final da questão.

React é uma biblioteca para a construção de interfaces de usuário dinâmicas e reutilizáveis. Ela permite criar interfaces a partir de unidades individuais chamadas componentes, que podem ser reutilizados e combinados para formar interfaces complexas e

[2] `https://react.dev/`

[3] `https://www.typescriptlang.org/`

interativas. Cada componente em React gerencia seu próprio estado e pode ser renderizado de forma eficiente conforme as mudanças nos dados, facilitando o desenvolvimento de aplicações dinâmicas e de fácil manutenção. React pode ser utilizado com outras linguagens que compilam para JavaScript[4], como TypeScript.

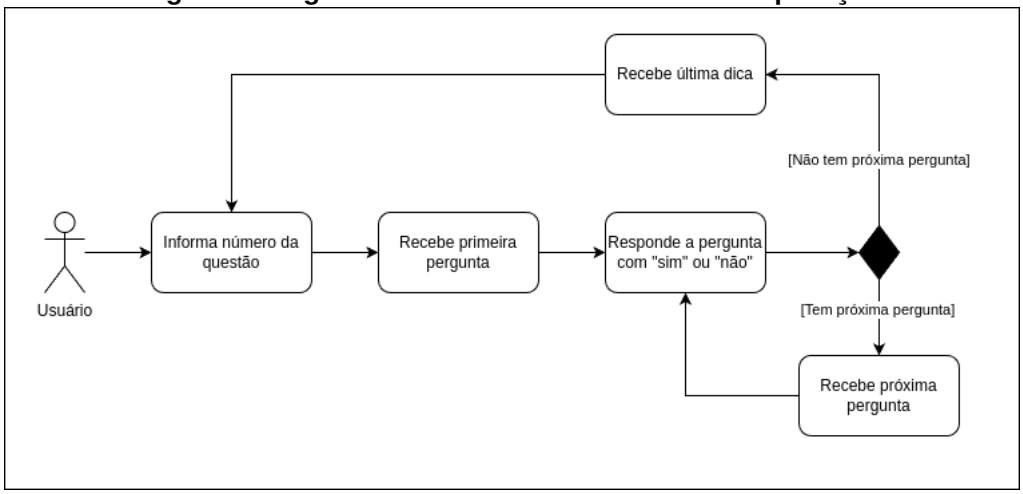
TypeScript é uma linguagem de programação que é um superconjunto do JavaScript, adicionando tipagem estática. Isso permite que os desenvolvedores definam tipos para variáveis e funções, ajudando a identificar erros antes da execução do código. TypeScript melhora a manutenção e segurança do código, especialmente em projetos grandes, e é amplamente utilizado com frameworks como React para criar aplicações mais escaláveis e robustas.

5.4. Arquitetura do Software

A aplicação propõe uma interação dinâmica em que o usuário, ao informar o número da questão sobre a qual deseja obter orientações, inicia um diálogo com o sistema de *chat*. O *chat* responde com perguntas específicas para compreender melhor as necessidades do usuário, permitindo direcionar as orientações de forma mais precisa. Conforme o usuário responde com ”sim” ou ”não” a cada pergunta, o sistema adapta as próximas questões e sugestões, guiando-o progressivamente até fornecer uma orientação final ajustada ao contexto das respostas acumuladas.

A Figura 1 apresenta o diagrama de atividades que detalha o fluxo de interação entre o usuário e o *chat* da aplicação. Após informar o número da questão sobre a qual deseja obter orientações, o usuário recebe uma pergunta inicial, respondendo com ”sim” ou ”não”. Se houver mais perguntas, o *chat* continua a apresentá-las até que todas sejam respondidas, conduzindo o usuário a uma orientação final baseada nas respostas acumuladas. Ao final, o usuário pode iniciar o processo novamente com outro número de questão ou com o mesmo, caso deseje explorar outras sugestões e perguntas relacionadas.

Figure 1. Diagrama de Atividades do Usuário da Aplicação

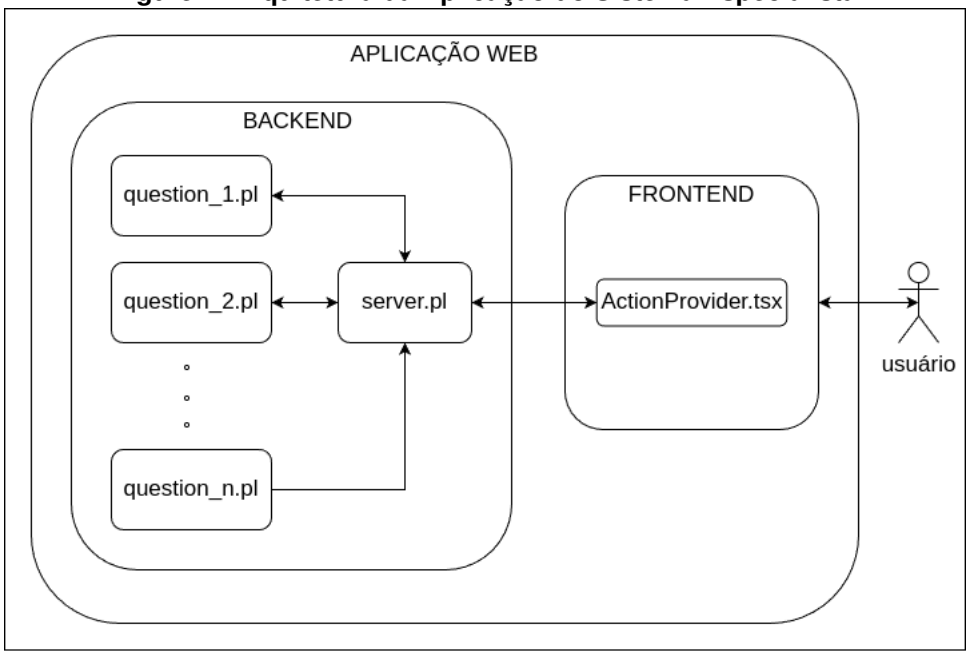


A Figura 2 representa a estrutura da aplicação, mostrando a interação entre o *frontend* e o *backend* através de requisições. No *frontend*, o arquivo *ActionProvider.tsx* gerencia as mensagens enviadas pelo usuário no *chat*, processando tanto o número da questão

[4]https://www.javascript.com/

informada, quanto as respostas de ”sim” ou ”não”. Por meio dessas requisições, o *frontend* envia esses dados ao *backend*, que, em resposta, fornece perguntas e dicas relacionadas à questão solicitada. Caso o usuário peça uma nova dica e o *backend* não tenha mais sugestões para aquela questão, ele envia a orientação final da questão, que é então apresentada ao usuário.

Figure 2. Arquitetura da Aplicação do Sistema Especialista



O *frontend* da aplicação foi desenvolvido utilizando as tecnologias React e TypeScript, conforme especificado no requisito não funcional **RNF6**, descrito na seção 5.2. Essas tecnologias foram escolhidas por sua capacidade de criar interfaces dinâmicas, escaláveis e de fácil manutenção, proporcionando uma experiência de usuário fluida e interativa **(RNF2)**.

Conforme o requisito funcional **RF1** e a regra de negócio **RN1**, foi implementado um sistema de *chat* como interface principal de interação com o usuário. Por meio deste *chat*, o usuário pode informar o código da questão do Beecrowd que está tentando resolver **(RF2)**. Além disso, conforme a regra de negócio **RN2**, o sistema especialista se comunica com o usuário por meio do *chat*, exibindo mensagens sequenciais baseadas nas respostas do sistema **(RF3)** em tempo real **(RNF1)**, e dicas finais **(RF4)**. O *chat* também permite que o usuário forneça respostas binárias (sim/não), conforme o fluxo de perguntas definido para personalizar a orientação fornecida, conforme as regras de negócios **RN2** e **RN3**.

O desenvolvimento do *frontend* foi baseado em bibliotecas e repositórios existentes que facilitaram a criação e configuração do sistema de *chatbot*. Especificamente, foi utilizado o repositório *react-chatbot-kit* de FredrikOseberg[5] como estrutura principal para a criação do *chatbot*, combinado com o repositório *Chatbot-using-React-Chatbot-Kit* de Subid Das[6], que forneceu recursos adicionais para personalização da interface e

[5]`https://github.com/FredrikOseberg/react-chatbot-kit`
[6]`https://github.com/devsubid/Chatbot-using-React-Chatbot-Kit`

integração do *chatbot* com o fluxo de mensagens interativo. Esses repositórios foram adaptados para atender às necessidades específicas da aplicação, garantindo que o *frontend* fosse capaz de processar interações dinâmicas e personalizadas, conforme os requisitos estabelecidos.

Para atender ao Requisito Funcional **RF5**, descrito na seção 5.2, foi criado um componente *Details* para a rota *details*, a fim de mostrar as descrições dos possíveis resultados do juíz online do Beecrowd.

O *backend* foi desenvolvido em Prolog, utilizando o SWI-Prolog para possibilitar o tratamento de requisições HTTP, conforme especificado no Requisito Não Funcional **RNF6**, na seção 5.2.

Além disso, conforme ilustrado na Figura 2, o *backend* é composto pelo arquivo principal *server.pl* e por arquivos adicionais, cada um representando um sistema especialista específico para cada questão, seguindo, também, o Requisito Não Funcional **RNF6**. Esses arquivos são nomeados no formato *questao_*{*numeroDaQuestao*}*.pl* e contêm dicas específicas para as respectivas questões. Por exemplo, o arquivo *questao_1234.pl* armazena as orientações para a questão 1234 do Beecrowd.

Dessa forma, arquivos do formato *questao_*{*numeroDaQuestao*}*.pl* devem a seguinte estrutura: conter o predicado *questao/3* para fornecer orientações sequenciais, e o predicado *diagnostico/3*, para mensagem final.

Além disso, o arquivo *server.pl* configura e inicializa um servidor HTTP, definindo rotas para processar requisições e habilitando o CORS, permitindo conexões da porta do *frontend*. Ele recebe requisições *POST* para iniciar ou continuar uma questão, associando cada sessão ao usuário correspondente, e também aceita requisições *GET* para fornecer o diagnóstico final com base nas respostas acumuladas durante a interação.

Os arquivos de questões são carregados dinamicamente no *server.pl* com base no número da questão recebido nas requisições *POST*. Se o usuário alterna para uma nova questão, por exemplo, a questão Y, o *backend* carrega o arquivo *questao_Y.pl* e passa a conduzir a interação de acordo com suas informações.

O arquivo *server.pl* também armazena na sessão do usuário as informações fornecidas, como número da questão e respostas, para garantir a continuidade da interação.

6. Manuais de Como Adicionar Novas Questões no Expert Bee

Foi criado um manual que auxilia o professor, passo a passo, em como adicionar dicas para uma nova questão no Bee Expert. Este manual proporciona uma abordagem prática para garantir que as questões sejam adicionadas de forma correta e eficiente, facilitando a inclusão de novas questões e o fornecimento de orientações adequadas aos usuários do sistema.

Para adicionar novas questões ao sistema, deve-se criar um arquivo Prolog no diretório `/backend`, nomeado conforme o formato `questao_numeroDaQuestao.pl`. Cada arquivo deve seguir uma estrutura padrão contendo dois predicados principais: `questao/3` e `diagnostico/3`.

O predicado `questao/3` associa uma sequência de respostas do usuário a uma

nova pergunta, enquanto o predicado `diagnostico/3` fornece um diagnóstico ou *feedback* final com base nas respostas completas. O sistema utiliza essas definições para iterar sobre as perguntas e fornecer respostas progressivas ou um diagnóstico final, de acordo com as respostas fornecidas até o momento. A consistência das respostas e do número da questão é essencial para o funcionamento adequado do sistema.

Ao adicionar uma nova questão, é fundamental que o arquivo siga corretamente essa estrutura, permitindo que o sistema integre automaticamente a questão ao fluxo de interações. As respostas parciais devem ser tratadas de forma coerente, e qualquer sequência de respostas não mapeada deve ser considerada com um curinga no predicado `diagnostico/3`.

7. Validação

Nesta seção, será feita uma análise crítica da integração entre o Moodle e o Beecrowd via LTI (*Learning Tools Interoperability*) e da aplicação com sistema especialista desenvolvida. Também será apresentada uma avaliação prática do uso da aplicação pelos alunos da disciplina *Programação Orientada a Objetos I* da UFSC, no segundo semestre de 2024, para verificar a efetividade do sistema na resolução das dúvidas dos estudantes.

7.1. Integração LTI entre o Moodle e o Beecrowd

7.1.1. Vantagens

- **Vantagens do Beecrowd:** O Beecrowd oferece uma plataforma robusta para a resolução de problemas de programação, com uma grande variedade de questões que permitem aos alunos praticar e desenvolver habilidades de programação em diversos níveis de dificuldade. A plataforma também facilita o acompanhamento do progresso dos alunos, oferecendo métricas e *feedback* instantâneo sobre o desempenho nas questões.
- **Facilidade de acesso através da integração LTI:** A integração LTI simplifica o acesso ao Beecrowd, pois tanto o professor quanto o aluno podem acessar a plataforma diretamente do Moodle com um único clique.

7.1.2. Desvantagens

- **Desenvolvimento externo e curva de aprendizado:** O desenvolvimento dos códigos e a criação das listas de exercícios ainda ocorre fora do Moodle, no Beecrowd, o que significa que o professor precisa aprender a usar o Beecrowd Academic, embora o processo seja relativamente simples. Isso pode exigir um tempo adicional de adaptação, especialmente para aqueles que não estão familiarizados com a plataforma.

7.2. Sistema Especialista para Resolução de Dúvidas no Beecrowd

7.2.1. Vantagens

- **Atendimento automatizado:** O sistema especialista pode fornecer respostas rápidas e precisas para questões frequentes ou comuns, permitindo que os alunos resolvam suas dúvidas sem a necessidade de intervenção humana constante.

- **Apoio no aprendizado:** O sistema pode ser configurado para fornecer explicações detalhadas sobre questões específicas do Beecrowd, ajudando os alunos a entender melhor os problemas e as soluções.
- **Escalabilidade:** Como o sistema é automatizado, ele pode lidar com um grande volume de dúvidas de maneira eficiente.
- **Redução da sobrecarga para os professores:** Ao automatizar o atendimento às dúvidas mais comuns, o sistema especialista permite que os professores se concentrem em questões mais complexas e em atividades pedagógicas, sem se sobrecarregar com um grande volume de perguntas repetitivas.
- **Auxílio para professores não familiarizados com o Beecrowd:** Para professores que ainda não estão familiarizados com as questões do Beecrowd, o sistema especialista oferece uma ferramenta valiosa para auxiliá-los a responder dúvidas dos alunos de maneira eficiente, sem a necessidade de um profundo conhecimento prévio sobre a plataforma ou as questões.

7.2.2. Desvantagens

- **Limitação nas respostas para questões complexas:** O sistema especialista é eficiente para lidar com questões simples ou frequentes, mas pode encontrar dificuldades ao lidar com questões mais complexas ou específicas. Para adicionar suporte a novas questões, o professor precisa criar uma matriz de decisões detalhada, o que pode tornar o processo demorado e propenso a erros, especialmente quando se trata de questões que exigem uma análise mais aprofundada. Sem a devida cautela, o sistema pode acabar gerando respostas imprecisas ou inadequadas, comprometendo a qualidade das interações e a efetividade do suporte oferecido.
- **Dependência da qualidade da base de conhecimento:** O desempenho do sistema especialista depende diretamente da qualidade das regras e conhecimentos que foram programados nele. Se a base de dados for limitada ou desatualizada, o sistema pode fornecer respostas inadequadas.
- **Falta de interação humana:** Em casos onde o aluno necessita de uma explicação mais personalizada ou de orientação emocional, o sistema especialista pode não ser suficiente, criando uma experiência impessoal que pode não atender completamente às necessidades do aluno.

7.3. Avaliação Prática

A avaliação prática foi realizada durante o segundo semestre de 2024, com os alunos da disciplina *Programação Orientada a Objetos I* da UFSC, que utilizaram o sistema especialista Expert Bee enquanto resolviam a lista de exercícios do Beecrowd. À medida que surgiam dúvidas, os alunos foram incentivados a usar a ferramenta para buscar soluções. O uso do Expert Bee foi monitorado para avaliar sua eficácia na resolução das dúvidas. Com base na análise dos resultados, as dúvidas não resolvidas pelo sistema foram incorporadas ao banco de conhecimento para futuras turmas.

É importante destacar que, até o momento, o Expert Bee tinha apenas dicas de resoluções para essas questões da lista de exercícios fornecida, e não dicas para dúvidas reais coletadas de alunos. Essas dicas de resoluções tinham sido disponibilizadas pelo

professor durante a primeira coleta de dados, descrita na seção **??**, para que fosse possível testar a eficácia da ferramenta na hora em que as dúvidas dos alunos surgiriam.

Assim, um dos objetivos da avaliação prática também era identificar dúvidas não tratadas pelo Expert Bee, para que elas pudessem ser adicionadas à ferramenta.

7.3.1. Resultados

Na Avaliação Prática realizada em 18 de novembro de 2024, 12 alunos estiveram presente na sala de aula e, sempre que um aluno apresentava uma dúvida, ele utilizava primeiro o Expert Bee para tentar resolvê-la, e, caso não conseguisse, recorria ao professor.

No total, de 11 dúvidas relatadas, o Bee Expert conseguiu resolver 9, resultando em uma taxa de sucesso de aproximadamente 81%. Além disso, 4 pontos de melhoria foram coletados.

8. Conclusão

Em conclusão, este trabalho abordou a relevância da integração de ferramentas tecnológicas no ensino de programação, com foco especial no uso de plataformas como Beecrowd e Moodle. A análise dos juízes online e outras plataformas educacionais, como o VPL, evidenciou desafios na usabilidade e na integração, mas também destacou o Beecrowd como uma solução eficaz para o ensino de algoritmos e programação, com recursos que incentivam o aprendizado ativo, como gamificação, *feedback* em tempo real e uma gama diversificada de questões.

Além disso, este estudo aprofundou o impacto dos juízes online no ensino de programação, enfatizando a relevância da integração entre plataformas como o Beecrowd e o Moodle. A integração do Beecrowd ao Moodle, por meio da tecnologia LTI, facilita o acesso de professores e alunos a uma vasta gama de questões, otimizando o uso da plataforma no ambiente acadêmico. Além disso, oferece uma interface amigável para a criação de listas de exercícios pelos professores e a resolução de problemas de programação pelos alunos. Com o objetivo de apoiar essa integração, foi desenvolvido um manual detalhado sobre como integrar o Moodle ao Beecrowd via LTI, bem como um guia para orientar os professores no uso da plataforma.

Para estimular o uso do Beecrowd no ambiente acadêmico, também foi criada uma ferramenta com um sistema especialista que auxilia os alunos ao esclarecer dúvidas recorrentes nas questões do Beecrowd. Esse sistema, testado em sala de aula, não só oferece suporte aos estudantes, mas também facilita a adaptação dos professores, promovendo um ambiente de aprendizado mais fluido e eficiente para todos os envolvidos.

8.1. Trabalhos Futuros

A seção a seguir apresenta algumas sugestões de melhorias e direções futuras para a aplicação de sistema especialista desenvolvida, a Expert Bee.

- **Adição de dicas de novas questões pela interface web:** Uma melhoria importante seria adaptar o sistema para permitir que os professores possam adicionar dicas para novas questões diretamente pela interface da aplicação web, sem a necessidade de modificar o código-fonte. Essa melhoria tornaria o sistema mais

acessível e eficiente, permitindo que os educadores personalizem o conteúdo de maneira prática e intuitiva, sem a dependência de habilidades técnicas.

- **Sistema autoalimentado (aprendizado com as interações dos alunos):** Uma possível evolução seria adaptar o sistema para se autoalimentar com base nas interações dos alunos. O sistema poderia aprender com as dúvidas mais frequentes e se ajustar automaticamente, oferecendo respostas e dicas mais precisas com o tempo. Essa melhoria contribuiria para uma experiência de aprendizagem mais personalizada, permitindo que os alunos superem suas dificuldades de forma mais eficiente.
- **Integração com ferramentas de processamento de linguagem natural (ChatGPT):** Uma adaptação interessante seria integrar o sistema especialista com ferramentas de processamento de linguagem natural, como o ChatGPT. Isso permitiria uma interação mais fluida e natural entre os alunos e o sistema, além de melhorar a precisão, contextualização e dinamismo das respostas. Com essa integração, seria possível oferecer um nível maior de personalização nas interações, aprimorando a qualidade do suporte oferecido.
- **Adição de dicas de novas questões de outras plataformas:** O Expert Bee também pode ser utilizado para fornecer dicas de questões de outras plataformas além do Beecrowd. Um possível aprimoramento seria adaptar o Expert Bee para identificar de qual plataforma as questões estão vindo e oferecer dicas específicas para essas questões. Dessa forma, seria possível ampliar a utilidade da ferramenta para diferentes plataformas de ensino, tornando-a mais flexível e adaptada a diferentes contextos de aprendizagem.

References

Barth, A. (2011). Http state management mechanism.

Berssanette, J. H. and de Francisco, A. C. (2018). Uma proposta de ensino de programação de computadores com base na pbl utilizando o portal uri online judge. In *II SIMPÓSIO IBERO-AMERICANO DE TECNOLOGIAS EDUCACIONAIS*, pages 348–354, Araranguá.

Bez, J. L. and Tonin, N. A. (2012). Uri online judge: A new classroom tool for interactive learning. In *The Steering Committee Of The World Congress In Computer Science, Computer Engineering And Applied Computing (Worldcomp)*, pages 1–5, Athens.

Bez, J. L. and Tonin, N. A. (2014). Uri online judge e a internacionalização da universidade. *Revista Eletrônica de Extensão da Uri*, 10(18):237–249.

Bohanec, M., Bratko, I., and Rajkovic, V. (1990). Expert system for decision making. *Sistemica*, 1(1):145–157.

Bray, T. (2017). The javascript object notation (json) data interchange format.

Chaves, J. O., Castro, A., Lima, R., Lima, M. V., and Ferreira, K. H. A. (2013). Uma ferramenta baseada em juízes online para apoio às atividades de programação de computadores no moodle. *Revista Novas Tecnologias na Educação*, 11(3).

Clark, K. L., McCabe, F., and McCabe, F. G. (1980). Prolog: a language for implementing expert systems. Master's thesis, Imperial College of Science and Technology. Department of Computing.

da Cruz, A. K. B. S., de Salles Soares Neto, C., da Cruz, P. T. M. B., and Teixeira, M. A. M. (2022). Utilização da plataforma beecrowd de maratona de programação como estratégia para o ensino de algoritmos. In *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital (Sbgames Estendido 2022)*, pages 754–764. Sociedade Brasileira de Computação.

de Campos, C. P. and Ferreira, C. E. (2004). Boca: um sistema de apoio a competições de programação. In *Sociedade Brasileira de Computação*, São Paulo.

de Freitas, L. M. (2016). Análise de usabilidade do módulo laboratório virtual de programação do moodle. Master's thesis, Universidade Federal de Santa Catarina, Araranguá.

Dunstan, N. (2013). An interactive webbased expert system degree planner. *The Second International Conference on Informatics Engineering Information Science (ICIEIS2013). The Society of Digital Information and Wireless Communication*, pages 302–308.

Ferreira, R. M. G. (2022). Broms: Brazilian online marathon scoreboard. Master's thesis, Faculdade Unb Gama, Universidade de Brasília, Brasília.

Fielding, R. (1999). Hypertext transfer protocol – http/1.1.

Francisco, R., Júnior, C. P., and Ambrósio, A. P. (2016). Juiz online no ensino de programação introdutória - uma revisao sistemática da literatura. In *Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE 2016)*, pages 11–20.

França, A. B. and Soares, J. M. (2011). Sistema de apoio a atividades de laboratório de programação via moodle com suporte ao balanceamento de carga. In *Anais do XXII SBIE - XVII WIE*, pages 710–719. SBIE.

Galasso, R. H. and Moreira, B. G. (2014). Integração do ambiente boca com o ambiente moodle para avaliação automática de algoritmos. In *COMPUTER ON THE BEACH*, pages 22–31, Ponta Grossa.

Lima, G. M. (2022). Gamma online judge: projeto de criação de uma plataforma para o armazenamento das questões das maratonas unb de programação. Master's thesis, Faculdade Unb Gama, Universidade de Brasília, Brasília.

Lima, J. M. M. (2021). Plataforma moodle: a educação por mediação tecnológica. *Revista Científica Multidisciplinar Núcleo Do Conhecimento*, 7(6):17–37.

Lobb, R. and Harlow, J. (2016). Coderunner: a tool for assessing computer programming skills. In *ACM Inroads*, volume 7, pages 47–51, Ponta Grossa.

Masse, M. (2011). Rest api design rulebook.

Moodle (2023). Moodle developer resource centre.

NASR, O. A., TALAB, A., and AL-GAHTANI, S. S. (2018). Building of a rule-based expert system for academic advising via web expert system tools. Master's thesis, FKing Khalid University - College of Business - Dept. of MIS.

Piekarski, A. E. T., Miazaki, M., da Rocha Junior, A. L., Militão, E. P., and da Silva, J. V. P. (2023). ProgramaÇÃo competitiva em um projeto de extensÃo para o ensino tÉcnico em informÁtica. *Revista Conexão UEPG*, 19(1):1–15.

Ribeiro, R. B., Fernandes, D., de Carvalho, L. S. G., and Oliveira, E. (2018). Gamificação de um sistema de juiz online para motivar alunos em disciplina de programação introdutória. In *Anais do XXIX Simpósio Brasileiro de Informática na Educação (SBIE 2018)*, pages 805–814.

Rodríguez-Del-Pino, J. C. (2023). Moodle plugins directory: Virtual programming lab.

Rodríguez-Del-Pino, J. C., Royo, E. R., and Figueroa, Z. H. (2012). A virtual programming lab for moodle with automatic assessment and anti-plagiarism features. In *THE 2012 INTERNATIONAL CONFERENCE ON E-LEARNING, E-BUSINESS, ENTERPRISE INFORMATION SYSTEMS, E-GOVERNMENT*, Hong Kong.

Russell, S. and Norvig, P. (2013). Inteligência artificial: tradução da terceira edição. Elsevier.

Santos, J. C. S. and Ribeiro, A. R. L. (2011). Jonline: proposta preliminar de um juiz online didático para o ensino de programação. In *Anais do Simpósio Brasileiro de Informática na Educação (SBIE 2011)*, pages 964–967.

Singla, J. (2013). The diagnosis of some lung diseases in a prolog expert system. *International Journal of Computer Applications*, 78(15):37–40.

van Kesteren, A. (2010). Cross-origin resource sharing, w3c working draft wd-cors-20100727.

Verdaguer, J. (2021). O que é lti e como ele pode melhorar seu ecossistema de aprendizagem.

Vickers, S. P. and Booth, S. (2014). Creating environments for learning through instigating a community of developers: A jisc-funded project. *Learning Tools Interoperability (LTI): A Best Practice Guide*.

VPL (2021). Virtual programming lab for moodle (vpl).

Wasik, S., Antczak, M., Badura, J., Laskowski, A., and Sternal, T. (2018). A survey on online judge systems and their applications. *Acm Computing Surveys*, 51(1):1–34.

WIELEMAKER, J. (2012). In *SWI-Prolog 6.1: Reference Manual*.

Wielemaker, J. (2014). Swi-prolog version 7 extensions. In *Workshop on Implementation of Constraint and Logic Programming Systems and Logic-based Methods in Programming Environments*.

Wielemaker, J. (2024). Swi-prolog http support.

WIELEMAKER, J., HUANG, Z., and MEIJ, L. V. D. (2008). Swi-prolog and the web. In *Theory and Practice of Logic Programming*, pages 363–392.

APÊNDICE E – CÓDIGO FONTE

docker-compose.yml

```
version: "3.9"

services:
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    container_name: backend-container
    networks:
      - app-network
    volumes:
      - ./backend:/app
    environment:
      VITE_FRONTEND_URL: "http://localhost"

  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    container_name: frontend-container
    ports:
      - "5173:5173"
    networks:
      - app-network
    depends_on:
      - backend
    environment:
      VITE_BACKEND_URL: "http://localhost"

  nginx:
    build:
      context: ./nginx
      dockerfile: Dockerfile
    image: nginx:alpine
    container_name: nginx-container
    ports:
```

```
      - "80:80"
    networks:
      - app-network
    depends_on:
      - frontend
      - backend

networks:
  app-network:
    driver: bridge
```

manual_adicionar_questoes.md

```
# Como adicionar respostas para novas questões

Basta adicionar um arquivo dentro do diretório `/backend`, com o nome no
↪  formato `questao_{numeroDaQuestao}.pl`. Cada arquivo deve seguir o
↪  seguinte padrão:

## Estrutura do arquivo

### 1. Definição do módulo

O arquivo deve começar com a declaração do módulo. O nome do módulo deve ser
↪  igual ao nome do arquivo (sem a extensão `.pl`). Além disso, o módulo
↪  deve exportar dois predicados principais:

- `questao/3`
- `diagnostico/3`

Exemplo:

```prolog
:- module(questao_2465, [questao/3, diagnostico/3]).
```

2. Predicado `questao/3`

O predicado `questao/3` deve ser usado para mapear uma sequência de
↪ respostas a uma nova pergunta. Ele segue o formato:
```

```prolog
questao(NumeroDaQuestao, RespostasAteAgora, Pergunta).
```

- **`NumeroDaQuestao`**: Número da questão a que o arquivo se refere.
- **`RespostasAteAgora`**: Lista de respostas fornecidas pelo usuário até o momento.
- **`Pergunta`**: Pergunta a ser exibida com base nas respostas recebidas.

Exemplo:

```prolog
questao(2465, [], "Você quer uma dica de qual solução adotar pra esse problema?").
questao(2465, ["sim"], "Você pode usar um while True para verificar, a partir da posição (a, b) na matriz, qual vizinho...").
```

3. Predicado `diagnostico/3`

O predicado `diagnostico/3` deve ser usado para fornecer um diagnóstico final ou mensagem de feedback com base nas respostas completas do usuário. Ele segue o formato:

```prolog
diagnostico(NumeroDaQuestao, RespostasCompletas, Diagnostico).
```

- **`NumeroDaQuestao`**: Número da questão.
- **`RespostasCompletas`**: Lista de respostas fornecidas.
- **`Diagnostico`**: Diagnóstico ou feedback final.

Exemplo:

```prolog
diagnostico(2465, ["sim", "sim", "sim"], "Legal! Parabéns!").
diagnostico(2465, _, "Atente-se bem às dicas já enviadas ou pergunte novamente!").
```

```
```

Considerações importantes

- Certifique-se de que todas as perguntas e diagnósticos estejam
↪ relacionados ao número da questão especificado.
- O sistema utilizará a lista de respostas fornecida até o momento para
↪ determinar qual pergunta exibir em seguida. Portanto, as sequências de
↪ respostas devem ser consistentes.
- Para mensagens genéricas ou casos em que as respostas não correspondam a
↪ nenhuma sequência específica, use `_` como curinga no campo
↪ `RespostasCompletas` no predicado `diagnostico/3`.

Com essa estrutura, o sistema poderá processar novas questões
↪ automaticamente, bastando que o arquivo correspondente seja adicionado
↪ ao diretório.
```

### README.md

```
Expert Bee

O frontend desta aplicação foi desenvolvido com base nos projetos
↪ [devsubid/Chatbot-using-React-Chatbot-
↪ Kit](https://github.com/devsubid/Chatbot-using-React-Chatbot-Kit) e
↪ [FredrikOseberg/react-chatbot-
↪ kit](https://github.com/FredrikOseberg/react-chatbot-kit).

Como executar:

Instale docker e docker-compose:

[Seguir esses passos pra instalar
↪ docker-compose](https://medium.com/lffintech/how-to-install-docker-and-
↪ docker-compose-on-ubuntu-24-04-arm64-without-sudo-b60c33d3c86d)

Instale nginx:
Execute o nginx com os comandos:

```bash
sudo apt update
```

```
sudo apt install nginx
```

Suba os containeres:

```bash
sudo docker-compose up --build
```

nginx/default.conf

```
server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;

    # Roteia as requisições para o frontend
    location / {
        proxy_pass http://frontend-container:5173;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /server {
        proxy_pass http://backend-container:6358;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /questions {
        proxy_pass http://backend-container:6358;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
```

```
}
```

nginx/Dockerfile

```
FROM nginx:alpine
COPY ./default.conf /etc/nginx/conf.d/default.conf
```

backend/questao_1953.pl

```
:- module(questao_1953, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1953, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1953, ["sim"], "Você apenas deve contar quantos alunos são de EPR,
↪  EHD e o restante são intrusos, ou seja, n - EPR - EHD.\n\c
                      Para contar quantos são de EPR e EHD, podemos usar um
                       ↪  dicionário, em que as chaves são EPR e EHD e o
                       ↪  valor inicia com 0. \n\c
                      Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(1953, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(1953, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                     saídas esperadas pela questão! Também é uma boa ideia
                     ↪  debugar seu código no Thonny, \c
                     revisando linha por linha, ou usando prints para
                     ↪  entender o que ele está executando.").
```

backend/questao_2807.pl

```
:- module(questao_2807, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2807, [], "Você quer uma ideia de solução pra esse problema?").
questao(2807, ["sim"], "Esses são os números da sequência de Fibonacci ao
↪  contrário. Então podemos primeiro \c
                      computar todos os N primeiros números da sequência de
                       ↪  Fibonacci e armazenar em um vetor. \c
                      \nPara a saída basta percorrer os elementos de trás
                       ↪  para frente, ou seja, da última \c
```

```
                        posição para a primeira.\n\c
                        Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2807, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2807, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                      saídas esperadas pela questão! Também é uma boa ideia
                      ↪  debugar seu código no Thonny, \c
                      revisando linha por linha, ou usando prints para
                      ↪  entender o que ele está executando.").
```

backend/questao_1281.pl

```
:- module(questao_1281, [questao/3, diagnostico/3]).

questao(1281, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1281, ["sim"], "Crie um dicionário onde cada nome de item da feira
↪  seja uma chave, \c
                        e seu preço seja o valor correspondente. \n\c
                       Depois, para cada item da lista de compras, busque no
                        ↪  dicionário o preço \c
                        e multiplique pela quantidade, acumulando para obter
                        ↪  o valor total.\n\c
                        Próxima dica?").

questao(1281, Respostas, "Deu Presentation Error?") :-
    Respostas = ["não"];
    Respostas = ["sim", "sim"].

questao(1281, Respostas, "Formate a saída com duas casas decimais. \n\c
                            Cuide com o espaço entre 'R$' e o valor total
                            ↪  para evitar erros de apresentação. \n\c
                            Deu certo?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "sim", "sim"].


% Diagnóstico com base nas respostas completas
```

```
diagnostico(1281, Respostas, "Que legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim"].


diagnostico(1281, Respostas, "Confira novamente o uso do dicionário e a
↪  formatação da saída. Certifique-se de que:\n\c
                    1. Todas as chaves e valores do dicionário estão
                    ↪  corretos.\n\c
                    2. O cálculo do total considera as quantidades
                    ↪  corretamente.\n\c
                    3. A saída está formatada com duas casas decimais e o
                    ↪  espaço entre 'R$' e o valor.\n\c
                    Além disso, você pode usar o udebug para verificar a
                    ↪  diferença entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.") :-
    Respostas = ["não", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "não"].


diagnostico(1281, _, "Além disso, você pode usar o udebug para verificar a
↪  diferença entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/questao_1187.pl

```
:- module(questao_1187, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1187, [], "Você já desenvolveu sua solução mas ela não funciona?").
questao(1187, ["sim"], "Pense se os parâmetros do seu for estão corretos: 1.
↪  Você está começando na posição \c
                        correta da linha?").
```

```
questao(1187, ["sim", "não"], "Verifique se você não está calculando junto
↪  os elementos da diagonal (o 'for' precisa \c
                                começar em i+1). Próxima dica?").

questao(1187, Respostas, "2. Você está indo até a posição correta da
↪  linha?") :-
    Respostas = ["sim", "sim"];
    Respostas = ["sim", "não", "sim"];
    Respostas = ["sim", "não", "não"].

questao(1187, Respostas, "Como você quer os elementos acima da diagonal
↪  primária e da diagonal secundária, o 'for' que \c
                              percorre as colunas precisa parar um pouco antes
                               ↪  da linha terminar. Perceba: à medida que \c
                              vamos para a próxima linha, o laço de colunas
                               ↪  começa mais adiante e termina mais cedo. O
                               ↪  \c
                             efeito é que ele forma o 'triângulo' de números que
                               ↪  queremos somar, ajustando onde começa e \c
                             termina a coleta dos números. \nEm resumo: o for de
                               ↪  colunas 'pula' alguns números no início \c
                              e no final de cada linha, criando uma seleção que
                               ↪  pega apenas os números na área triangular \c
                              superior. \nA quantidade de colunas que ele pula
                               ↪  no início e no final é a mesma. Ou seja, \c
                              se o for começa em 'i+1', termina em 'quantidade
                               ↪  máxima de colunas - i - 1'.\n Próxima
                               ↪  dica?") :-
    Respostas = ["sim", "sim", "não"];
    Respostas = ["sim", "não", "sim", "não"];
    Respostas = ["sim", "não", "não", "não"].

questao(1187, Respostas, "3. Verifique se você está calculando junto os
↪  elementos das diagonais da matriz, isso não deve \c
                              acontecer. Próxima dica?") :-
    Respostas = ["sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim"];
    Respostas = ["sim", "não", "não", "sim"];
    Respostas = ["sim", "sim", "não", "sim"];
```

```
    Respostas = ["sim", "não", "sim", "não", "sim"];
    Respostas = ["sim", "não", "não", "não", "sim"].

questao(1187, Respostas, "Você quer uma dica de qual solução adotar pra esse
↪  problema?") :-
    Respostas = ["não"];
    Respostas = ["sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "não", "não", "não", "sim", "sim"].

questao(1187, Respostas, "Para entender melhor, imagine que estamos
↪  interessados apenas nos números que estão \c
                            na parte 'triangular' no topo da matriz, sem
                            ↪  incluir os da diagonal principal (que \c
                            vai do canto superior esquerdo ao canto inferior
                            ↪  direito) e nem os da diagonal \c
                            secundária (que vai do canto superior direito ao
                            ↪  canto inferior esquerdo). \nAqui está \c
                            como cada for ajuda a alcançar esse objetivo:\n1.
                            ↪  Primeiro for (para as linhas): Esse \c
                           laço percorre apenas as primeiras linhas da matriz,
                            ↪  ou seja, a parte de cima. A quantidade \c
                            de linhas que ele percorre depende de quantas
                            ↪  queremos incluir na nossa 'região
                            ↪  triangular'. \n\c
                            2. Segundo for (para as colunas): Esse laço vai
                            ↪  selecionar só algumas das colunas em cada
                            ↪  linha:\n\c
                            Ele começa algumas colunas à frente, ou seja,
                            ↪  pula os primeiros elementos da linha para
                            ↪  evitar a \c
                            diagonal principal.\nEle também termina um pouco
                            ↪  antes do fim da linha para evitar a diagonal
                            ↪  secundária.\c\n
                            Isso significa que, à medida que vamos para a
                            ↪  próxima linha, o laço de colunas começa mais
                            ↪  adiante e \c
```

```
                                termina mais cedo. O efeito é que ele forma o
                                ↪  'triângulo' de números que queremos somar,
                                ↪  ajustando \c
                                onde começa e termina a coleta dos números.\n Em
                                ↪  resumo: o for de colunas 'pula' alguns
                                ↪  números no \c
                                início e no final de cada linha, criando uma
                                ↪  seleção que pega apenas os números na área
                                ↪  triangular \c
                                superior.\nA quantidade de colunas que ele pula
                                ↪  no início e no final é a mesma. Ou seja, \c
                                se o for começa em 'i+1', termina em 'quantidade
                                ↪  máxima de colunas - i - 1'.\n Próxima
                                ↪  dica?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "não", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "não", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "não", "não", "sim", "sim", "sim"].

questao(1187, Respostas, "Ainda recebe resposta errada?") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "não"];
    Respostas = ["sim", "não", "sim", "sim", "não"];
    Respostas = ["sim", "não", "não", "sim", "não"];
    Respostas = ["sim", "sim", "não", "sim", "não"];
    Respostas = ["sim", "não", "sim", "não", "sim", "não"];
    Respostas = ["sim", "não", "não", "não", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "não", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "não", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "não", "não", "sim", "sim", "sim", "sim"].

questao(1187, Respostas, "Verifique se a divisão da média está sendo feita
↪  de forma adequada! Deu certo?") :-
```

```
    Respostas = ["não", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "não", "sim"];
    Respostas = ["sim", "não", "sim", "sim", "não", "sim"];
    Respostas = ["sim", "não", "não", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "não", "sim"];
    Respostas = ["sim", "não", "sim", "não", "sim", "não", "sim"];
    Respostas = ["sim", "não", "não", "não", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "não", "sim", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "não", "sim", "sim", "sim", "sim",
    ↪  "sim"];
    Respostas = ["sim", "não", "não", "não", "sim", "sim", "sim", "sim",
    ↪  "sim"].


% Diagnóstico com base nas respostas completas
diagnostico(1187, Respostas, "Que legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "não", "não", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "não", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "não", "não", "não", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim", "sim", "sim", "sim", "sim",
    ↪  "sim"];
    Respostas = ["sim", "não", "não", "sim", "sim", "sim", "sim", "sim",
    ↪  "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim", "sim", "sim", "sim",
    ↪  "sim"];
    Respostas = ["sim", "não", "sim", "não", "sim", "sim", "sim", "sim",
    ↪  "sim", "sim"];
    Respostas = ["sim", "não", "não", "não", "sim", "sim", "sim", "sim",
    ↪  "sim", "sim"].

% Diagnóstico com base nas respostas completas
```

```
diagnostico(1187, _, "Atente-se bem às dicas já enviadas ou pergunte
↪  novamente! Além disso, você pode \c
                        usar o udebug para verificar a diferença entre as
                        ↪  saídas do seu código, e as \c
                       saídas esperadas pela questão! Também é uma boa ideia
                        ↪  debugar seu código no Thonny, \c
                        revisando linha por linha, ou usando prints para
                        ↪  entender o que ele está executando!").
```

backend/questao_1171.pl

```
:- module(questao_1171, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1171, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1171, ["sim"], "Como os valores de X podem ir de 1 até 2000, você
↪  pode criar um vetor onde o índice \c
                       do vetor representa cada número de o valor que estará
                        ↪  em cada posição do vetor é a quantidade \c
                        de vezes que ele aparece. \nComo no Python o vetor
                        ↪  inicia na posição zero, sugiro criar um \c
                        vetor de 2001 posições para poder incluir o número
                        ↪  2000. \nPara imprimir a saída em ordem, \c
                        basta percorrer o vetor de 1 até 2000. As posições
                        ↪  que contém 0 não devem ser impressas e as \c
                        demais basta imprimir seguindo o formato da saída.
                        ↪  \c\n
                        Próxima dica?").

questao(1171, Respostas, "Deu Wrong Answer?") :-
    Respostas = ["não"];
    Respostas = ["sim", "sim"].

questao(1171, Respostas, "Caso resposta errada: Verifique um caso de teste
↪  que inclua o valor 2000. Lembre-se \c
                          de na saída percorrer todos os valores de 1 até
                          ↪  2000, ou seja o range deve ser \c
                          range(1,2001) e não range(1, 2000). \n\c
                          https://docs.python.org/3/howto/sorting.html \n\c
```

```
                          Deu certo?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "sim", "sim"].

% Diagnóstico com base nas respostas completas
diagnostico(1171, Respostas, "Legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim"].

diagnostico(1171, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/questao_1911.pl

```
:- module(questao_1911, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1911, [], "Você quer saber como usar dicionários nessa questão?").
questao(1911, ["sim"], "Crie primeiramente um dicionário onde a chave vai
↪  ser o nome da pessoa, e o \c
                       valor vai ser a sua assinatura original.\n\c
                       Depois, para cada entrada, verifique se a assinatura
                       ↪  daquela pessoa na aula é igual \c
                       à assinatura daquela pessoa no dicionário de
                       ↪  assinaturas originais.\n\c
                       Próxima dica?").

questao(1911, Respostas, "Você quer uma dica de qual solução adotar pra esse
↪  problema?") :-
    Respostas = ["não"];
    Respostas = ["sim", "sim"].

questao(1911, Respostas, "Use um dicionário para armazenar cada assinatura
↪  de cada nome, ou seja, mapear nome para assinatura.\n\c
                          Depois, verifique para cada nome e assinatura
                          ↪  escrita na folha da chamada o seguinte: \n\c
```

```
                        1. toda a assinatura original e a assinatura da
                        ↪  chamada para o nome dado na chamada. \c
                        Se houver diferença em alguma posição, anote.\n\c
                        2. Se a quantidade de diferenças encontradas for
                        ↪  superior à 1, então some a quantidade de
                        ↪  assinaturas falsas. \n\c
                        Deu certo?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "sim", "sim"].

% Diagnóstico com base nas respostas completas
diagnostico(1911, Respostas, "Legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim"].

diagnostico(1911, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/questao_1449.pl

```
:- module(questao_1449, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1449, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1449, ["sim"], "Nesse problema, você deve considerar utilizar um
↪  dicionário para armazenar as traduções das palavras de japonês para
↪  português.\c
                        Esse dicionário deve ser utilizado zerado para cada
                        ↪  instância da entrada.\n\c
                        Depois de preencher o dicionário com a entrada, leia
                        ↪  o texto e para cada palavra do texto pesquise se
                        ↪  a palavra existe \c
                        ou não no dicionário. Se não existir, imprima ela,
                        ↪  senão se existir, então imprima a sua tradução.
                        ↪  \c\n
```

```
                        Próxima dica?").

questao(1449, Respostas, "Deu Presentation Error?") :-
    Respostas = ["não"];
    Respostas = ["sim", "sim"].

questao(1449, Respostas, "Cuide ao imprimir a primeira palavra da saída,
↪  visto que não devem haver \c
                        espaços em branco no início e fim de cada linha. \n\c
                          Deu certo?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "sim", "sim"].

% Diagnóstico com base nas respostas completas
diagnostico(1449, Respostas, "Legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim"].


diagnostico(1449, Respostas, "Certifique-se de que o dicionário está sendo
↪  reinicializado corretamente para cada instância. Verifique também:\n\c
                    1. Se as palavras e traduções estão sendo armazenadas
                    ↪  corretamente no dicionário.\n\c
                    2. Se a busca no dicionário está retornando as traduções
                    ↪  adequadas ou mantendo as palavras originais quando
                    ↪  não traduzidas.\n\c
                    3. Se a formatação da saída está sem espaços no início e
                    ↪  no fim de cada linha. \n\c
                    Além disso, você pode usar o udebug para verificar a
                    ↪  diferença entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.") :-
    Respostas = ["não", "sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim", "não"].


diagnostico(1449, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
```

```
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/questao_2134.pl

```
:- module(questao_2134, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2134, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(2134, ["sim"], "Esse problema é muito parecido com o problema do
↪  GodoFor. Não preciso ordenar todos os \c
                      dados da entrada e nem armazenar eles em um vetor.
                      ↪  Basta apenas armazenar o infeliz \c
                      reprovado atual. \nAssim, ao iniciar a leitura dos
                      ↪  dados da entrada, use o primeiro \c
                      a ser lido como o infeliz reprovado até o momento.
                      ↪  Para cada novo aluno lido, compare \c
                      o número de problemas. Se o novo aluno tiver um
                      ↪  número de problemas menor que o atual \c
                     infeliz reprovado, então atualize o infeliz reprovado
                      ↪  para ser o novo aluno lido. Se a \c
                      quantidade de problemas for igual, pode compara os
                      ↪  nomes lexicograficamente (pode usar < ou >).\n\c
                      Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2134, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2134, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                     saídas esperadas pela questão! Também é uma boa ideia
                     ↪  debugar seu código no Thonny, \c
                     revisando linha por linha, ou usando prints para
                     ↪  entender o que ele está executando.").
```

backend/questao_1259.pl

```
:- module(questao_1259, [questao/3, diagnostico/3]).
```

```
% Predicado para fornecer perguntas com base na sequência de respostas
questao(1259, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1259, ["sim"], "Para este problema, você deve ordenar os números
↪  primeiro pelos pares e depois pelos \c
                        ímpares. \nPara poder fazer isso no Python você pode
                        ↪  primeiro ler todos os números e \c
                        armazenar em um vetor da seguinte forma: crie um par
                        ↪  (t,x) onde t pode ser -1 se o \c
                       número é par e 1 se o número é ímpar. Se o número for
                        ↪  par você armazena então (-1,x)... \c
                        porém, como os números ímpares devem ser listados em
                        ↪  ordem decrescente, então basta \c
                        armazenar (1, -x) para o caso dos números serem
                        ↪  ímpares. \nPara fazer a ordenação, \c
                        basta chamar vetor.sort() e depois imprimir os
                        ↪  números. \nLembre-se que se o número for \c
                        ímpar você deve multiplicar novamente por -1 para
                        ↪  retornar o sinal correto na saída.\n\c
                        Leia mais sobre ordenação em Python em
                        ↪  https://www.geeksforgeeks.org/sort-in-
                        ↪  python/\n\c
                        Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(1259, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(1259, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                      saídas esperadas pela questão! Também é uma boa ideia
                      ↪  debugar seu código no Thonny, \c
                      revisando linha por linha, ou usando prints para
                      ↪  entender o que ele está executando.").
```

backend/questao_1184.pl

```
:- module(questao_1184, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1184, [], "Você já desenvolveu sua solução, mas não está saindo a
↪  resposta correta?").
```

```
questao(1184, ["sim"], "Você quer os elementos abaixo da diagonal principal.
↪  Ou seja, você vai percorrer por \c
                        todas as linhas, mas até o elemento com o mesmo
                        ↪  índice do número da linha da coluna. \c
                        Por exemplo, se você estiver na linha 0, você não
                        ↪  quer nenhum elemento daquela linha; \c
                        se você estiver na linha 1, você quer o primeiro
                        ↪  elemento daquela linha (elemento da \c
                       coluna 1); e assim por diante. Quer saber como fica o
                        ↪  'for?'").
questao(1184, ["sim", "sim"], "for i in range(0, 12): for j in range(0, i).
↪  Próxima dica?").
questao(1184, ["sim", "sim", "sim"], "Verifique se você inverteu esses dois
↪  'for's! Próxima dica?").
questao(1184, ["sim", "sim", "sim", "sim"], "Essa questão não quer que some
↪  a diagonal principal junto! Verifique \c
                                                se você não está somando
                                                ↪  também esses elementos.
                                                ↪  Deu certo?").


questao(1184, ["sim", "sim", "sim", "sim", "não"], "Verifique se o número
↪  pelo qual você está tentando dividir para \c
                                                 para conseguir a média está
                                                    ↪  correto. Próxima
                                                    ↪  pergunta?").


questao(1184, Respostas, "Deu Presentation Error?") :-
    Respostas = ["não"];
    Respostas = ["sim", "não"];
    Respostas = ["sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim", "não", "sim"].

questao(1184, Respostas, "Perceba que a formatação que a questão pede é:
↪  Imprima o resultado solicitado \c
                            (a soma ou média), com 1 casa após o ponto
                            ↪  decimal. Ou seja, '%.1f'. Deu certo?") :-
```

```
    Respostas = ["não", "sim"];
    Respostas = ["sim", "não", "sim"];
    Respostas = ["sim", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "não", "sim", "sim"].


% Diagnóstico com base nas respostas completas
diagnostico(1383, Respostas, "Legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "não", "sim", "sim", "sim"].

diagnostico(1184, _, "Atente-se bem às dicas já enviadas ou pergunte
↪  novamente! Além disso, você pode \c
                        usar o udebug para verificar a diferença entre as
                        ↪  saídas do seu código, e as \c
                       saídas esperadas pela questão! Também é uma boa ideia
                        ↪  debugar seu código no Thonny, \c
                        revisando linha por linha, ou usando prints para
                        ↪  entender o que ele está executando!").
```

backend/questao_1582.pl

```
:- module(questao_1582, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1582, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1582, ["sim"], "Para esse problema, basta usar as fórmulas do
↪  enunciado e computar o máximo \c
                        divisor comum usando do Algoritmo de Euclides. \c\n
                        Próxima dica?").

questao(1582, Respostas, "Deu Wrong Answer?") :-
    Respostas = ["não"];
    Respostas = ["sim", "sim"].
```

```
questao(1582, Respostas, "Cuidado com espaços e acentos. Não deve haver
↪  acento. \n\c
                          Deu certo?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "sim", "sim"].

% Diagnóstico com base nas respostas completas
diagnostico(1582, Respostas, "Legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim"].

diagnostico(1582, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/questao_1244.pl

```
:- module(questao_1244, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1244, [], "Você quer uma ideia de solução pra esse problema?").
questao(1244, ["sim"], "Esse problema é parecido com o dos ímpares e pares
↪  (1259). Podemos ler todas as palavras \c
                        e armazenar elas em um vetor usando um par (t, w),
                        ↪  onde t é o tamanho da palavra e w é a \c
                        própria palavra. \nO Python já mantém a ordem
                        ↪  original no caso de haver empates das chaves \c
                        da ordenação. \nComo estamos usando aqui um par,
                        ↪  devemos usar como chave apenas o tamanho. \c
                       Ou seja, seja v o vetor a ser ordenado, então podemos
                        ↪  fazer v.sort(key=lambda x: x[0]) onde \c
                       x é um dado a ser ordenado e o retorno é x[0] que é a
                        ↪  chave que será usada para ordenação. \c
                        O índice 0 do par é justamente o tamanho. \nNote
                        ↪  também que devemos fazer o t ser negativo, \c
                        assim (-t,w), pois queremos em ordem decrescente de
                        ↪  tamanho.\n\c
```

```
                        Veja mais sobre ordenação em Python em
                        ↪ https://docs.python.org/3/howto/sorting.html\n\c
                        Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(1244, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(1244, _, "Você pode usar o udebug para verificar a diferença
↪ entre as saídas do seu código, e as \c
                      saídas esperadas pela questão! Também é uma boa ideia
                      ↪ debugar seu código no Thonny, \c
                      revisando linha por linha, ou usando prints para
                      ↪ entender o que ele está executando.").
```

backend/questao_2479.pl

```
:- module(questao_2479, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2479, [], "Você quer uma ideia de solução pra esse problema?").
questao(2479, ["sim"], "Para esse problema podemos primeiro ler todas as
↪ crianças e ir armazenando \c
                        os nomes delas em um vetor. Ao ler cada criança, já
                        ↪ podemos também ir acumulando \c
                        em uma variável quantas se comportaram e quantas não
                        ↪ se comportaram. \nPara a saída, \c
                        podemos usar a função .sort() para ordenar os nomes
                        ↪ da criança e depois seguir o \c
                        formato do enunciado para imprimir quantas se
                        ↪ comportaram e quantas não se comportaram.\n\c
                        Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2479, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2479, _, "Você pode usar o udebug para verificar a diferença
↪ entre as saídas do seu código, e as \c
                      saídas esperadas pela questão! Também é uma boa ideia
                      ↪ debugar seu código no Thonny, \c
                      revisando linha por linha, ou usando prints para
                      ↪ entender o que ele está executando.").
```

backend/questao_2091.pl

```
:- module(questao_2091, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2091, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(2091, ["sim"], "Nesse problema, note que para encontrar o valor sem
↪  par basta encontrar uma quantidade ímpar de ocorrências de algum
↪  valor.\n\c
                        Para fazer a contagem de ocorrências dos valores
                        ↪  informados, use um dicionário, onde a chave do
                        ↪  dicionário é o valor informado.\n\c
                        Para cada chave armazene então a quantidade de
                        ↪  ocorrências de cada valor.\n\c
                        Por fim, basta percorrer todas as chaves do
                        ↪  dicionário buscando pelo valor com quantidade de
                        ↪  ocorrências sendo um número ímpar.\n\c
                        Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2091, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2091, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                      saídas esperadas pela questão! Também é uma boa ideia
                      ↪  debugar seu código no Thonny, \c
                      revisando linha por linha, ou usando prints para
                      ↪  entender o que ele está executando.").
```

backend/questao_1739.pl

```
:- module(questao_1739, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1739, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1739, ["sim"], "Essa função terá como objetivo encontrar os 60
↪  primeiros números da sequência de threebonacci. \n\c
                        Para isso vou gerando números da sequência de
                        ↪  Fibonacci e testando se eles fazem parte da \c
```

```
                        sequência de threebonacci. \nComo cada número da
                        ↪  sequência de Fibonacci é sempre gerado pela \c
                        soma dos dois últimos, então não preciso armazenar
                        ↪  todos eles. Bata eu utilizar a informação \c
                        dos dois últimos.\nComo na entrada são vários casos
                        ↪  de teste, basta armazenar no vetor uma \c
                        única vez todos os números da sequência. \n\c
                        Próxima dica?").

questao(1739, Respostas, "Deu Wrong Answer?") :-
    Respostas = ["não"];
    Respostas = ["sim", "sim"].

questao(1739, Respostas, "Note que basta uma das condições ser verdade para
↪  que ele seja um número da sequência de threebonacci. \n\c
                          Deu certo?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "sim", "sim"].

% Diagnóstico com base nas respostas completas
diagnostico(1739, Respostas, "Legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim"].

diagnostico(1739, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/questao_1181.pl

```
:- module(questao_1181, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1181, [], "Você já desenvolveu sua solução, mas não está saindo a
↪  resposta correta?").
questao(1181, ["sim"], "Tome cuidado que a questão quer todos os elementos
↪  da linha, e não da coluna! Verifique \c
```

```
                        como você está iterando sobre a matriz: se está
                        ↪  pegando todos os elementos de uma linha \c
                        ou todos os elementos de uma coluna. Mais uma dica
                        ↪  sobre isso?").
questao(1181, ["sim", "sim"], "Se sua matriz é feita de forma que é iterada
↪  assim: matriz[linha][coluna], verifique \c
                                se o 'for' está correto. Quer saber como o
                                ↪  for fica?").
questao(1181, ["sim", "sim", "sim"], "considerando que l é a entrada do
↪  usuário que indica a linha que será a \c
                                     considerada para operação: \nfor i in
                                      ↪  range(0, 12): soma +=
                                      ↪  matriz[l][i]. \n\c
                                      Próxima dica?").
questao(1181, ["sim", "sim", "sim", "sim"], "Verifique se o número pelo qual
↪  você está tentando dividir para \c
                                       para conseguir a média está correto.
                                            ↪  Próxima pergunta?").


questao(1181, Respostas, "Deu Presentation Error?") :-
    Respostas = ["não"];
    Respostas = ["sim", "não"];
    Respostas = ["sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim", "sim"].

questao(1181, Respostas, "Perceba que a formatação que a questão pede é:
↪  Imprima o resultado solicitado \c
                           (a soma ou média), com 1 casa após o ponto
                           ↪  decimal. Ou seja, '%.1f'. Deu certo?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "não", "sim"];
    Respostas = ["sim", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim"].


% Diagnóstico com base nas respostas completas
```

```
diagnostico(1181, Respostas, "Legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "não", "sim", "sim"].

diagnostico(1181, _, "Atente-se bem às dicas já enviadas ou pergunte
↪  novamente! Além disso, você pode \c
                        usar o udebug para verificar a diferença entre as
                        ↪  saídas do seu código, e as \c
                       saídas esperadas pela questão! Também é uma boa ideia
                        ↪  debugar seu código no Thonny, \c
                        revisando linha por linha, ou usando prints para
                        ↪  entender o que ele está executando!").
```

backend/questao_1028.pl

```
:- module(questao_1028, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1028, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1028, ["sim"], "O que precisamos nesse problema é do máximo divisor
↪  comum entre F1 e F2. \c
                        Para encontrar o máximo divisor comum, sugiro que
                        ↪  pesquise sobre o Algoritmo de Euclides.\n\c
                        Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(1028, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(1028, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                      saídas esperadas pela questão! Também é uma boa ideia
                      ↪  debugar seu código no Thonny, \c
                      revisando linha por linha, ou usando prints para
                      ↪  entender o que ele está executando.").
```

backend/questao_2482.pl

```
:- module(questao_2482, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2482, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(2482, ["sim"], "Nesse problema, note que você deve armazenar a
↪  tradução da saudação Merry Christmas nas diversas línguas de
↪  entrada.\n\c
                        Aqui cabe bem o uso de dicionário para isso, onde a
                        ↪  chave do dicionário é a língua e o valor de cada
                        ↪  chave é a saudação naquela língua.\n\c
                       Depois de ler todas as saudações em todas as línguas,
                        ↪  note que basta ler os nomes das crianças e
                        ↪  nacionalidades \c
                      para simplesmente consultar o dicionário e imprimir a
                        ↪  saudação na nacionalidade da criança.\n\c
                        Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2482, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2482, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                     saídas esperadas pela questão! Também é uma boa ideia
                      ↪  debugar seu código no Thonny, \c
                     revisando linha por linha, ou usando prints para
                      ↪  entender o que ele está executando.").
```

backend/questao_1430.pl

```
:- module(questao_1430, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1430, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1430, ["sim"], "Nesse problema, você deve primeiramente armazenar a
↪  tabela dada no enunciado como um dicionário, assim você vai conseguir \c
                        rapidamente descobrir a duração de cada nota. \nUma
                        ↪  vez que souber isso, basta acumular o valor de
                        ↪  cada nota num mesmo compasso.\n\c
```

```
                        Note que, na leitura, você pode utilizar diretamente
                        ↪ o input().split('/') para quebrar os compassos
                        ↪ pelo /. \n\c
                        A função .split() do Python divide uma string em
                        ↪ partes menores (substrings) com base no
                        ↪ separador enviado. \c
                        Por exemplo, se executar
                        ↪ 'maçã/banana/laranja'.split('/'), vai criar a
                        ↪ lista ['maçã', 'banana', 'laranja']. Caso não \c
                        seja enviado um separador, a função .split() usa
                        ↪ espaços em branco como padrão.\n\c
                        Após isso, basta percorrer cada compasso acumulando a
                        ↪ soma das notas. \nPor fim, ao terminar de
                        ↪ percorrer um compasso, \c
                        verifique se a soma das notas dele é igual a 1.
                        ↪ \nCuide para comparar números com ponto decimal,
                        ↪ assim \c
                        a solução é verificar se a diferença de 1 e a soma
                        ↪ das notas é menor que um certo decimal muito
                        ↪ pequeno.\nDeu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(1430, ["sim", "sim"], "Legal! Parabéns!").

diagnostico(1430, ["sim", "não"], "Certifique-se de que a tabela está
↪ corretamente armazenada como um dicionário. Verifique também:\n\c
                        1. Se os compassos estão sendo divididos corretamente
                        ↪ com `split('/')`.\n\c
                        2. Se as durações das notas estão sendo acumuladas
                        ↪ corretamente para cada compasso.\n\c
                        3. Se a soma das notas está sendo comparada a 1 com a
                        ↪ tolerância adequada para números decimais.\n\c
                        Você pode usar o udebug para verificar a diferença entre
                        ↪ as saídas do seu código, e as \c
                        saídas esperadas pela questão! Também é uma boa ideia
                        ↪ debugar seu código no Thonny, \c
                        revisando linha por linha, ou usando prints para
                        ↪ entender o que ele está executando.").
```

```
diagnostico(1430, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/questao_2398.pl

```
:- module(questao_2398, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2398, [], "Você quer uma ideia de solução pra esse problema?").
questao(2398, ["sim"], "A partir de cada dica, marco todas as casas que
↪  consigo alcançar que respeitam \c
                      aquela dica. Essa marcação é feita por meio de um
                      ↪  acumulador, então sempre somo 1, para \c
                      contar a quantidade de dicas que levam até aquela
                      ↪  casa. \nAo final da última dica, devo \c
                      contar a quantidade de casas que atenderam todas as
                      ↪  dicas. Se for mais de uma casa, \c
                      então não sei onde está o tesouro, caso contrário,
                      ↪  sei onde está o tesouro.\n\c
                      Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2398, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2398, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                     saídas esperadas pela questão! Também é uma boa ideia
                     ↪  debugar seu código no Thonny, \c
                     revisando linha por linha, ou usando prints para
                     ↪  entender o que ele está executando.").
```

backend/questao_2987.pl

```
:- module(questao_2987, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2987, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
```

```
questao(2987, ["sim"], "Cada letra do alfabeto é chave do dicionário e o seu
↪  valor é a posição da letra.\n\c
                      Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2987, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2987, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                     saídas esperadas pela questão! Também é uma boa ideia
                     ↪  debugar seu código no Thonny, \c
                     revisando linha por linha, ou usando prints para
                     ↪  entender o que ele está executando.").
```

backend/questao_1383.pl

```
:- module(questao_1383, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1383, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1383, ["sim"], "No sudoku, cada linha, coluna e bloco 3x3 deve
↪  conter todos os números de 1 a 9, \n
                      sem repetições. \nNão adianta verificar se a soma de
                      ↪  todos os elementos de uma linha,\c
                      por exemplo, é igual a soma de 1 a 9, porque existem
                      ↪  outras combinações de números que \c
                      resultam nessa soma.\n Uma possível solução é criar
                      ↪  um vetor para cada linha, coluna e \c
                     bloco 3x3. Em seguida, para cada número encontrado na
                      ↪  linha, coluna ou bloco, incremente \c
                      em 1 a posição correspondente no vetor. Por fim,
                      ↪  verifique se todos os elementos desse \c
                      vetor são iguais a 1. Deu certo?").


% Diagnóstico com base nas respostas completas
diagnostico(1383, ["sim", "sim"], "Legal! Parabéns!").

diagnostico(1383, _, "Atente-se bem às dicas já enviadas ou pergunte
↪  novamente! Além disso, você pode \c
```

```
                        usar o udebug para verificar a diferença entre as
                        ↪ saídas do seu código, e as \c
                       saídas esperadas pela questão! Também é uma boa ideia
                        ↪ debugar seu código no Thonny, \c
                        revisando linha por linha, ou usando prints para
                        ↪ entender o que ele está executando!").
```

backend/questao_1435.pl

```
:- module(questao_1435, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1435, [], "Você quer uma dica de qual solução adotar pra esse
↪ problema?").
questao(1435, ["sim"], "Pense na menor distância entre as quatro bordas. A
↪ partir do ponto inicial, qual a  \c
                      distância do ponto até a primeira linha, até a última
                       ↪ linha, até a primeira coluna e \c
                       até a última coluna? E qual é o mínimo entre essas
                       ↪ quatro distâncias? Pense um pouco \c
                       sobre isso, e me informe assim que entender: você
                       ↪ gostaria de uma dica sobre como fazer \c
                       isso no código?").
questao(1435, ["sim", "sim"], "Crie uma matriz de n linhas e colunas, e,
↪ para cada posição da matriz, calcule \c
                             a menor a distância entre aquela posição e as
                              ↪ extremidades da matriz. Você quer \c
                              saber como calcular essas distâncias?").
questao(1435, ["sim", "sim", "sim"], "Para a distância entre o ponto e o
↪ último elemento da sua linha, faça \c
                                   [ total de colunas - coluna do elemento
                                      ↪ ]. Para a distância entre o \c
                                   ponto e o último elemento da sua coluna,
                                      ↪ faça [ total de linhas - \c
                                      linha do elemento ]. As distâncias
                                      ↪ entre o ponto e os primeiros
                                      ↪ elementos \c
                                      da linha e da coluna são a própria
                                      ↪ posição do elemento! Agora é só
                                      ↪ calcular \c
```

```
                                        o mínimo entre essas quatro
                                        ↪ distâncias! Quer ir para a
                                        ↪ próxima pergunta?").

questao(1435, Respostas, "Deu Run-time Error?") :-
    Respostas = ["não"];
    Respostas = ["sim", "não"];
    Respostas = ["sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim"].

questao(1435, Respostas, "Verifique se você não está acessando alguma
↪ posição inválida da matriz, como, por exemplo, \c
                            se sua matriz tiver 10 linhas e você tenta
                            ↪ acessar a 11ª linha. Verifique \c
                            também se a estrutura do código está correta.
                            ↪ Gostaria de ir para a próxima pergunta?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "não", "sim"];
    Respostas = ["sim", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim"].

questao(1435, Respostas, "Deu Presentation Error?") :-
    Respostas = ["não", "não"];
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "não", "não"];
    Respostas = ["sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "não"];
    Respostas = ["sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim"].

questao(1435, Respostas, "Preste atenção nos seguintes requisitos: 1. Deve
↪ formatar o print do número com 3 casas \c
                            decimais (com %3d); 2. Deve ter espaço em branco
                            ↪ entre cada número; 3. Deve ter quebra \c
                           de linha entre os casos de teste. Deu certo?") :-
    Respostas = ["não", "não", "sim"];
    Respostas = ["não", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "não", "sim"];
```

```
    Respostas = ["sim", "não", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "não", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim", "sim"].


% Diagnóstico com base nas respostas completas
diagnostico(1435, Respostas, "Atente-se bem às dicas já enviadas! Além
↪  disso, você pode usar o udebug para  \c
                                verificar a diferença entre as saídas do seu
                                ↪  código, e as saídas esperadas \c
                                pela questão! Também é uma boa ideia debugar
                                ↪  seu código no Thonny, revisando \c
                                linha por linha, ou usando prints para
                                ↪  entender o que ele está executando!") :-
    Respostas = ["não", "não", "sim", "não"];
    Respostas = ["não", "sim", "sim", "sim", "não"];
    Respostas = ["sim", "não", "não", "sim", "não"];
    Respostas = ["sim", "não", "sim", "sim", "sim", "não"];
    Respostas = ["sim", "sim", "não", "não", "sim", "não"];
    Respostas = ["sim", "sim", "não", "sim", "sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim", "não", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim", "sim", "não"].

% Diagnóstico com base nas respostas completas
diagnostico(1435, Respostas, "Que legal! Parabéns!") :-
    Respostas = ["não", "não", "sim", "sim"];
    Respostas = ["não", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "não", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim", "sim", "sim"].

% Diagnóstico com base nas respostas completas
diagnostico(1435, _, "Atente-se bem às dicas já enviadas ou pergunte
↪  novamente! Além disso, você pode \c
```

```
                                usar o udebug para verificar a diferença
                                ↪ entre as saídas do seu código, e as \c
                               saídas esperadas pela questão! Também é uma boa
                                ↪ ideia debugar seu código no Thonny, \c
                               revisando linha por linha, ou usando prints
                                ↪ para entender o que ele está
                                ↪ executando!").
```

backend/questao_2149.pl

```
:- module(questao_2149, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2149, [], "Você quer uma dica de qual solução adotar pra esse
↪ problema?").
questao(2149, ["sim"], "Para esse problema apenas precisamos apenas dos
↪ primeiros 17 números da sequência, \c
                        então podemos criar um vetor com 17 posições e
                        ↪ computar todos os números logo no início, \c
                       pois são vários casos de teste e podemos reusar o que
                        ↪ já computamos. Assim, podemos criar \c
                        uma função para computar esses valores.\n\c
                        Note que a sequência é basicamente a sequência de
                        ↪ Fibonacci modificada. Se a posição do \c
                        elemento que queremos é par, então multiplicamos os
                        ↪ dois anteriores. Se a posição do \c
                        elemento que queremos é ímpar, então somamos os dois
                        ↪ anteriores.\n\c
                        Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2149, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2149, _, "Você pode usar o udebug para verificar a diferença
↪ entre as saídas do seu código, e as \c
                      saídas esperadas pela questão! Também é uma boa ideia
                      ↪ debugar seu código no Thonny, \c
                      revisando linha por linha, ou usando prints para
                      ↪ entender o que ele está executando.").
```

backend/questao_2137.pl

```
:- module(questao_2137, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2137, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(2137, ["sim"], "Para esse problema basta armazenar todos os dados da
↪  entrada em um vetor e \c
                      então usar a função .sort() do vetor para ordenar.\n\c
                        Leia mais sobre ordenação em Python em
                        ↪  https://www.geeksforgeeks.org/sort-in-
                        ↪  python/\n\c
                        Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2137, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2137, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                      saídas esperadas pela questão! Também é uma boa ideia
                      ↪  debugar seu código no Thonny, \c
                      revisando linha por linha, ou usando prints para
                      ↪  entender o que ele está executando.").
```

backend/questao_1261.pl

```
:- module(questao_1261, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1261, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1261, ["sim"], "Crie um dicionário em que cada palavra de entrada
↪  (as M palavras) seja uma chave, \c
                        e o valor correspondente em dólar seja o valor. \n\c
                       Sempre que encontrar uma dessas palavras na descrição
                        ↪  do cargo, \c
                        some o valor correspondente ao salário total.\n\c
                        Próxima dica?").

questao(1261, ["sim", "sim"], "Em cada linha recebida, use o input().split()
↪  para tratar cada palavra da linha. \c
```

```
                                Não é necessário receber o texto inteiro
                                ↪ primeiramente, para depois verificar cada
                                ↪ linha.
                                Próxima dica?").

questao(1261, Respostas, "Seu resultado foi Wrong Answer?"):-
    Respostas = ["não"];
    Respostas = ["sim", "não"];
    Respostas = ["sim", "sim", "sim"].

questao(1261, Respostas, "Verifique se você está considerando a quantidade
↪ de vezes que uma palavra aparece. \c
                        Quando uma palavra aparece várias vezes, a pontuação
                          ↪ final fica maior? Ou apenas é considerada \c
                          uma ocorrência da palavra no salário total?
                          ↪ Verifique. \n\c
                          Próxima dica?"):-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "sim"].

questao(1261, Respostas, "Para cada palavra do texto, verifique se ela
↪ aparece no dicionário. Se sim, some a pontuação que ela tem. \n\c
                          Próxima dica?"):-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim"].

questao(1261, Respostas, "Seu resultado foi Run-time Error?"):-
    Respostas = ["não", "não"];
    Respostas = ["sim", "não", "não"];
    Respostas = ["sim", "sim", "sim", "não"];
    Respostas = ["não", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim", "não"];
    Respostas = ["não", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim"].

questao(1261, Respostas, "Você precisa verificar se está tentando fazer
↪ input().split() na linha que possui apenas um '.' \n\c
```

```
                          Para tratar esse caso, antes de usar a função
                          ↪ split(), verifique se o conteúdo da linha não
                          ↪ é igual a '.'"):-
    Respostas = ["não", "não", "sim"];
    Respostas = ["sim", "não", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "não", "sim"];
    Respostas = ["não", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "não", "sim"];
    Respostas = ["não", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim", "sim"].

% Diagnóstico com base nas respostas completas
diagnostico(1261, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(1261, _, "Você pode \c
                      usar o udebug para verificar a diferença entre as
                      ↪ saídas do seu código, e as \c
                     saídas esperadas pela questão! Também é uma boa ideia
                      ↪ debugar seu código no Thonny, \c
                      revisando linha por linha, ou usando prints para
                      ↪ entender o que ele está executando!").
```

backend/questao_2465.pl

```
:- module(questao_2465, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2465, [], "Você quer uma dica de qual solução adotar pra esse
↪ problema?").
questao(2465, ["sim"], "Você pode usar um while True para verificar, a
↪ partir da posição (a, b) na matriz, qual vizinho \c
                      tem valor '1'. Quando encontrar um vizinho com valor
                      ↪ 1, o robô avança para essa posição. Por
                      ↪ exemplo:\c\n
                      Se for o vizinho da direita, incremente b em 1; \n\c
                      Se for o vizinho da esquerda, decremente b em 1;\n\c
                      Se for o vizinho de cima, decremente a em 1;\n\c
                      Se for o vizinho de baixo, incremente a em 1.\n\c
                      Caso não encontre nenhum vizinho com valor 1, use
                      ↪ break para sair do while True.\c\n
```

```
                        Quer uma dica de como não voltar para a posição que
                        ↪ estava antes?").
questao(2465, ["sim", "sim"], "Marca a posição (a, b) como zero, antes de ir
↪ para o próximo vizinho! Deu certo?").


% Diagnóstico com base nas respostas completas

diagnostico(2465, ["sim", "sim", "sim"], "Legal! Parabéns!").

diagnostico(2465, _, "Atente-se bem às dicas já enviadas ou pergunte
↪ novamente! Além disso, você pode \c
                        usar o udebug para verificar a diferença entre as
                        ↪ saídas do seu código, e as \c
                       saídas esperadas pela questão! Também é uma boa ideia
                        ↪ debugar seu código no Thonny, \c
                        revisando linha por linha, ou usando prints para
                        ↪ entender o que ele está executando!").
```

backend/questao_2654.pl

```
:- module(questao_2654, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2654, [], "Você quer saber como usar dicionários nessa questão?").
questao(2654, ["sim"], "Você pode criar um dicionário, onde a chave seria o
↪ nome do personagem, e o valor \c
                        seria uma lista com os atributos daquele personagem.
                        ↪ No final, procure qual personagem \c
                        possui o melhor atributo.\n\c
                        Próxima dica?").


questao(2654, Respostas, "Você quer uma dica de qual solução adotar pra esse
↪ problema?") :-
    Respostas = ["não"];
    Respostas = ["sim", "sim"].

questao(2654, Respostas, "Nesse problema você deve encontrar o primeiro
↪ elemento de uma ordenação com base em alguns critérios. \c
```

```
                            Como não é necessário saber quem é o segundo,
                             ↪  terceiro, etc... apenas deve pensar em
                             ↪  armazenar o \c
                           'primeiro atual' e sempre comparar com os seguintes
                             ↪  verificando se alguém é melhor que ele.\c
                          Para isso, armazene inicialmente o primeiro ser como
                             ↪  o melhor de todos. Depois, para cada novo \c
                            ser lido, basta comparar com o ser armazenado como
                             ↪  melhor de todos.\n\c
                           Se o novo ser é melhor, então troco, caso contrário
                             ↪  mantenho o melhor.\n\c
                            Note que não é necessário ordenar todos os seres,
                             ↪  pois basta controlar o primeiro colocado.\n\c
                            Próxima dica?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "sim", "sim"].

questao(2654, Respostas, "Você já desenvolveu sua solução mas ela não
↪  funciona? Ou gostaria de outra dica?") :-
    Respostas = ["não", "não"];
    Respostas = ["sim", "sim", "não"];
    Respostas = ["sim", "não"];
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim"].

questao(2654, Respostas, "Maiúsculo e minúsculo são diferentes. Assim, 'G'
↪  vem antes de 'a' lexicograficamente.\n\c
                            Deu certo?") :-
    Respostas = ["não", "não", "sim"];
    Respostas = ["sim", "sim", "não", "sim"];
    Respostas = ["sim", "não", "sim"];
    Respostas = ["não", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim"].

% Diagnóstico com base nas respostas completas
diagnostico(2654, Respostas, "Legal! Parabéns!"):-
    Respostas = ["não", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim"];
```

```
    Respostas = ["não", "sim", "sim", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim"].

diagnostico(2654, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                     saídas esperadas pela questão! Também é uma boa ideia
                     ↪  debugar seu código no Thonny, \c
                     revisando linha por linha, ou usando prints para
                     ↪  entender o que ele está executando.").
```

backend/questao_2729.pl

```
:- module(questao_2729, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2729, [], "Você quer uma ideia de solução pra esse problema?").
questao(2729, ["sim"], "Para esse problema podemos notar que existem no
↪  máximo 1000 itens em cada lista \c
                      (são poucos), então podemos simplesmente ler cada uma
                      ↪  das listas e ordenar os \c
                      ítens dessa lista. \nPara a impressão da resposta
                      ↪  devemos apenas imprimir um \c
                      determinado item se ele ainda não foi impresso antes.
                      ↪  Como agora os itens já \c
                      estão ordenados, então basta sempre olhar se o
                      ↪  anterior é diferente do atual. Se \c
                      for diferente, então podemos imprimir o atual.\n\c
                      Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2729, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2729, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                     saídas esperadas pela questão! Também é uma boa ideia
                     ↪  debugar seu código no Thonny, \c
                     revisando linha por linha, ou usando prints para
                     ↪  entender o que ele está executando.").
```

backend/questao_1763.pl

```
:- module(questao_1763, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1763, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1763, ["sim"], "Crie um dicionário onde a chave é o país e a
↪  mensagem de Natal é o valor. \n\c
                       Para cada item da entrada, verifique se ele existe no
                        ↪  dicionário. \n\c
                       Se não existir, imprima o not found. Caso contrário,
                        ↪  imprima a mensagem de Natal. \n\c
                       Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(1763, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(1763, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                     saídas esperadas pela questão! Também é uma boa ideia
                     ↪  debugar seu código no Thonny, \c
                     revisando linha por linha, ou usando prints para
                     ↪  entender o que ele está executando.").
```

backend/questao_2251.pl

```
:- module(questao_2251, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2251, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(2251, ["sim"], "Note que o número de movimentos necessários cresce
↪  exponencialmente. Assim, o número \c
                       de movimentos pode ser obtido pela fórmula 2^N -1
                        ↪  onde N é o tamanho da torre original\n\c
                       Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2251, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2251, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
```

```
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪ debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪ entender o que ele está executando.").
```

backend/questao_1583.pl

```
:- module(questao_1583, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1583, [], "Você quer uma dica de qual solução adotar pra esse
↪ problema?").
questao(1583, ["sim"], "Nesse problema, similar ao problema 2465, devo
↪ verificar todas as casas que \c
                     consigo atingir com o agente contaminante. \nAssim,
                     ↪ inicialmente devo descobrir \c
                    todas as casas com T e armazenar as coordenadas delas
                     ↪ (i,j) em uma lista. \c
                     Depois, enquanto houverem casas com T nessa lista,
                     ↪ devo marcar todas as casas \c
                     com A em sua vizinhança (os 4 vizinhos de sempre)
                     ↪ agora com o símbolo T e armazenar \c
                     essas novas casas com T também nessa lista. \nEu
                     ↪ repito esse procedimento enquanto \c
                    essa lista contiver algum elemento. \nNote que também
                     ↪ devo controlar para evitar que \c
                     uma mesma posição (i,j) seja incluída denovo nessa
                     ↪ mesma lista. Assim, devo utilizar \c
                     uma matriz auxiliar para marcar, por exemplo, as
                     ↪ casas já participantes dessa lista em algum
                     ↪ momento.\n\c
                     Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(1583, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(1583, _, "Você pode usar o udebug para verificar a diferença
↪ entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪ debugar seu código no Thonny, \c
```

```
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/questao_1185.pl

```
:- module(questao_1185, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1185, [], "Você já desenvolveu sua solução, mas não está saindo a
↪  resposta correta?").
questao(1185, ["sim"], "Você quer os elementos acima da diagonal secundária.
↪  Ou seja, você vai percorrer por \c
                      todas as linhas, mas até o último elemento da coluna
                      ↪  menos o índice do número da linha. \c
                      Por exemplo, se você estiver na linha 0, você todos
                      ↪  os elementos daquela linha; se você estiver \c
                      na linha 1, você todos os elementos daquela linha -
                      ↪  1; se você estiver na linha 2, \c
                      você quer todos os elementos daquela linha - 2, e
                      ↪  assim por diante. \n Preste atenção que \c
                      você não quer os elementos da diagonal secundária!
                      ↪  Pensei um pouco sobre isso e depois me \c
                      responda: quer saber como fica o 'for?'").
questao(1185, ["sim", "sim"], "for i in range(0, 12): for j in range(0,
↪  12-i-1). Próxima dica?").
questao(1185, ["sim", "sim", "sim"], "Verifique se você inverteu esses dois
↪  'for's! Próxima dica?").
questao(1185, ["sim", "sim", "sim", "sim"], "Verifique se o número pelo qual
↪  você está tentando dividir para \c
                                        para conseguir a média está correto.
                                        ↪  Próxima pergunta?").


questao(1185, Respostas, "Deu Presentation Error?") :-
    Respostas = ["não"];
    Respostas = ["sim", "não"];
    Respostas = ["sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "não"];
    Respostas = ["sim", "sim", "sim", "sim", "sim"].
```

```
questao(1185, Respostas, "Perceba que a formatação que a questão pede é:
↪  Imprima o resultado solicitado \c
                            (a soma ou média), com 1 casa após o ponto
                            ↪  decimal. Ou seja, '%.1f'. Deu certo?") :-
    Respostas = ["não", "sim"];
    Respostas = ["sim", "não", "sim"];
    Respostas = ["sim", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "não", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "sim", "sim"].


% Diagnóstico com base nas respostas completas
diagnostico(1383, Respostas, "Legal! Parabéns!") :-
    Respostas = ["não", "sim", "sim"];
    Respostas = ["sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "não", "sim", "sim"];
    Respostas = ["sim", "sim", "sim", "sim", "não", "sim", "sim"].

diagnostico(1185, _, "Atente-se bem às dicas já enviadas ou pergunte
↪  novamente! Além disso, você pode \c
                        usar o udebug para verificar a diferença entre as
                        ↪  saídas do seu código, e as \c
                       saídas esperadas pela questão! Também é uma boa ideia
                        ↪  debugar seu código no Thonny, \c
                        revisando linha por linha, ou usando prints para
                        ↪  entender o que ele está executando!").
```

backend/questao_1715.pl

```
:- module(questao_1715, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1715, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1715, ["sim"], "Você poderia multiplicar os elementos da linha, e,
↪  caso essa multiplicação for diferente de 0, \c
                        soma 1 no contador de jogadores que fizeram gols em
                        ↪  todas as partidas! Deu certo?").
```

```
% Diagnóstico com base nas respostas completas

diagnostico(1715, ["sim", "sim"], "Legal! Parabéns!").

diagnostico(1715, _, "Atente-se bem às dicas já enviadas ou pergunte
↪  novamente! Além disso, você pode \c
                        usar o udebug para verificar a diferença entre as
                        ↪  saídas do seu código, e as \c
                       saídas esperadas pela questão! Também é uma boa ideia
                        ↪  debugar seu código no Thonny, \c
                        revisando linha por linha, ou usando prints para
                        ↪  entender o que ele está executando!").
```

backend/server.pl

```
:- use_module(library(http/thread_httpd)).
:- use_module(library(http/http_dispatch)).
:- use_module(library(http/http_error)).
:- use_module(library(http/html_write)).
:- use_module(library(http/http_json)).
:- use_module(library(http/http_session)).
:- use_module(library(http/http_client)).
:- use_module(library(http/http_cors)).

% Define o handler para receber as requisições HTTP
:- http_handler(root(server), handle_post_request, [method(post)]).
:- http_handler(root(questions), handle_get_questions, [method(get)]).
:- http_handler(root(.), handle_options, [method(options)]).

get_frontend_url(URL) :-
    getenv('VITE_FRONTEND_URL', URL).

handle_options(Request) :-
    get_frontend_url(FrontendUrl),

    cors_enable(Request, [methods([post, get, options])]),
    format('Access-Control-Allow-Origin: ~w~n', [FrontendUrl]),
    format('Access-Control-Allow-Credentials: true~n'),
    format('Content-Length: 0~n~n').
```

```
% Inicialização do servidor na porta 6357
:- initialization main.

main :-
    server(6358).

server(Port) :-
    http_server(http_dispatch, [port(Port), allow_cors(true)]).

% Manipulador de requisições POST
handle_post_request(Request) :-
    member(method(post), Request), !,

    % Garante que a sessão está ativa para o cliente
    http_session_id(_SessionID),

    % Lê os dados da requisição
    http_read_data(Request, Data, [encoding(utf8)]),

    get_frontend_url(FrontendUrl),

    % Definindo os headers para permitir CORS
    format('Access-Control-Allow-Origin: ~w~n', [FrontendUrl]),
    format('Access-Control-Allow-Credentials: true~n'),
    format('Content-Type: application/json; charset=UTF-8'),
    cors_enable(Request, [methods([post])]),

    (   % Caso seja a primeira requisição da questão
        member(questionNumber=QN, Data)
    ->  % Inicializa a sessão e a questão
        http_session_retractall(questionNumber(_)),
        http_session_retractall(answers("Answers", _)),
        http_session_assert(answers("Answers", [])),

        http_session_assert(questionNumber(QN)),
        start_question(QN, FirstQuestion),

        reply_json_dict(FirstQuestion)
```

```
    ;   % Caso receba respostas subsequentes
        member(answer=Answer, Data),

        % Garante que a sessão tenha a variável questionNumber
        ( http_session_data(questionNumber(QN)) ->
            continue_question(QN, Answer, NextQuestionOrResult),
            reply_json_dict(NextQuestionOrResult)
        ;   % Erro caso a sessão não tenha um questionNumber
            reply_json_dict(_{error: "Número da questão não encontrado na
            ↪  sessão"})
        )
    ).

% Carrega o arquivo de questão com base no número e executa a questão
↪  correspondente
start_question(QuestionNumber, FirstQuestion) :-
    atom_concat('questao_', QuestionNumber, FileName),
    atom_concat(FileName, '.pl', FilePath),

    % Verifica se o arquivo existe antes de tentar carregá-lo
    (   exists_file(FilePath)
    ->  % Carrega dinamicamente o arquivo da questão
        abolish(questao/3),
        abolish(diagnostico/3),

        ensure_loaded(FilePath),
        atom_number(QuestionNumber, QN),

        % Executa o predicado correspondente à questão com o número recebido
        (   catch(call(questao(QN, [], FirstQuestionResult)), E2,
                (FirstQuestion = _{error: "Erro ao processar a questão: " +
                ↪  E2}))
        ),
        (   var(FirstQuestionResult) % Verifica se ocorreu erro na chamada
        ->  FirstQuestion = _{error: "Erro desconhecido"}
        ;   FirstQuestion = _{question: FirstQuestionResult}
        )
    ;   FirstQuestion = _{error: "Questão não encontrada."}
```

```
    ).

% Processa a resposta recebida e decide a próxima pergunta ou resultado
↪  final
continue_question(QuestionNumber, Answer, Response) :-
    (   string(Answer) -> AW = Answer
    ;   atom(Answer) -> atom_string(Answer, AW)
    ;   term_string(Answer, AW)
    ),
    atom_number(QuestionNumber, QN),

    % Salva a resposta na sessão
    http_session_data(answers("Answers", AnswerList)),
    append(AnswerList, [AW], UpdatedAnswers),
    http_session_retractall(answers("Answers", _)),
    http_session_assert(answers("Answers", UpdatedAnswers)),

    % Chama a questão correspondente com a lista atualizada de respostas
    (   questao(QN, UpdatedAnswers, NextQuestion)
    ->  Response = _{question: NextQuestion}
    ;   (   diagnostico(QN, UpdatedAnswers, Result)
        ->  true
        ;   Result = "Sem conclusão final."
        ),
        Response = _{result: Result}
    ).


% Manipulador de requisições GET
handle_get_questions(Request) :-
    cors_enable(Request, [methods([get])]),

    get_frontend_url(FrontendUrl),

    % Definindo os headers para permitir CORS
    format('Access-Control-Allow-Origin: ~w~n', [FrontendUrl]),
    format('Access-Control-Allow-Credentials: true~n'),
    format('Content-Type: application/json; charset=UTF-8'),
```

```
    working_directory(CWD, CWD),
    find_question_files(CWD, QuestionNumbers),
    reply_json(QuestionNumbers).

% Encontrar arquivos de questão no diretório e extrair os números
find_question_files(Directory, QuestionNumbers) :-
    directory_files(Directory, Files), % Listar todos os arquivos no
    ↪  diretório
    include(is_question_file, Files, QuestionFiles), % Filtrar arquivos no
    ↪  formato correto
    maplist(extract_question_number, QuestionFiles, QuestionNumbers).

% Verifica se o arquivo está no formato 'questao_{numero}.pl'
is_question_file(File) :-
    sub_string(File, _, _, _, 'questao_'),
    sub_string(File, _, 3, 0, '.pl').

% Extrair o número da questão a partir do nome do arquivo
extract_question_number(File, Number) :-
    sub_string(File, Start, Length, _, 'questao_'),
    AfterPrefix is Start + Length,
    sub_string(File, AfterPrefix, _, 3, NumberString), % Extrai o número
    ↪  antes do '.pl'
    atom_number(NumberString, Number). % Converte para número.
```

backend/questao_2492.pl

```
:- module(questao_2492, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2492, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(2492, ["sim"], "Nesse problema devo dizer se as conexões
↪  apresentadas formam uma função invertível, não invertível \c
                        ou não são uma função.\n\c
                        Para uma função ser invertível, preciso para uma
                        ↪  mesma entrada ter uma única saída, se inverter a
                        ↪  saída \c
                        e a entrada, isso deve se manter também.\n\c
```

```
                        Para uma função ser não invertível, preciso apenas
                        ↪ que para uma mesma entrada eu tenha uma única
                        ↪ saída, \c
                       porém se inverter a saída pela entrada, isso pode não
                        ↪ se manter.\n\c
                        Se não for nenhuma das duas acima, então não é uma
                        ↪ função.\n\c
                        Quer saber como resolver isso no código?").

questao(2492, ["sim", "sim"], "Posso resolver esse problema usando dois
↪ dicionários. Um para armazenar todas as saídas para uma \c
                        determinada entrada e outro para armazenar todas as
                        ↪ entradas para uma determinada saída.\n\c
                       Assim, posso contar quantas saídas diferentes eu tenho
                        ↪ para cada entrada e contar quantas entradas \c
                        diferentes eu tenho para cada saída.\n\c
                        Para o primeiro dicionário, a chave seria a entrada
                        ↪ da função e o valor seria uma lista com as
                        ↪ saídas;\n\c
                        Para a contagem, basta contar quantos elementos
                        ↪ existem na lista.\n\c
                        Não existe na entrada uma conexão repetida, ou seja
                        ↪ duas vezes A -> B. Dessa forma também poderia
                        ↪ usar como \c
                        valor do meu dicionário apenas um contador ao invés
                        ↪ da lista.\n\c
                        Próxima dica?").

questao(2492, ["sim", "sim", "sim"], "Para e entrada de cada conexão posso
↪ usar input().split('->') assim, quebrando onde existe '->'.\n\c
                        left, right = input().split('->') #Posso quebrar a
                        ↪ entrada pelo '->'\n\c
                       left = left.strip() #Depois removo os espaços antes e
                        ↪ depois\n\c
                        right = right.strip()\n\c
                        Deu certo?").


% Diagnóstico com base nas respostas completas
```

```
diagnostico(2492, ["sim", "sim", "sim", "sim"], "Legal! Parabéns!").
diagnostico(2492, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/questao_2949.pl

```
:- module(questao_2949, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(2949, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(2949, ["sim"], "Cada caractere será chave do dicionário e terá como
↪  0 seu valor inicial. Para cada pessoa \c
                      da entrada, veja seu tipo e some 1 no valor
                      ↪  correspondente à chave no dicionário.\n\c
                      Deu certo?").

% Diagnóstico com base nas respostas completas
diagnostico(2949, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(2949, _, "Você pode usar o udebug para verificar a diferença
↪  entre as saídas do seu código, e as \c
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/questao_1483.pl

```
:- module(questao_1483, [questao/3, diagnostico/3]).

% Predicado para fornecer perguntas com base na sequência de respostas
questao(1483, [], "Você quer uma dica de qual solução adotar pra esse
↪  problema?").
questao(1483, ["sim"], "Para cada número n e m, compare primeiramente os 4
↪  últimos dígitos, depois os 3 últimos dígitos e depois os dois últimos
↪  dígitos.\n\c
```

```
                        Para pegar os 4 últimos dígitos basta usar o módulo
                        ↪ (resto) por 10000.\n\c
                        Para pegar os 3 últimos, o módulo é por 1000... e
                        ↪ assim por diante.\n\c
                        Se nenhum dos casos os dígitos são iguais, então
                        ↪ verifique se os dois últimos dígitos fazem parte
                        ↪ de um mesmo grupo de 4.\n\c
                        Por exemplo, 01, 02, 03, 04 deveriam fazer parte de
                        ↪ um mesmo grupo.\n\c
                       Note que é possível subtrair 1 desses valores acima e
                        ↪ extrair o quociente da divisão por 4.\n\c
                       Se dois números possuem o mesmo quociente, então eles
                        ↪ fazem parte do mesmo grupo.\n\c
                        Por exemplo: se m e n terminarem com 29 e 31, eles
                        ↪ fazem parte do mesmo grupo, pois 29-1=28 e
                        ↪ 31-1=30 e o quociente da divisão \c
                        por 4 de ambos é 7 (eles fazem parte do grupo 7).
                        Deu certo?").


% Diagnóstico com base nas respostas completas
diagnostico(1483, ["sim", "sim"], "Legal! Parabéns!").
diagnostico(1483, ["sim", "não"], "Certifique-se de que está comparando os
↪ dígitos corretamente. Verifique também:\n\c
                                   1. Se os módulos estão sendo calculados
                                   ↪ corretamente para 10000, 1000 e
                                   ↪ 100.\n\c
                                   Você pode usar o udebug para verificar a
                                   ↪ diferença entre as saídas do seu
                                   ↪ código, e as \c
                                   saídas esperadas pela questão! Também é
                                   ↪ uma boa ideia debugar seu código no
                                   ↪ Thonny, \c
                                   revisando linha por linha, ou usando
                                   ↪ prints para entender o que ele está
                                   ↪ executando.").

diagnostico(1483, _, "Você pode usar o udebug para verificar a diferença
↪ entre as saídas do seu código, e as \c
```

```
                    saídas esperadas pela questão! Também é uma boa ideia
                    ↪  debugar seu código no Thonny, \c
                    revisando linha por linha, ou usando prints para
                    ↪  entender o que ele está executando.").
```

backend/Dockerfile

```
FROM swipl:latest

WORKDIR /app

COPY server.pl /app/

COPY questao_*.pl /app/

EXPOSE 6358

CMD ["swipl", "server.pl"]
```

frontend/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/bee_helper.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Expert Bee</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link h-
    ↪  ref="https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600
    ↪  rel="stylesheet">
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
```

frontend/.gitignore

```
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*
lerna-debug.log*

node_modules
dist
dist-ssr
*.local

# Editor directories and files
.vscode/*
!.vscode/extensions.json
.idea
.DS_Store
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?

yarn.lock
package-lock.json
```

frontend/package.json

```
{
  "name": "chat-bot",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
```

```
    "lint": "eslint src --ext ts,tsx --report-unused-disable-directives
    ↪ --max-warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "@emotion/react": "^11.13.3",
    "@emotion/styled": "^11.13.0",
    "@mui/icons-material": "^6.1.6",
    "@mui/material": "^6.1.6",
    "@reduxjs/toolkit": "^1.9.5",
    "axios": "^0.27.2",
    "framer-motion": "^10.12.16",
    "react": "^18.2.0",
    "react-chatbot-kit": "^2.1.2",
    "react-dom": "^18.2.0",
    "react-icons": "^4.9.0",
    "react-redux": "^8.0.7",
    "react-router-dom": "^6.12.0",
    "styled-components": "^6.0.0-rc.3"
  },
  "devDependencies": {
    "@types/react": "^18.0.37",
    "@types/react-dom": "^18.0.11",
    "@typescript-eslint/eslint-plugin": "^5.59.0",
    "@typescript-eslint/parser": "^5.59.0",
    "@vitejs/plugin-react-swc": "^3.0.0",
    "typescript": "^5.0.2",
    "vite": "^4.3.9"
  }
}
```

frontend/tsconfig.node.json

```
{
  "compilerOptions": {
    "composite": true,
    "skipLibCheck": true,
    "module": "ESNext",
    "moduleResolution": "bundler",
    "allowSyntheticDefaultImports": true
```

```
  },
  "include": ["vite.config.ts"]
}
```

frontend/vite.config.ts

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react-swc'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
  server: {
    host: true,
    port: 5173,
    strictPort: true,
    hmr: {
      protocol: 'ws',
      host: 'localhost',
      port: 5173,
    },
  },
})
```

frontend/tsconfig.json

```
{
  "compilerOptions": {
    "target": "ES2020",
    "useDefineForClassFields": true,
    "lib": ["ES2020", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "skipLibCheck": true,

    /* Bundler mode */
    "moduleResolution": "bundler",
    "allowImportingTsExtensions": true,
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react-jsx",
```

```
    /* Linting */
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noFallthroughCasesInSwitch": true
  },
  "include": ["src", "redux"],
  "references": [{ "path": "./tsconfig.node.json" }]
}
```

frontend/Dockerfile

```
FROM node:18-alpine

WORKDIR /app

COPY package.json .

RUN npm install

COPY . .

RUN npm run build

EXPOSE 5173

CMD [ "npm", "run", "dev" ]
```

frontend/bot/config.tsx

```
import { createChatBotMessage } from "react-chatbot-kit";
import YesNo from "./widgets/options/YesNo";
import ExerciseDropdown from "./widgets/options/ExerciseDropdown";
import IWidget from "react-chatbot-kit/build/src/interfaces/IWidget";
import IConfig from "react-chatbot-kit/build/src/interfaces/IConfig";
import beeHelper from "/bee_helper.svg";
import user from "/user.svg";

const config: IConfig = {
  botName: "Expert Bee",
```

```
  initialMessages: [
    createChatBotMessage(`Olá! Bem-vindo ao Especialista em Beecrowd! Sobre
    ↪  qual questão você gostaria de tirar dúvidas?`, {
      widget: "exerciseDropdown",
    }),
  ],
  customStyles: {
    botMessageBox: {
      backgroundColor: "#6A2483",
    },
    chatButton: {
      backgroundColor: "#6A2483",
    },
  },
  customComponents: {
    botAvatar: (props: any) => <img src={beeHelper} alt="bot" {...props} />,
    userAvatar: (props: any) => <img src={user} alt="bot" {...props} />
  },
  widgets: [
    {
      widgetName: "yesNo",
      widgetFunc: (props: any) => <YesNo {...props} />,
    },
    {
      widgetName: "exerciseDropdown",
      widgetFunc: (props: any) => <ExerciseDropdown {...props} />,
    },
  ] as IWidget[],
};

export default config;
```

frontend/bot/MessageParser.tsx

```
import React from "react";

const MessageParser = ({
  children,
  actions,
}: {
```

```
  children: any;
  actions: {
    handleFirstMessage: (questionNumber?: string) => void;
    handleUserInput: (answer: string) => void;
  };
}) => {

  const parse = (message: string) => {
    const parsedMessage = parseInt(message, 10);
    if (!isNaN(parsedMessage)) {
      actions.handleFirstMessage(message);
    } else {
      if (message == 'nao') message = 'não'
      actions.handleUserInput(message);
    }
  };

  return (
    <div>
      {React.Children.map(children, (child) =>
        React.cloneElement(child, {
          parse,
          actions,
        })
      )}
    </div>
  );
};

export default MessageParser;
```

frontend/bot/ActionProvider.tsx

```
import React from "react";
import { IMessageOptions } from
↪  "react-chatbot-kit/build/src/interfaces/IMessages";

const URL = import.meta.env.VITE_BACKEND_URL

const ActionProvider = ({
```

```
  createChatBotMessage,
  setState,
  children,
}: {
  createChatBotMessage: (
    message: string,
    options: IMessageOptions
  ) => {
    loading: boolean;
    widget?: string;
    delay?: number;
    payload?: any;
    message: string;
    type: string;
    id: number;
  };
  setState: any;
  children: any;
}) => {
  // const dispatch = useDispatch<AppDispatch>();

  const fetchData = async (url: string, body: URLSearchParams) => {
    try {
      const response = await fetch(url, {
        method: "POST",
        headers: { 'Content-Type':
        ↪  'application/x-www-form-urlencoded;charset=UTF-8' },
        credentials: 'include',
        body,
      });

      if (response.status !== 200) throw new Error("Erro ao consultar o
      ↪  backend.");

      return await response.json();
    } catch (error) {
      return { error: "Erro ao consultar o backend." };
    }
  };
```

```
const handleMessageResponse = async (data: any, additionalMessages: any[])
↪  => {
  const { question, result, error } = data;

  if (question) {
    /* Exemplo de atualização do messages-slice, caso precise no futuro
    dispatch(addQuestion(question));
    setTimeout(() => dispatch(startCount(question.length)), 5000);
    */
    const lines = question.split('\n');
    lines.forEach((line: string, index: number) => {
        if (line.trim() !== "") {
            const isLastLine = index === lines.length - 1;
            const messageOptions = isLastLine ? { widget: "yesNo" } : {};
            additionalMessages.push(createChatBotMessage(line,
            ↪  messageOptions));
        }
    });
  } else if (result || error) {
    const responseText = result || error || "Mensagem inválida recebida.";
    const lines = responseText.split('\n').filter((line: string) =>
    ↪  line.trim() !== "");

    lines.forEach((line: string) => {
      additionalMessages.push(createChatBotMessage(line, {}));
    });

    additionalMessages.push(createChatBotMessage("Gostaria de tirar
    ↪  dúvidas novamente?", { widget: "exerciseDropdown" }));
  }
};

const handleFirstMessage = async (questionNumber?: string) => {
  const body = new URLSearchParams({ questionNumber:
  ↪  questionNumber?.toString() || '' });
  const data = await fetchData(`${URL}/server`, body);
  let additionalMessages: {
    messages: {
```

```
        message: string;
        type: string;
        id: number;
        loading?: boolean;
        widget?: string | undefined;
        delay?: number | undefined;
        payload?: any;
      }[];
    }[] = [];

    await handleMessageResponse(data, additionalMessages);

    setState((prev: {
      messages: {
        message: string;
        type: string;
        id: number;
        loading?: boolean;
        widget?: string | undefined;
        delay?: number | undefined;
        payload?: any;
      }[];
    }) => ({
      ...prev,
      messages: [...prev.messages, ...additionalMessages],
    }));
  };

  const handleUserInput = async (answer: string) => {
    const body = new URLSearchParams({ answer: answer.toLowerCase() });
    const data = await fetchData(`${URL}/server`, body);
    let additionalMessages: {
      messages: {
        message: string;
        type: string;
        id: number;
        loading?: boolean;
        widget?: string | undefined;
        delay?: number | undefined;
```

```
      payload?: any;
    }[];
  }[] = [];

  await handleMessageResponse(data, additionalMessages);

  setState((prev: {
    messages: {
      message: string;
      type: string;
      id: number;
      loading?: boolean;
      widget?: string | undefined;
      delay?: number | undefined;
      payload?: any;
    }[];
  }) => ({
    ...prev,
    messages: [
      ...prev.messages,
      ...additionalMessages
    ],
  }));
};

return (
  <div>
    {React.Children.map(children, (child) => {
      return React.cloneElement(child, {
        actions: {
          handleFirstMessage,
          handleYes: () => handleUserInput("sim"),
          handleNo: () => handleUserInput("não"),
          handleUserInput,
        },
      });
    })}
  </div>
);
```

```
};

export default ActionProvider;
```

frontend/bot/widgets/options/YesNo.tsx

```
import React from "react";
import { styled } from "styled-components";

const StyledButton = styled.button`
  padding: 0.5rem 1rem;
  border: 1px solid rgb(var(--primary-color));
  color: rgb(var(--primary-color));
  border-radius: 5rem;
  transition: 0.15s;
  &:active {
    transform: scale(0.95);
  }
`;

const GotIt: React.FC<any> = (props) => {
  const options = [
    {
      name: "Sim",
      handler: props.actionProvider.handleYes,
      id: 1,
    },
    {
      name: "Não",
      handler: props.actionProvider.handleNo,
      id: 2,
    },
  ];
  return (
    <div className="options-container">
      {options.map((option) => {
        return (
          <StyledButton
            key={option.id}
            className="option-button"
```

```
            onClick={option.handler}
          >
            {option.name}
          </StyledButton>
        );
      })}
    </div>
  );
};

export default GotIt;
```

frontend/bot/widgets/options/ExerciseDropdown.tsx

```
import React, { useEffect, useState } from "react";
import { styled } from "styled-components";

const StyledSelect = styled.select`
  position: relative;
  padding: 0.5rem 1rem;
  border-radius: 5rem;
  appearance: none;
`;

const URL = import.meta.env.VITE_BACKEND_URL

const ExerciseDropdown: React.FC<any> = (props) => {
  const [questions, setQuestions] = useState<number[]>([]);

  useEffect(() => {
    const fetchExercises = async () => {
      try {
        const response = await fetch(`${URL}/questions`); // Atualize o URL
        ↪  conforme necessário
        const data = await response.json();
        setQuestions(data.sort());
      } catch (error) {
        console.error("Erro ao buscar exercícios:", error);
      }
    };
```

```
    fetchExercises();
  }, []);

  const handleExercise = (e: React.ChangeEvent<HTMLSelectElement>) => {
    props.actionProvider.handleFirstMessage(e.target.value);
  };

  return (
    <StyledSelect onChange={handleExercise} title="Selecione aqui">
      <option>Selecione aqui ou digite o número da questão</option>
      {Array.from(questions).map((exercise) => (
        <option key={exercise} value={exercise}>
          {exercise}
        </option>
      ))}
    </StyledSelect>
  );
};

export default ExerciseDropdown;
```

frontend/src/vite-env.d.ts

```
/// <reference types="vite/client" />
```

frontend/src/main.tsx

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import { BrowserRouter as Router } from "react-router-dom";
import App from "./App.tsx";
import { ReduxProvider } from "../redux/provider.tsx";

createRoot(document.getElementById("root") as HTMLElement).render(
  <StrictMode>
    <Router>
      <ReduxProvider>
        <App />
      </ReduxProvider>
    </Router>
```

```
  </StrictMode>
);
```

frontend/src/App.tsx

```
import { Suspense, lazy, useState } from "react";
import { createGlobalStyle } from "styled-components";
import { AnimatePresence } from "framer-motion";
import { Route, Routes, useLocation, useNavigate } from "react-router-dom";
import Details from "./page/Details";

const Chat = lazy(() => import("./page/Chat"));
const NotFound = lazy(() => import("./page/404"));

const GlobalStyle = createGlobalStyle`
  :root{
    --light-color: 241 241 241;
    --dark-color: 52 17 64;
    --primary-color: 106 36 131;
    --secondary-color: 169 87 190;
    --tertiary-color: 230, 213, 14;
    --quaternary-color: 245 243 213;
    --success-color: 3 179 10;
    --like-color: 16 110 190;
    --font-poppins: 'Poppins', system-ui, sans-serif;
    scroll-behavior: smooth;
  }
  ::-webkit-scrollbar {
    width: 0;
  }
  html {
    color-scheme: light;
  }
  html, body {
    overflow-x: hidden;
  }
  * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
```

```
  font-family: var(--font-poppins);
  outline: none;
}
body {
  position: relative;
  background-color: rgb(var(--light-color));
  color: rgb(var(--dark-color));
  min-height: 100vh;
  display: grid;
  place-items: center;
}
.container {
  margin-inline: auto;
  width: min(90%, 70rem);
}
a {
  text-decoration: none;
  color: inherit;
  transition: 0.15s;
}
input{
  background-color: transparent;
}
button {
  cursor: pointer;
  border: none;
  background-color: transparent;
  user-select: none;
}
& span.loader {
  margin-bottom: 2rem;
  width: 1rem;
  height: 1rem;
  border: 2px solid #FFF;
  border-bottom-color: rgb(var(--dark-color));
  border-radius: 50%;
  display: inline-block;
  animation: rotation 1s linear infinite;
}
```

```
  @keyframes rotation {
    0% {
        transform: rotate(0deg);
    }
    100% {
        transform: rotate(360deg);
    }
  }
  #root {
    width: 100%;
    height: 100%;
  }
  & .details-button {
    position: fixed;
    bottom: 1rem;
    left: 1rem;
    z-index: 1000;
    & > h1 {
      font-size: 1rem;
    }
    & > button {
      color:rgb(var(--dark-color));
      background-color:rgb(var(--light-color));
      margin-top: 1rem;
      padding: 0.5rem 2rem;
      font-size: 1.1rem;
      border-radius: 5rem;
      border: 1px solid rgb(var(--dark-color));
      transition: 0.15s;
      &:active {
        transform: scale(0.95);
        border-color: rgb(var(--tertiary-color));
      }
    }
  }
`;

const App = () => {
  const location = useLocation();
```

```
const navigate = useNavigate();

const [isDetails, setIsDetails] = useState(location.pathname ===
↪  "/details");

const handleToggleNavigation = () => {
  if (isDetails) {
    navigate("/");
  } else {
    navigate("/details");
  }
  setIsDetails(!isDetails);
};

return (
  <>
    <GlobalStyle />
    <div className="details-button">
      <button onClick={handleToggleNavigation}>
        {isDetails
          ? "Voltar para a página inicial"
          : "Clique aqui para ler sobre os tipos de resposta do Beecrowd!"}
      </button>
    </div>
    <Suspense
      fallback={
        <div
          className="loader-container"
          style={{
            height: "100vh",
            width: "100vw",
            display: "grid",
            placeItems: "center",
          }}
        >
          <span className="loader"></span>
        </div>
      }
    >
```

```
        <AnimatePresence mode="wait">
          <Routes location={location} key={location.pathname}>
            <Route path="/" element={<Chat />} />
            <Route path="/details" element={<Details />} />
            <Route path="*" element={<NotFound />} />
          </Routes>
        </AnimatePresence>
      </Suspense>
    </>
  );
};

export default App;
```

frontend/src/page/Details.tsx

```
import "react-chatbot-kit/build/main.css";
import { styled } from "styled-components";
import Accordion from '@mui/material/Accordion';
import AccordionActions from '@mui/material/AccordionActions';
import AccordionSummary from '@mui/material/AccordionSummary';
import AccordionDetails from '@mui/material/AccordionDetails';
import ExpandMoreIcon from '@mui/icons-material/ExpandMore';

const StyledChat = styled.div`
  display: flex;
  align-items: start;
  width: 100%;
  height: 100vh;

  .accordion-container {
    margin: 1rem;
  }
`;

const Details = () => {
  return (
    <StyledChat>
      <div className="accordion-container">
        <Accordion>
```

```
        <AccordionSummary
        expandIcon={<ExpandMoreIcon />}
        aria-controls="panel1-content"
        id="panel1-header"
        >
        Resposta errada/Wrong answer
        </AccordionSummary>
        <AccordionDetails>
        Se o URI dizer "resposta errada/wrong answer" você deve criar
        ↪  novos casos de teste
        para tentar adivinhar onde teu programa está falhando. Note que
        ↪  no enunciado são
        dados apenas poucos casos de teste, mas o URI usa muitos outros
        ↪  casos de teste para
        saber se teu programa funciona ou não. Teu programa pode
        ↪  funcionar para os exemplos do
        enunciado, mas pode falhar para outras situações. Utilize o
        ↪  "uDebug" que está no menu
        do URI na página do enunciado do problema para encontrar
        ↪  exemplos de outros casos de
        teste. Teste tua solução com todos esses novos casos de teste e
        ↪  verifique se as respostas
        dadas pelo teu programa estão corretas ou não. Caso teu programa
        ↪  não está dando resposta
        correta, estude esses casos de teste e volte para o passo 4 para
        ↪  tentar simular no papel
        a tua estratégia de solução para o problema. Caso você conseguir
        ↪  chegar no resultado usando
        o papel, então você saberá que escreveu algo errado no programa
        ↪  Python, portanto você
        deve rever o programa em Python para ajustá-lo. Você também pode
        ↪  utilizar o Thonny para
        debugar (executar o programa passo-a-passo) e assim tentar
        ↪  identificar onde está o erro
        para o caso de teste que não está funcionando.
        </AccordionDetails>
    </Accordion>
    <Accordion>
        <AccordionSummary
```

```
        expandIcon={<ExpandMoreIcon />}
        aria-controls="panel2-content"
        id="panel2-header"
        >
        Tempo limite excedido/Time limit exceeded
        </AccordionSummary>
        <AccordionDetails>
        Significa que teu programa não está parando ou é muito lento.
        ↪  Confira se você tem
        estruturas de repetição e se elas tem alguma condição de parada.
        ↪  Ex: se você tiver um
        while (i &lt; 100), confira se este i chegará mesmo em 100,
        ↪  incrementando, por exemplo, de
        um em um. Se a entrada do problema dizer que termina com EOF,
        ↪  confira nos slides da
        disciplina como terminar a entrada de dados por EOF, no tema de
        ↪  "Estruturas de Repetição".
        </AccordionDetails>
    </Accordion>
    <Accordion>
        <AccordionSummary
        expandIcon={<ExpandMoreIcon />}
        aria-controls="panel3-content"
        id="panel3-header"
        >
        Erro de compilação/Compile error
        </AccordionSummary>
        <AccordionDetails>
        Confira a linguagem que você está submetendo no URI. Sempre
        ↪  escolha qualquer versão do
        Python 3.x (tanto faz se x for 4, 5, 6, 7 8). Além disso,
        ↪  confira se em algum momento do
        código fonte você está usando "import". Se sim, tente resolver
        ↪  seu problema sem "import".
        </AccordionDetails>
        <AccordionActions>
        </AccordionActions>
    </Accordion>
    <Accordion>
```

```
            <AccordionSummary
            expandIcon={<ExpandMoreIcon />}
            aria-controls="panel3-content"
            id="panel3-header"
            >
            Erro em tempo de execução/Run-time error
            </AccordionSummary>
            <AccordionDetails>
            Significa que seu programa está gerando um erro quando está
             ↪  sendo executado pelo URI.
            Confira se você tem algum tipo de divisão no programa, seja por /
             ↪  ou % ou //. Se tiver,
            veja se pode ocorrer do denominador ser zero (note que não
             ↪  existe divisão por 0 e seu
            programa pode gerar erro por isso). Se não tiver divisão, veja
             ↪  se você está trabalhando
            com algum tipo de estrutura (vetor, matriz, lista), e então
             ↪  confira se você pode estar
            acessando uma posição inválida desta estrutura. Por exemplo, se
             ↪  o vetor tem 10 posições
            e você está tentando acessar a posição 15, obviamente irá gerar
             ↪  um erro. Outro causador
            desse erro é na leitura dos dados de entrada. Por exemplo, se os
             ↪  dados de entrada são
            fornecidos numa mesma linha separados por espaços, deve-se
             ↪  atentar para a forma correta
            de fazer a leitura desses dados.
            </AccordionDetails>
            <AccordionActions>
            </AccordionActions>
        </Accordion>
        <Accordion>
            <AccordionSummary
            expandIcon={<ExpandMoreIcon />}
            aria-controls="panel3-content"
            id="panel3-header"
            >
            Erro de apresentação/Presentation error
            </AccordionSummary>
```

```
            <AccordionDetails>
            Esse erro ocorre normalmente por falta de quebras de linhas ou
            ↪  espaços, ou excesso de
            quebras de linhas ou espaços. Assim, por exemplo, se o enunciado
            ↪  pede para imprimir uma
            linha em branco entre os casos de testes, você deve imprimir uma
            ↪  linha em branco entre
            os casos de testes. Um caso comum em que esse erro também ocorre
            ↪  é ao imprimir uma lista
            de valores numa mesma linha, e então há um espaço sobrando após
            ↪  o último valor impresso
            (você deve cuidar para que ele não seja impresso).
            </AccordionDetails>
            <AccordionActions>
            </AccordionActions>
        </Accordion>
      </div>
    </StyledChat>
  );
};

export default Details;
```

frontend/src/page/404.tsx

```
const NotFound = () => {
  return <div>404</div>;
};

export default NotFound;
```

frontend/src/page/Chat.tsx

```
import { useState } from "react";
import Chatbot from "react-chatbot-kit";
import "react-chatbot-kit/build/main.css";
import { styled } from "styled-components";
import config from "../../bot/config";
import ActionProvider from "../../bot/ActionProvider";
import MessageParser from "../../bot/MessageParser";
// import { AppDispatch, useAppSelector } from "../../redux/store";
```

```
// import { useDispatch } from "react-redux";
// import { decrementCount } from "../../redux/features/messages-slice";
// import { useNavigate } from "react-router-dom";

const StyledChat = styled.div`
  position: relative;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  width: 100%;
  height: 95vh;


  .react-chatbot-kit-chat-container {
    width: 35rem !important;
    height: 40rem;
  }
  .react-chatbot-kit-chat-input-container {
    width: 34.9rem !important;
  }

  .react-chatbot-kit-chat-message-container {
    height: 35.5rem;
  }

  .react-chatbot-kit-chat-bot-message-container {
    & > img {
      margin-right: 0.8rem;
    }
  }

  .react-chatbot-kit-user-chat-message-container {
    & > img {
      margin-left: 0.8rem;
    }
  }

  & .intro {
```

```
    height: 9rem;
    text-align: center;
    & > h1 {
      font-size: 3.5rem;
    }
    & > button {
      color:rgb(var(--dark-color));
      margin-top: 1rem;
      padding: 0.75rem 2rem;
      font-size: 1.5rem;
      border-radius: 5rem;
      border: 1px solid rgb(var(--dark-color));
      transition: 0.15s;
      &:active {
        transform: scale(0.95);
        border-color: rgb(var(--tertiary-color));
      }
    }
  }
  & > div:not(.intro) {
    & .react-chatbot-kit-chat-inner-container {
      border: 1px solid rgb(var(--dark-color) / 0.25);
      border-radius: 1rem;
      overflow: hidden;
      width: 35rem;
      height: 40rem;
      & .react-chatbot-kit-chat-bot-message {
        width: 20rem;
        margin-left: 0 !important;
        border-bottom-left-radius: 0;
        & .react-chatbot-kit-chat-bot-message-arrow {
          border-width: 0;
        }
      }
    }
  }
`;

const Chat = () => {
```

```
  const [isOpened, setIsOpened] = useState<boolean>(false)
  /* Exemplo de uso do messages-slice, caso precise no futuro
  const navigate = useNavigate();
  const count = useAppSelector((state) => state.messageReducer.count);
  const dispatch = useDispatch<AppDispatch>();
  useEffect(() => {
    const interval = setInterval(() => {
      if (count > 0) {
        dispatch(decrementCount());
      } else if (count === 0) {
        navigate("/success");
      }
    }, 1000);
    return () => {
      clearInterval(interval);
    };
  }, [count, dispatch, navigate]);
  */
  return (
    <StyledChat>
      <div className="intro">
        <h2>Aqui você pode tirar dúvidas sobre algumas questões do
        ↪  Beecrowd!</h2>
        <button onClick={() => setIsOpened((prev) => !prev)}>
          Comece agora!
        </button>
      </div>
      {isOpened && (
        <Chatbot
          config={config}
          messageParser={MessageParser}
          actionProvider={ActionProvider}
        />
      )}
    </StyledChat>
  );
};

export default Chat;
```

frontend/public/bee_helper.svg

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg version="1.0" xmlns="http://www.w3.org/2000/svg"
 width="41" height="41" viewBox="0 0 500.000000 500.000000"
 preserveAspectRatio="xMidYMid meet">

<g transform="translate(0.000000,500.000000) scale(0.100000,-0.100000)"
fill="#e6d50e" stroke="none">
<path d="M1812 4183 c-12 -48 -22 -91 -22 -94 0 -4 27 -16 59 -28 73 -26 161
-85 220 -145 l44 -45 -37 -43 c-70 -82 -126 -223 -126 -320 0 -27 -4 -48 -8
-48 -5 0 -67 22 -138 49 -363 137 -665 200 -954 201 -230 0 -412 -40 -567
-126 -255 -141 -342 -410 -220 -678 69 -150 208 -291 375 -382 l54 -29 4 -60
c18 -262 263 -460 594 -481 173 -11 359 33 512 121 43 25 81 45 85 45 5 0 9
-78 9 -173 4 -383 108 -709 306 -953 94 -117 241 -216 363 -245 64 -14 196
-14 260 0 122 29 269 128 363 244 198 245 302 572 306 954 0 95 4 173 9 173 4
0 42 -20 85 -45 153 -88 339 -132 512 -121 330 21 576 220 594 479 l4 62 54
29 c167 91 306 232 375 382 38 84 53 150 53 241 0 234 -143 404 -424 502 -122
43 -245 61 -416 61 -289 -1 -591 -64 -954 -201 -71 -27 -133 -49 -138 -49 -4
0 -8 21 -8 48 0 97 -56 238 -126 320 l-37 43 44 45 c59 60 147 119 220 145 32
12 59 24 59 28 0 3 -10 46 -22 94 l-23 88 -35 -6 c-98 -19 -241 -107 -343
-213 l-68 -71 -42 15 c-29 10 -82 15 -172 15 -90 0 -143 -5 -172 -15 l-42 -15
-68 71 c-102 106 -245 194 -343 213 l-35 6 -23 -88z m-674 -724 c148 -25 286
-60 452 -116 204 -68 202 -67 74 -87 -510 -82 -869 -249 -1047 -485 l-44 -59
-59 32 c-78 41 -191 155 -231 231 -86 166 -56 302 87 394 75 48 225 98 332
110 110 12 296 4 436 -20z m3133 20 c190 -19 372 -98 438 -189 87 -121 47
-296 -104 -445 -38 -38 -97 -84 -129 -101 l-59 -32 -44 59 c-178 236 -537 403
-1047 485 -128 20 -130 19 74 87 220 74 428 121 599 137 128 11 159 11 272 -1z
m-2230 -442 c-5 -13 -33 -55 -63 -95 -61 -81 -141 -236 -184 -357 -26 -72 -37
-88 -109 -160 -151 -152 -324 -234 -510 -243 -206 -9 -367 64 -431 193 -41 86
-27 150 59 263 163 214 570 369 1092 416 77 7 143 11 147 9 4 -2 3 -14 -1 -26z
m1324 -18 c392 -69 690 -208 822 -381 86 -113 100 -177 59 -263 -64 -129 -225
-202 -431 -193 -186 9 -359 91 -510 243 -72 72 -83 88 -109 160 -40 113 -119
268 -174 343 -54 72 -83 122 -77 132 6 10 286 -18 420 -41z m-630 -175 c110
-19 133 -32 179 -103 57 -85 51 -99 -58 -134 -234 -74 -488 -74 -722 0 -78 25
-89 32 -92 54 -4 31 59 130 97 152 25 15 157 41 256 51 67 7 245 -4 340 -20z
m-779 -948 c50 -24 168 -62 274 -88 37 -9 124 -13 265 -13 141 0 228 4 265 13
106 26 224 64 274 88 27 13 54 24 59 24 18 0 -24 -303 -49 -352 -35 -69 -328
```

```
-148 -549 -148 -221 0 -514 79 -549 148 -25 49 -67 352 -49 352 5 0 32 -11 59
-24z"/>
</g>
</svg>
```

frontend/public/user.svg

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg version="1.0" xmlns="http://www.w3.org/2000/svg"
 width="41" height="41" viewBox="0 0 256.000000 256.000000"
 preserveAspectRatio="xMidYMid meet">

<g transform="translate(0.000000,256.000000) scale(0.100000,-0.100000)"
fill="#e6d50e" stroke="none">
<path d="M1124 2545 c-214 -46 -402 -229 -459 -445 -20 -73 -19 -236 1 -314
42 -163 172 -321 325 -394 106 -51 182 -66 307 -60 163 8 286 64 403 183 185
188 229 465 112 704 -52 105 -169 223 -273 274 -128 63 -279 82 -416 52z"/>
<path d="M630 1301 c-136 -44 -229 -123 -297 -256 -99 -191 -149 -525 -108
-719 34 -161 153 -275 324 -312 97 -20 1365 -20 1462 0 88 19 141 47 207 110
96 91 125 174 124 356 -3 374 -119 665 -308 776 -61 36 -159 64 -222 64 -56 0
-101 -18 -180 -72 -66 -45 -180 -95 -249 -108 -83 -16 -202 -8 -283 19 -61 21
-90 36 -234 125 -52 31 -68 36 -120 35 -34 0 -86 -8 -116 -18z"/>
</g>
</svg>
```

frontend/redux/provider.tsx

```
import { store } from "./store";
import { Provider } from "react-redux";

export const ReduxProvider = ({ children }: { children: React.ReactNode })
↪  => {
  return <Provider store={store}>{children}</Provider>;
};
```

frontend/redux/store.ts

```
import { configureStore } from "@reduxjs/toolkit";
import messageReducer from "./features/messages-slice";
import { TypedUseSelectorHook, useSelector } from "react-redux";
```

```
export const store = configureStore({
  reducer: {
    messageReducer,
  },
});

export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;

export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector;
```

frontend/redux/features/messages-slice.ts

```
import { createSlice, PayloadAction } from "@reduxjs/toolkit";

type MessageState = {
  count: number;
  questions: string[];
};

const initialState = {
  count: -1,
  questions: [],
} as MessageState;

export const messageSlice = createSlice({
  name: "Messages",
  initialState,
  reducers: {
    addQuestion: (state, action: PayloadAction<string>) => {
      state.questions.push(action.payload);
    },
    startCount: (state, action: PayloadAction<number>) => {
      state.count = action.payload;
    },
    decrementCount: (state) => {
      state.count -= 1;
    },
  },
```

```
});

export const { addQuestion, startCount, decrementCount } =
  messageSlice.actions;
export default messageSlice.reducer;
```