UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

Rodrigo Joaquim Nunes

Sistema educacional Web para ensino de programação orientada a objetos com Python

Florianópolis - SC
2024 / 2

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

Sistema educacional Web para ensino de programação orientada à objetos com Python

Rodrigo Joaquim Nunes

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Florianópolis - SC
2024 / 2

Rodrigo Joaquim Nunes

Sistema educacional Web para ensino de programação orientada à objetos com Python

Trabalho de conclusão de curso apresentado como parte dos requisitos para a obtenção do grau de Bacharel em curso de Sistemas de Informação

Florianópolis, 18 de novembro de 2024.

Banca examinadora

_____

Prof. Dr. Elder Rizzon Santos
Orientador

_____

Prof. Dr. Alexandre Gonçalves Silva
Instituição Universidade Federal de Santa Catarina

_____

Prof. Maicon Rafael Zatelli
Instituição Universidade Federal de Santa Catarina

Para minha esposa, que sempre me apoia.
À minha mãe, que buscou que eu realizasse meus sonhos,
E ao professor Elder Rizzon Santos pela paciência de me orientar.

AGRADECIMENTOS

Primeiramente, quero agradecer a minha mãe, que por toda a minha vida me incentivou, buscou me fazer feliz e priorizou os meus estudos, enfrentando qualquer barreira que aparecesse.

À minha esposa Wanessa, minha apoiadora incondicional da vida, me fazendo companhia em todas as noites em claro durante a produção dos trabalhos da graduação, me confortando nos momento difíceis, me acalmando mesmo quando eu estava prestes a desmoronar, sempre acreditou no meu potencial. Enfrentou o período de muitas incertezas que foi pandemia ao meu lado. Te amo.

Aos meus filhos Gael e Analice, que me motivam sempre a ser melhor e a tornar o mundo deles um lugar melhor.

Ao meu orientador, Elder Rizzon Santos, por ter aceitado quando o convidei para ser meu orientador, pela paciência de me explicar cada passo do desenvolvimento do trabalho, por entender a minha rotina fora da universidade e mesmo assim disponibilizar do seu tempo para me orientar.

Aos professores da minha banca avaliadora, Alexandre Gonçalves Silva e Maicon Rafael Zatelli, por terem aceitado meu convite.

Aos professores Carina e Cislaghi, por terem me auxiliado com as minhas dúvidas sobre os procedimentos do TCC.

E, por fim, mas não menos importante, à UFSC, por proporcionar um ensino de qualidade e possibilitar realizar um dos meus sonhos: me formar em uma universidade federal.

RESUMO

Este trabalho apresenta o desenvolvimento do GAPy, um sistema educacional Web para o ensino de Programação Orientada a Objetos (POO) com Python. Utilizando tecnologias modernas como WebAssembly e Pyodide, o GAPy permite a execução de código diretamente no navegador, eliminando a necessidade de instalações locais. Inspirado em ferramentas como BlueJ e Greenfoot, o sistema combina interatividade e elementos gráficos com uma abordagem pedagógica voltada para iniciantes. Ele permite a criação, manipulação e visualização de classes e objetos, facilitando a compreensão dos conceitos fundamentais de POO. Comparações qualitativas indicam que o GAPy apresenta vantagens significativas em termos de simplicidade e acessibilidade, sendo uma ferramenta promissora para o ensino de POO no contexto educacional.

Palavras-chave: Ferramentas de Ensino, Ensino de Programação; Programação Orientada a Objetos; WebAssembly; Python

ABSTRACT

This work introduces GAPy, a web-based educational system for teaching Object-Oriented Programming (OOP) with Python. Leveraging modern technologies such as WebAssembly and Pyodide, GAPy enables direct code execution in the browser, eliminating the need for local installations. Inspired by tools like BlueJ and Greenfoot, the system combines interactivity and graphical elements with a pedagogical approach tailored to beginners. It facilitates the creation, manipulation, and visualization of classes and objects, enhancing the understanding of fundamental OOP concepts. Qualitative comparisons showed that GAPy offers significant advantages in simplicity and accessibility, making it a promising tool for OOP education in academic settings.

Keywords: Learning Tools, Programming Education, Object-Oriented Programming, WebAssembly, Python

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 Introdução

Atualmente, o ensino de disciplinas de programação cresce constantemente desde a educação básica até o ensino superior. Nos cursos de graduação tecnológicas, como as engenharias, Ciência da Computação e Sistemas de Informação é comum que possua mais contato com disciplinas com foco em programação. Contudo, outras áreas também que abordam programação em disciplinas introdutórias. Porém várias linguagens de programação são utilizadas nos diversos contextos existentes, seja para o ensino ou para desenvolvimento de projetos (BELANI, 2020).

No cursos de graduação que possuem disciplinas de programação em sua grade curricular há um nivelamento no conhecimento desenvolvimento de software com linguagens de programação, partindo dos conceitos mais básicos até os mais avançados. Porém há uma grande dificuldade de aprendizado dos conceitos iniciais de programação, tornando a experiência de aprendizado negativa (GOMES, HENRIQUES & MENDES, 2008).

O ensino de programação está associado a um paradigma de programação e a uma linguagem. O paradigma de programação é o conjunto de regras de codificação que uma determinada linguagem deve seguir, como por exemplo: paradigma de programação orientado a objetos, que busca abstrair os elementos do mundo real com o maior fidelidade que o problema exija (FARINELLI, 2007).

O paradigma de orientação a objetos pode ser implementado por diversas linguagens de programação, dentre elas o Python é apontado como uma tendên por ser simples de aprender, possui uma participação ativa da comunidade e recomendado para fins pedagógicos no ensino superior (YADAV & DEBELLO, 2019) (IEEE SPECTRUM, 2024).

O desenvolvimento de aplicações *Web* têm substituído os sistemas *on-premise (*que são instalados e executados nos equipamentos da própria organização*)*, ao qual as empresas alocavam suas aplicações em estruturas proprietárias e possuíam a total responsabilidade de manutenção de todos os hardwares e software que envolvem a aplicação, já as aplicações Web fazem uso de uma estrutura terceirizada que facilita o manutenibilidade da aplicação.

Com o passar do tempo as equipes de desenvolvimento dos maiores navegadores Web perceberam a baixa dos níveis de segurança e desempenho, logo, as equipes de desenvolvedores dos maiores browsers se reuniram e desenvolveram o WebAssembly, que é uma abstração de hardware desacoplado de uma linguagem ou plataforma e permitindo a eficiência e a segurança para o desenvolvimento de aplicações Web (HAAS et al., 2017).

Existem diversas aplicações com propósitos gerais disponíveis na Web, como editores de texto, planilhas e formulários (GOOGLE, 2022). Porém há um crescimento na necessidade de desenvolver sistemas especializados para melhor atender a problemas específicos. Por isso, o intuito deste trabalho é desenvolver um sistema educacional em WebAssembly para facilitar o ensino de programação

orientada a objetos com a linguagem Python a estudantes universitários (iniciantes em programação).

2 Objetivos

2.1 Objetivo Geral

O objetivo principal deste trabalho é desenvolver um sistema educacional em ambiente Web com ênfase no ensino e aprendizado dos conceitos fundamentais da programação orientada a objeto.

O sistema utilizará WebAssembly para interpretação do código diretamente no navegador e fará uso da linguagem de programação Python.

2.2 Objetivos Específicos

- Analisar o estado da arte em ensino e aprendizagem de programação, ambientes de aprendizagem de programação e aplicabilidade de WebAssembly
- Elencar abordagens de ensino mais adequada para ensino de programação orientada a objetos em Python
- Desenvolver protótipo de sistema educacional com WebAssembly para Python
- Aplicar análise qualitativa comparativa ferramenta desenvolvida perante BlueJ e Thonny

3 Fundamentação Teórica

Este capítulo abrange conceitos fundamentais para o desenvolvimento deste trabalho. Na seção 3.1 introduz o tema referente ao ensino e aprendizagem de programação. Na seção 3.2 aborda as características principais dos ambientes de aprendizagem de programação e na seção 3.3 abrange as aplicabilidade de WebAssembly.

3.1 Ensino e aprendizagem de programação

A programação está cada vez mais presente no dia-a-dia na era digital, é necessário fomentar a consciência computacional sobre a programação de computadores, principalmente aos estudantes de disciplinas introdutórias de programação para estejam aptos para enfrentar os desafios em disciplinas mais avançadas ou em ambientes de trabalho.

O ensino de programação deve focar na transmissão de conhecimento técnico, o desenvolvimento de habilidades analíticas e de resolução de problemas (Papert, 1980). Cada linguagem de programação deve ser utilizada através de um paradigma de programação, o paradigma de programação orientada a objetos (POO) permite a modularização e reuso das implementações do software que agrega à metodologias de ensino com aprendizagem ativa e construtivista (Johnson et. al., 2016).

Porém o ensino de POO pode apresentar alguns desafios, principalmente aos alunos iniciantes em programação, como a abstração do elementos do mundo real para as classes de programação e a complexidade de transcrição para a linguagem de programação que podem ser mitigados com a utilização de contextos mais próximo do dia-a-dia desses estudantes (Hagan et al., 2020).

O feedback e a avaliação contínua é uma metodologia efetiva de ensino de programação, que por meio de avaliações formativas automatizadas ou revisões em pares e feedbacks pontuais auxiliam os estudantes a desenvolverem suas habilidades de programação (Hattie & Timperley, 2007).

A aprendizagem baseada em projetos também é uma metodologia efetiva, porém visa ensinar através da criação de jogos ou aplicativos para aplicar os conhecimentos de programação com o uso de POO (Kafai & Resnick, 1996)

3.2 Ambientes de aprendizagem de programação

Os ambientes de aprendizagem de programação são fundamentais para o ensino computacional, pois devem permitir que os alunos adquiram conhecimento de forma facilitada por meio de recursos pedagógicos e de software que tornam possível o desenvolvimento das habilidades dos estudantes nas linguagens de programação fomentando também o senso crítico e analítico dos estudantes, tais ambientes podem incluir desde IDE tradicionais até plataformas baseada no web Johnson (2018).

Essas ferramentas frequentemente incluem recursos para ajudar os instrutores no monitoramento do progresso dos alunos e identificar áreas que necessitam de maior atenção, principalmente para o alunos que estão tendo seu primeiro contato com a programação (Brown et al., 2014)

Com foco nos alunos iniciantes, os ambientes devem cumprir os seguintes critérios:

- Facilidade de uso: A interface deve ser clara e intuitiva, para minimizar a curva de aprendizado associada ao uso da ferramenta (Kölling, 2010).
- Recursos Didáticos: Deve incluir tutoriais, exemplos e documentação que facilitam a compreensão dos conceitos (Kölling et al., 2003)..
- Interatividade: Capacidade de interação em tempo real com objetos e classes, facilitando a compreensão da relação entre estrutura de código e comportamento esperado (Henriksen & Kölling, 2004).
- Feedback Imediato: Ferramentas que fornecem feedback imediato sobre erros de sintaxe e lógica ajudam os alunos a aprender de forma mais eficaz (Jadud, 2006).
- Suporte a Conceitos Fundamentais: A ferramenta deve promover a prática de princípios básicos como encapsulamento, herança e polimorfismo, fundamentais na POO (Kölling et al., 2003).
- Suporte a Metodologias de Avaliação Efetivas: Permite a implementação de métodos de avaliação, como a avaliação baseada em projetos, para melhor compreensão da absorção de conhecimento POO dos alunos (Bennedsen & Caspersen, 2007).

3.3 Aplicabilidade de WebAssembly

WebAssembly (Wasm) é uma linguagem de baixo nível que permite a execução de código compilado diretamente nos navegadores, com desempenho próximo ao código nativo. Utiliza um formato binário compacto e é projetado para ser seguro, executando em um ambiente isolado dentro do navegador. (WebAssembly.org).

Outras iniciativas já haviam tentado utilizar código binário nos navegadores, porém sem sucesso, como ActiveX, Silverlight, Google Native Client (NaCl) e Adobe Flash, e tais iniciativas tinham problemas de compatibilidade com alguns navegadores, de segurança ou lentidão (Haas et. al., 2017).

A Wasm é utilizada em aplicações Web que necessitam executar operações que requerem rapidez e eficiência, aplicações buscam desempenho gráfico como jogos e simulação, também utilizado no meio científico, financeiro e de engenharia para computação paralela (Garcia, 2022). Sua principal finalidade é a compilação do código-fonte de outras linguagens, como C, C++ e Python (Haas et. al., 2017).

As vantagens de utilizar Wasm é de uma execução mais rápida comparada ao JavaScript, especialmente para aplicações intensivas em computação (Jones & Brown, 2020), sua portabilidade permite que código escrito em diversas linguagens seja compilado para um formato binário comum, executável em diferentes

plataformas (Lee et al., 2021) e reduz o risco de vulnerabilidades de segurança, como explorações de buffer, devido ao seu modelo de execução em ambiente de teste isolado (Doe, 2018).

Porém, a sua integração com outras aplicações pode ser complexa, exigindo um maior esforço de novas ferramentas e processos de desenvolvimento (Johnson, 2022). Outro ponto é a sua depuração limitada pois as ferramentas ainda estão em estágios iniciais de desenvolvimento, o que pode dificultar a identificação e correção de bugs (Black, 2017) e tamanho do código binário gerado de aplicações pode ser maior do que em JavaScript (White & Green, 2023).

4 Trabalhos relacionados

Foram realizadas pesquisas, através das ferramentas Google Scholar, IEEE e DPLP, a fim de encontrar trabalhos sobre ensino e aprendizagem de programação e ambientes de aprendizado de programação.

A primeira subseção aborda a revisão sistemática da literatura brasileira sobre o ensino e aprendizado de programação. As subseções seguintes apresentam as ferramentas de aprendizado BlueJ, Greenfoot, Thonny, jGrasp que têm relação direta com a proposta desse trabalho, e por fim, é apresentada uma análise comparativa sobre o tema dos trabalhos citados nesta seção.

4.1 Ensino e aprendizagem de introdução à programação

Medeiros, Falcão e Ramalho (2020) evidenciaram as dificuldades enfrentadas por alunos e professores nas disciplinas introdutórias de programação presentes nos cursos de graduação tecnológica por possuírem temas fundamentais para a compreensão de disciplinas mais complexas. Este trabalho é baseado em outra produção acadêmica, que apresenta uma revisão sistemática da literatura (RSL) internacional referente ao ensino e aprendizagem de introdução à programação no ensino superior.

Os autores ressaltam a necessidade da criação de uma RSL para o contexto brasileiro por meio deste artigo, para melhor contextualizar a RSL nas possíveis dificuldades, abordagens pedagógicas e aspectos sociais específicos do país, fornecendo uma visão ampla e uma melhor compreensão dos desafios do ensino e aprendizagem de introdução à programação no ensino superior brasileiro.

A produção desta RSL teve a finalidade de tentar entender as dificuldades dos alunos iniciantes, para isso foi utilizado 3 perguntas que abordaram os conhecimentos prévios fundamentais para aprender programação, dificuldades enfrentadas pelos alunos para aprender e dificuldades enfrentadas pelos professores para ensinar disciplinas introdutórias de programação.

Em discussão sobre a RSL no contexto nacional foi apontado que o déficit em matemática e português acarretam problemas na interpretação e dificuldade de abstração dos problemas, desmotivando os alunos, também é exigido o conhecimento prévio em inglês básico para aprender o básico de programação.

Além disso, a relação entre a quantidade de alunos e de professores bem como a falta de infraestrutura adequada dificultam o aprendizado do aluno.

Também foi apontado que mesmo havendo várias publicações sobre métodos de ensino e ferramentas para os cursos de introdução à programação ainda há um baixo aproveitamento nas disciplinas de introdução à programação.

O artigo fomenta a discussão sobre desafios de programação introdutória e sugere questões que possam ser relevantes para a criação de um roteiro de pesquisa sobre aprendizado e ensino de programação introdutória no contexto brasileiro.

4.2 BlueJ

Kolling, Patterson & Rosenberg (2003) ressaltaram que os problemas que estudantes de curso introdutórios de programação enfrentam ao se depararem com os ambientes de desenvolvimentos tradicionais.

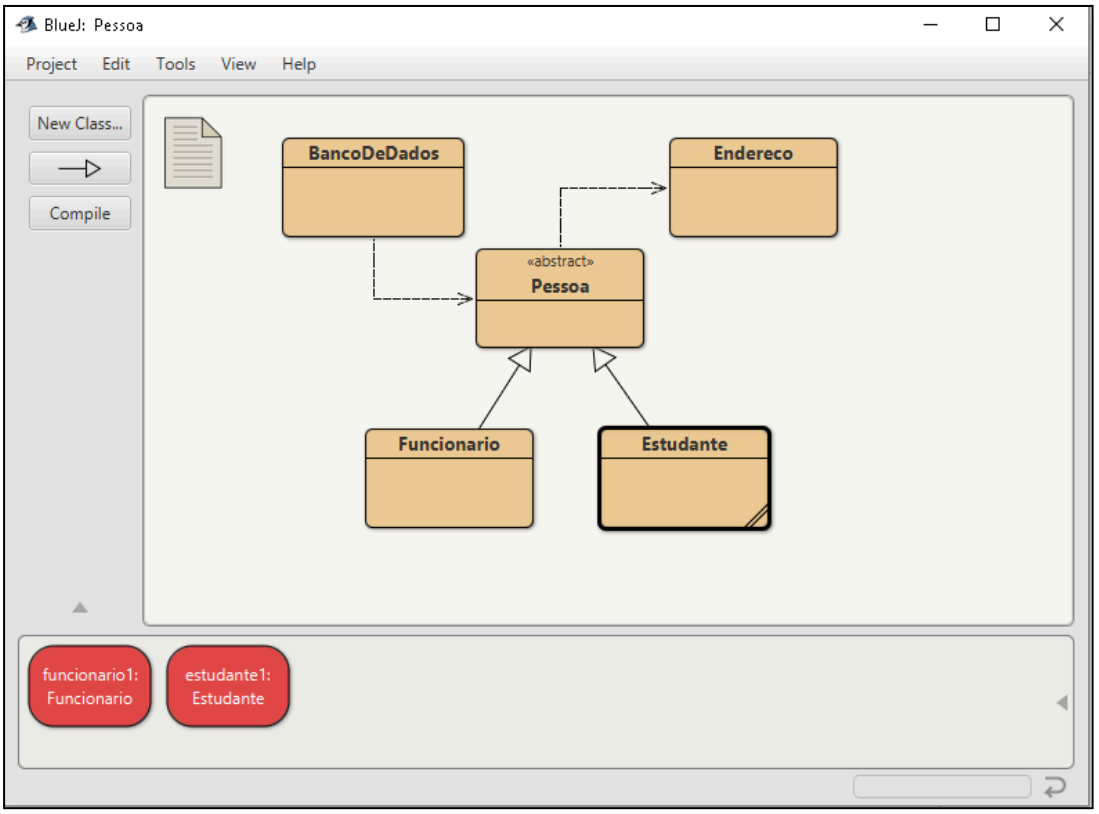
Um dos problemas citados é os ambientes tradicionais não serem orientados a objetos pois a interação com usuário é feito pela alteração das linhas de um código-fonte e muitas vezes tendo que lidar como sistema de arquivos do sistema operacional tornando mais difícil o foco no aprendizado de orientação à objetos.

Além disso, também observaram que os ambientes tradicionais possuem alto nível de complexidade na sua utilização, fazendo que os alunos precisem gastar um esforço prévio para memorizar linhas de comando que permitam executar as instruções do código-fonte.

Outro problema relatado foi utilização de muitos elementos gráficos que como construtores de interface que não agregam no aprendizado dos conceitos fundamentais de orientação a objetos ao invés da exibição gráfica da estrutura de classes e dos objetos instanciados com seus atributos e métodos.

BlueJ é um ambiente integrado de desenvolvimento (IDE - Integrated Development Environment) voltado ao ensino introdutório de programação com a linguagem Java sua interface tem como foco o aprendizado do conceito orientação a objetos, como exemplificado na Figura 1.

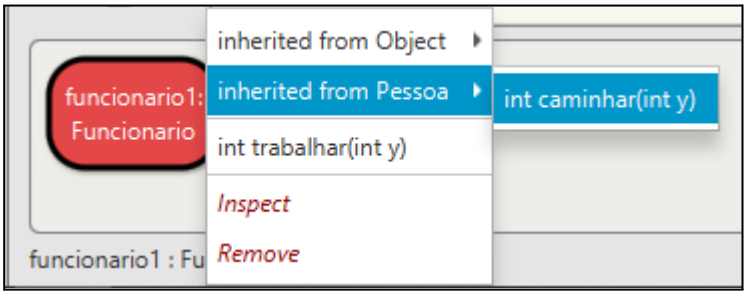
Figura 1 - Tela inicial com diagrama de classe e instâncias de objetos



Fonte: elaborado pelo autor

Esta ferramenta permite ao usuário em poucos cliques criar instâncias de objetos, chamadas de métodos e inspeção de objetos, como demonstrado na Figura 2. Além disso, possui visualização gráfica do diagrama de classes e instâncias de objetos, fornecendo aos estudantes maior simplicidade para que possam focar a atenção de aprendizado na orientação a objetos e em pouco tempo de instruções de uso do BlueJ.

Figura 2 - Métodos a serem executados pelo objeto selecionado “funcionário1”



Fonte: elaborado pelo autor

Os autores apoiam a utilização do BlueJ para ensino baseado em diretrizes que foram abordadas em um estudo anterior (Kolling & Rosenberg, 2001) que sugerem o aprendizado de orientação a objetos através de grandes projetos que visam abordar conceitos fundamentais .

O estudo expõe BlueJ como uma ferramenta facilitadora da aprendizagem de orientação a objetos e sugere apresentar aos alunos primeiramente a interação dos objetos através de projetos grandes para depois se terem contato com a linguagem de programação.

Em contrapartida, os autores relatam também que há dependência dos alunos com o BlueJ, uma resistência dos estudantes em migrarem para ferramentas profissionais de desenvolvimento e dificuldade da aplicação de algoritmos e estrutura de dados.

4.3 Greenfoot

Kölling (2008) apresenta a IDE Greenfoot como uma alternativa para estimular os alunos a ter mais interesse pela área da computação. Com uma abordagem que pode ser aplicada a alunos do ensino médio aos cursos superiores e descreve a proposta de uso da ferramenta de acordo com os grau de ensino. A Figura 3 apresenta um exemplo de implementação com Greenfoot.

Figura 3 - Flappy Bird desenvolvido no Greenfoot

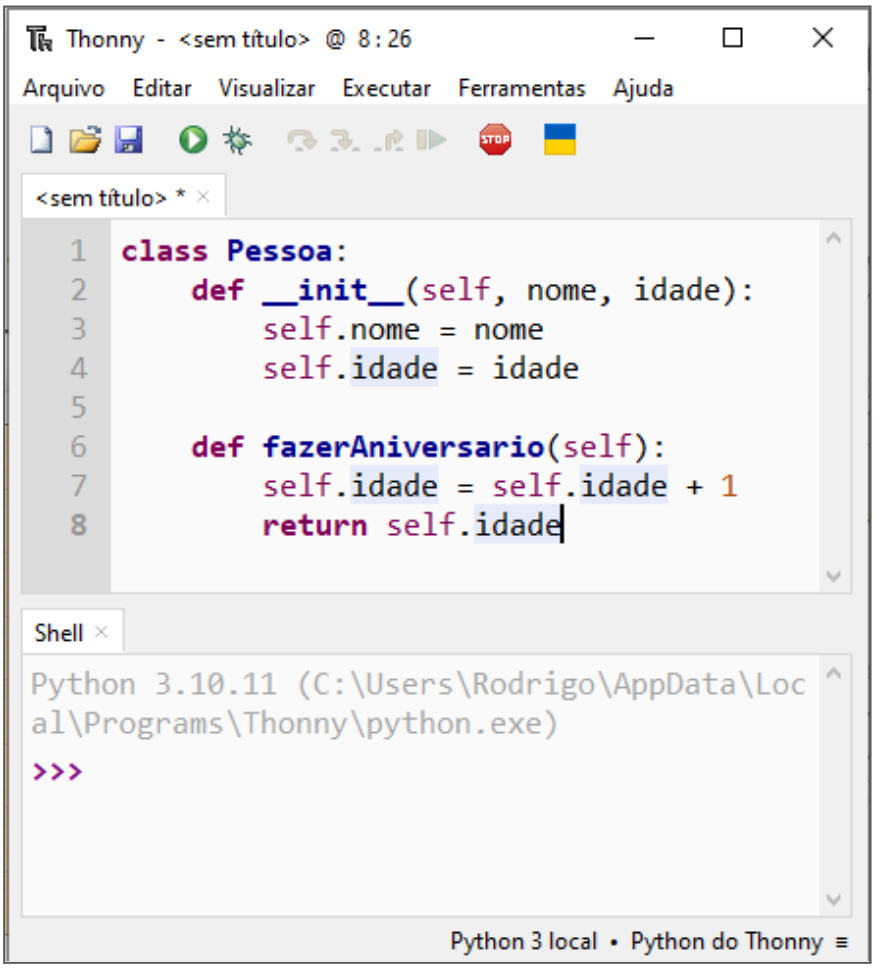
Fonte: elaborado pelo autor

Tendo em vista que este ambiente apresenta a programação orientada a objetos (POO) de forma interativa, apresentando respostas visuais imediatas e permite que sejam construídos aplicativos diversos. O Greenfoot se propõe a abordar o conceito de micromundo com foco na utilização de elementos gráficos animados bidimensionais para representar os elementos implementados.

4.4 Thonny

O Thonny é uma IDE (Integrated Development Environment) projetada para auxiliar iniciantes no aprendizado de programação com Python, oferecendo uma interface simples e recursos didáticos voltados à compreensão gradual de conceitos básicos. A Figura 4 demonstra a ferramenta em uso.

Figura 4 - Exemplo Thonny com Python



Fonte: elaborado pelo autor

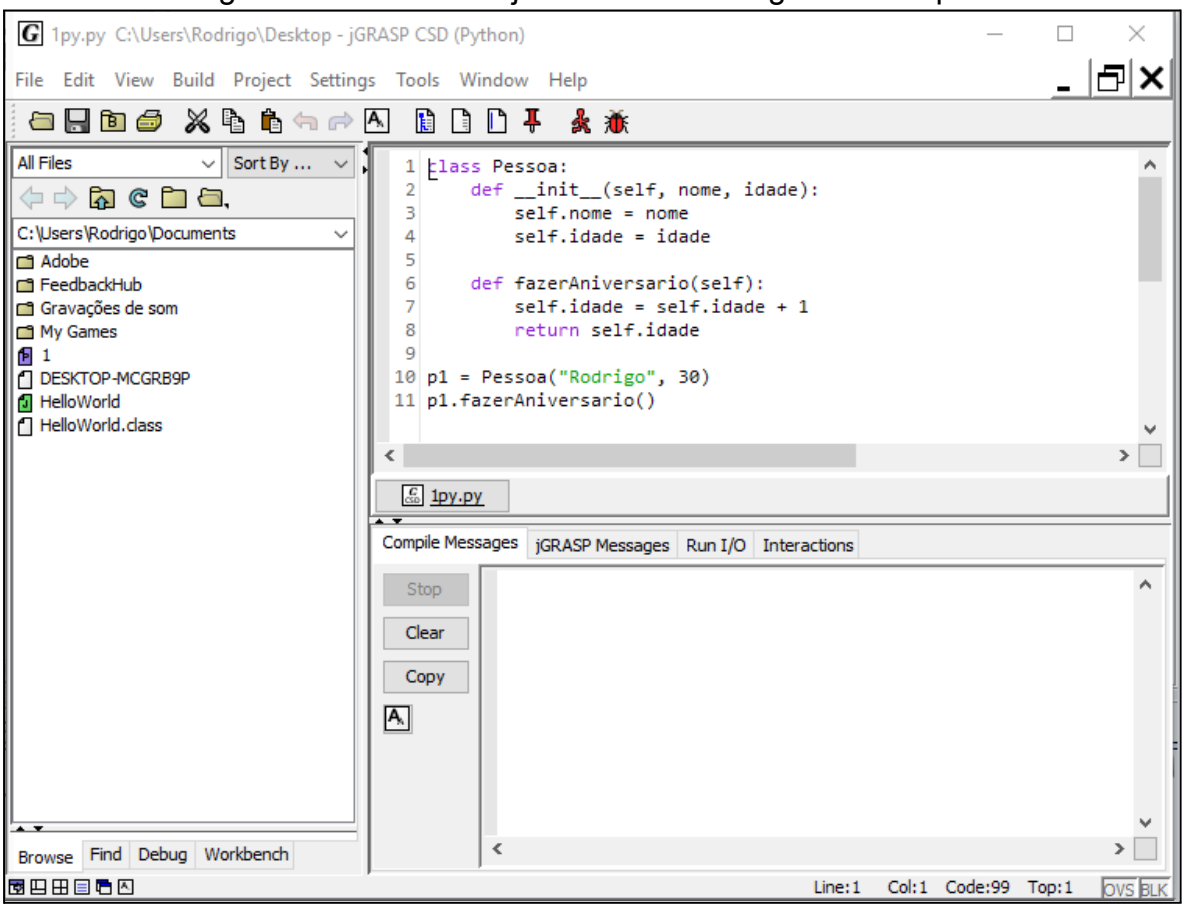
A ferramenta apresenta um depurador integrado que permite executar o código passo a passo, visualizando o estado das variáveis e a pilha de chamadas, o que facilita a compreensão de fluxos de controle e estruturas de dados. Além disso, o Thonny destaca erros de forma clara e fornece mensagens explicativas que ajudam no processo de correção. Essa abordagem tem se mostrado eficiente no ensino introdutório de programação, especialmente devido à sua simplicidade e à baixa curva de aprendizado exigida dos usuários. (LIU, LI & WANG, 2021)

4.5 jGRASP

O artigo de Miller, Reges e Obourn (2017), apresenta uma análise abrangente do ambiente de programação visual jGRASP, que foi projetado para facilitar o aprendizado e o ensino de ciência da computação, particularmente nos cursos introdutórios. O jGRASP é reconhecido por suas ferramentas de geração

automática de visualizações, como diagramas de estruturas de dados e execuções lógicas, que ajudam os estudantes a compreender conceitos complexos de programação. A interface do jGRASP pode ser vista na Figura 5.

Figura 5 - Ferramenta jGRASP com código de exemplo



Fonte: elaborado pelo autor

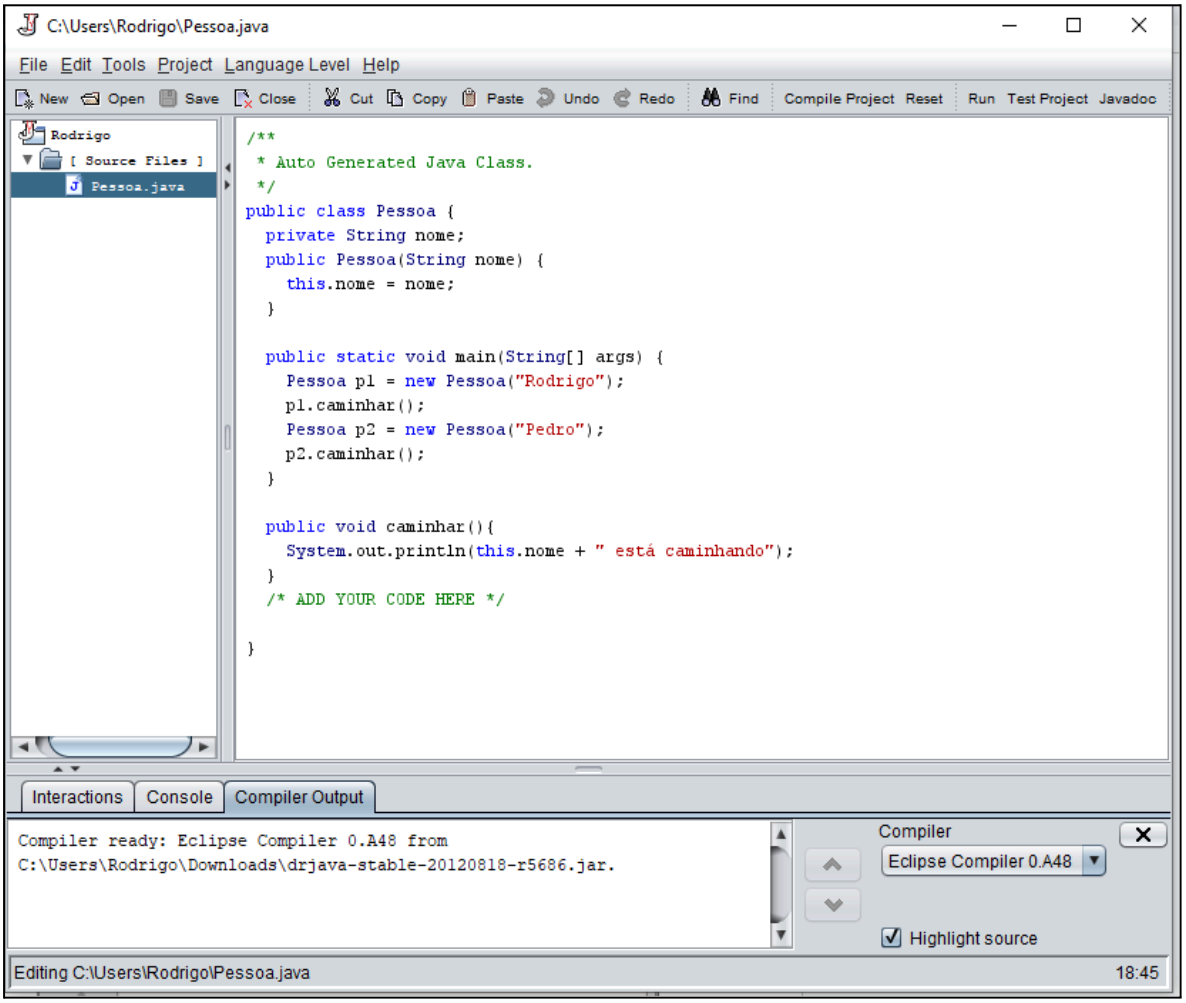
O ambiente se destaca pela simplicidade e pela interface intuitiva, promovendo uma aprendizagem interativa e acessível. A ênfase está na criação de um suporte visual que não só melhora a codificação, mas também amplia a compreensão conceitual de algoritmos e estruturas de dados. Os autores também discutem os benefícios do jGRASP em diversos contextos educacionais e como ele se integra a práticas pedagógicas modernas.

4.6 DrJava

O artigo de Allen, Cartwright e Stoler (2002), apresenta o DrJava como um ambiente leve e educacional voltado para o ensino de Java. Ele foi desenvolvido com foco na simplicidade, para permitir que estudantes se concentrem no design e

na lógica de programação, em vez de nas complexidades da ferramenta em si, conforme demonstra a Figura 6.

Figura 6 - Exemplo de projeto no DrJava



Fonte: elaborado pelo autor

DrJava utiliza um loop de leitura-avaliação-impressão (REPL), que possibilita o desenvolvimento incremental e iterativo de programas. O artigo descreve tanto a justificativa pedagógica por trás do ambiente quanto suas características técnicas e benefícios para o aprendizado.

4.7 Análise Comparativa

Com o intuito de definir um panorama das referências ao ensino e aprendizagem de programação foi elaborado uma análise comparativa das ferramentas BlueJ, Greenfoot e Thonny levando em consideração se o ambiente é

Web ou Standalone (que opera sem conexão de rede), possui o foco de ensinar POO e os critérios previamente definidos que os ambientes devem ter.

Tabela 1 - Comparação entre ambiente de ensino e programação

	BlueJ	Greenfoot	Thonny
Ambiente	Standalone	Standalone	Standalone
Facilidade de uso	Alta: ●Interface gráfica e simples	Moderada: ●Interface gráfica e complexa	Moderada: ●Interface textual e simples
Recursos Didáticos	Amplo ●Tutoriais ●Exemplos	Amplo: ●Tutoriais ●Exemplos	Básico: ● Tutoriais sem foco POO
Interatividade	Alta: ●Manipulação de Objetos e métodos	Alta: ●Simulação de cenários e atores	Moderada: ●Visualização de Variáveis e depuração
Feedback Imediato	Alta: ●Erros destacados de forma clara	Alta: ● Simulações com retornos visuais	Alta: ● Indicação de erro ● Depuração
Suporte a Conceitos Fundamentais de POO	●Abstração ●Herança ●Polimorfismo ●Encapsulamento	●Abstração ●Herança	●Somente por codificação
Suporte a Metodologias de Avaliação	Alta: ●Avaliação baseada em projetos e testes	Alta: ●Avaliação por meio de simulações	Básico: ●Saídas simples facilitam avaliação
Linguagem	●Java	●Java	●Python

Fonte: elaborado pelo autor

As ferramentas DrJava e jGrasp não foram incluídas nesta análise, pois não apresentam recursos didáticos nem suporte a conceitos fundamentais de POO.

5 Desenvolvimento do projeto

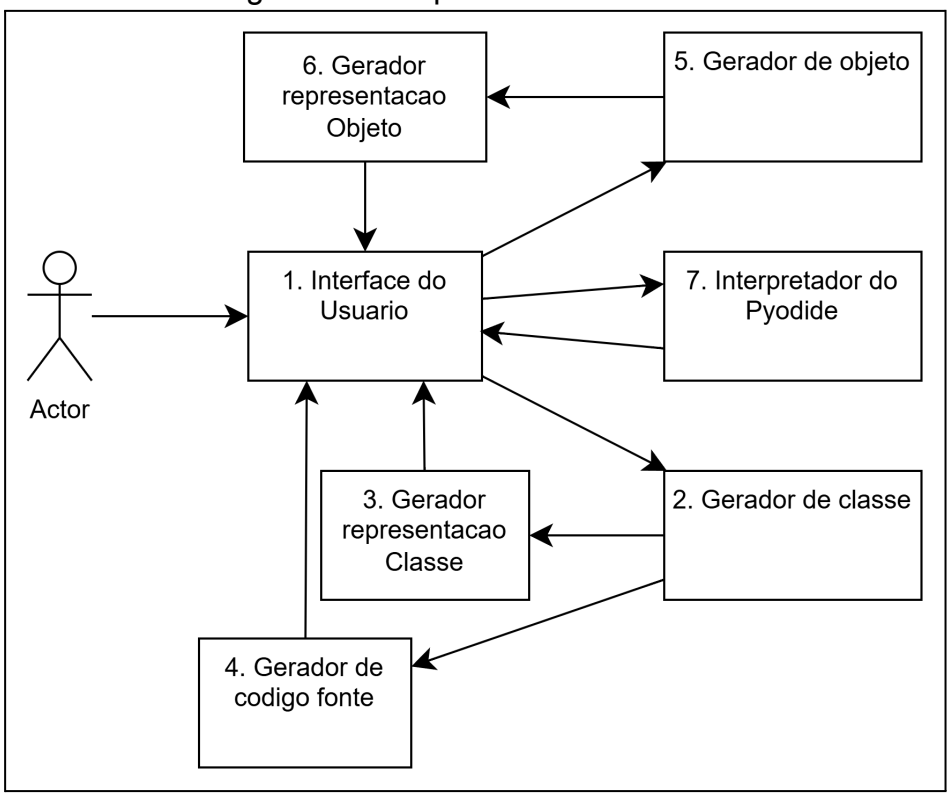
O intuito deste trabalho foi desenvolver uma aplicação Web para facilitar a compreensão dos principais conceitos da orientação a objetos relacionados a classes e objetos, priorizando abstrair a necessidade de conhecimento da linguagem de programação, gerando automaticamente o código-fonte e representações gráfica de classes e objetos a partir de entradas dos usuários, como cliques e textos digitados, por meio de na interface gráfica com interpretação de código no navegador.

Para criação da aplicação foi utilizado HTML e CSS para a estruturação e disposição dos elementos da interface gráfica, Javascript puro para a manipulação de comportamentos dos elementos da interface gráfica e tratamento das entradas do usuário e a integração do Pyodide (uma implementação da linguagem Python que roda diretamente no navegador, usando WebAssembly para interpretar o código). O código fonte da aplicação pode ser acessado através do repositório GitHub (https://github.com/rjnunesdev/rjnunesdev.github.io).

A solução proposta foi nomeada com o nome Gapy, pela junção para inglesa “*gap”*, que significa lacuna de conhecimento de POO e “py” é sigla da extensão de Python, que é a linguagem de programação que será utilizada para aprendizagem.

Aplicação desenvolvida é subdividida em oito componentes, como é possível observar na Figura 4, que serão detalhados nas subseções a seguir.

Figura 7 - Componentes de sistema



Fonte: elaborado pelo autor

O componente 1, “Interface de Usuário”, é responsável pela interação com os usuários, receber suas entradas de dados, direcioná-las para os devidos componentes e exibir o resultado dos processamentos como representações de classes, objetos, código-fonte e execução de métodos.

O componente 2, “Gerador de Classe”, utiliza os dados recebidos do componente 1 para identificar as definições de classe, atributos e métodos, organizá-las em dados estruturados que representam a arquitetura das classes no programa e encaminhá-las para os componentes 3 e 4.

O componente 3 intitulado “Gerador de Código Fonte”, usa os dados estruturados recebidos do componente 2 para converter em código Python e enviá-lo para o componente 1 para exibir ao usuário.

O “Gerador de representação de Classe”, dito como Componente 4, também utiliza estruturas das classes recebidas mas com a finalidade de gerar representações visuais das classes para apresentar aos usuários.

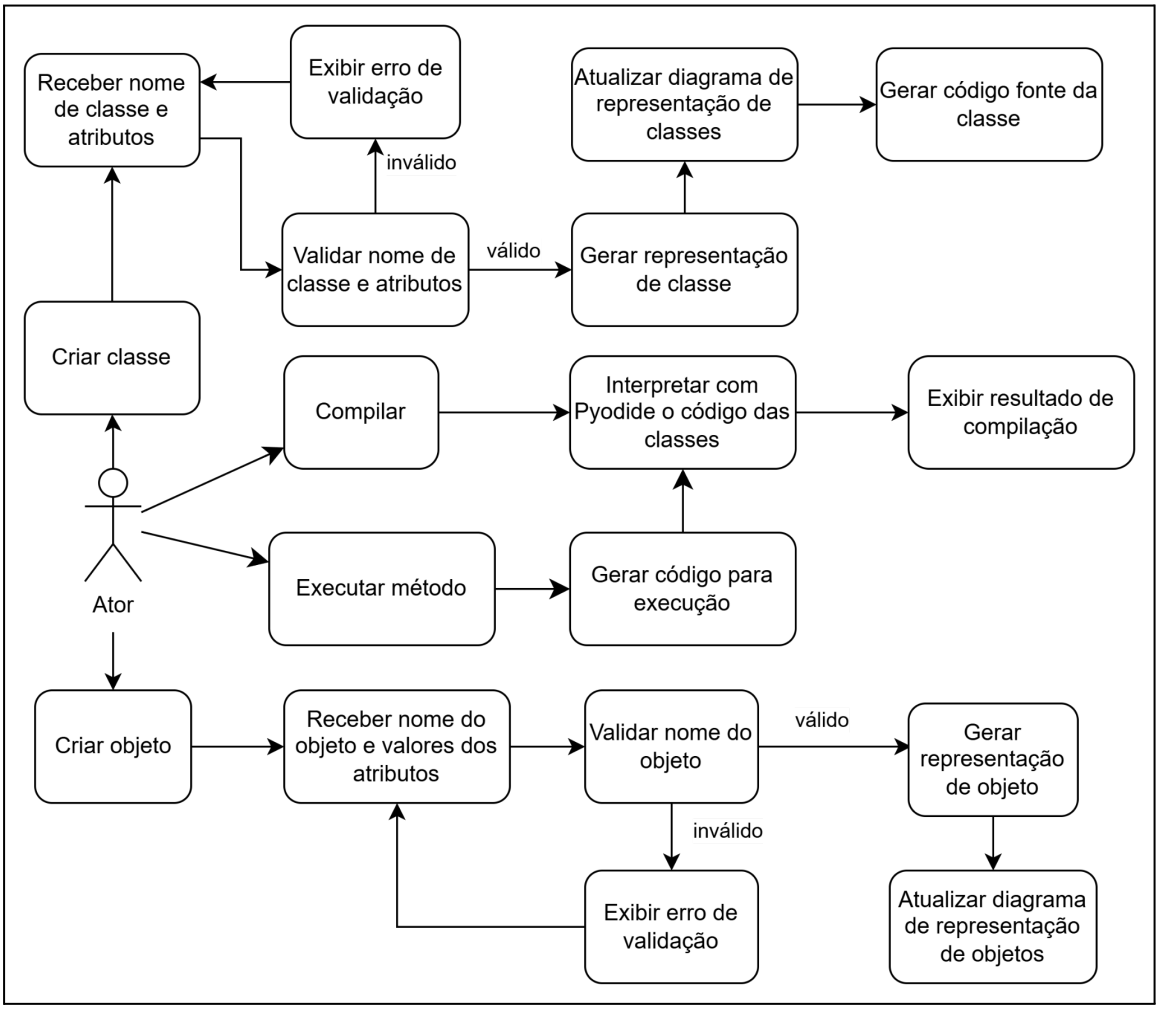
O “Gerador de objeto”, representado como componente 2, é responsável por receber os dados vindo do componente 1, identificar as definições da classe a ser instanciada, atributos e métodos dos objetos, bem como a criação de dados

estruturados para que possam se refletir em representações visuais que serão criadas de pelo “Gerador de representação de objetos” no componente 6.

Já o “Interpretador do Pyodide”, descrito no componente 7, interpreta o código-fonte gerado, executando-o e retornando os resultados.

A Figura 5 detalha o funcionamento do sistema desenvolvido através de um fluxograma, a partir das ações do usuário.

Figura 8 - Visão geral de arquitetura da solução Gapy



Fonte: elaborado pelo autor

O fluxo de criação e manipulação de classes e objetos inicia com o usuário definindo uma nova classe, fornecendo seu nome e atributos desejados. Em seguida, o sistema valida esses dados; caso encontre algum erro, exibe uma mensagem informativa, e, se a validação for bem-sucedida, avança para gerar uma

representação visual da classe. Essa representação é então integrada ao diagrama de classe.

Após gerar a representação, o sistema cria o código-fonte da classe com base na estrutura definida pelo usuário e realiza a compilação usando Pyodide, apresentando o resultado ao usuário para que ele possa verificar a ocorrência de possíveis erros ou a confirmação do sucesso.

A etapa seguinte envolve a criação de um objeto. O usuário fornece o nome do objeto e os valores dos atributos, que são validados pelo sistema. Se houver algum erro na validação, uma mensagem de erro é exibida; caso contrário, o sistema gera uma representação visual do objeto e atualiza o diagrama correspondente.

Por fim, o usuário pode executar métodos no objeto. Para isso, o sistema gera o código de execução e processa os métodos conforme solicitado. Esse fluxo oferece uma experiência interativa, permitindo ao usuário visualizar e corrigir erros, além de observar as representações visuais das classes e objetos criados ao longo do processo.

Esse capítulo está subdivido em oito subseções, a primeira subseção trata do componente 1 que se refere à descrição da disposição dos elementos visuais e a interação do usuário com o sistema.

Nas três subseções seguintes são descritos os componentes que têm relação com a criação de classes. Na segunda subseção, é apresentado o componente 2 descrevendo a validação dos dados de entradas do usuário fornecidos pelo componente 1 e como é feita a estruturação dos dados da classe. Na subseção seguinte, expõe-se como o componente 3 traduz os dados estruturados de classe, vindo do componente 2, em representações visuais de classe a serem exibidos para os usuários. A quarta subseção aborda como o componente 4 converte os dados estruturados de classe, vindos do componente 2, em código-fonte em Python.

A quinta e sexta subseções abordam sobre as criações do objeto, onde a quinta se refere a validação dos dados de entradas do usuário fornecidos pelo componente 1 e a forma que o componente 5 cria os dados estruturados dos objetos a partir de dados de entrada fornecidos pelo usuário no componente 1. E a sexta expõe como o componente 6 traduz os dados estruturados, vindos do componente 5, em representações visuais de objeto a serem exibidos para os usuários.

A sétima subseção define como é feita a interpretação do código-fonte gerado, além de descrever sobre o uso do Pyodide para interpretação de código Python e como é tratado o resultado da interpretação para exibir para o usuário pelo componente 1.

A última seção é a análise da aplicação em comparação a outras ferramentas já existentes, bem como a utilidade para ensino do conceitos fundamentais de programação orientada a objetos.

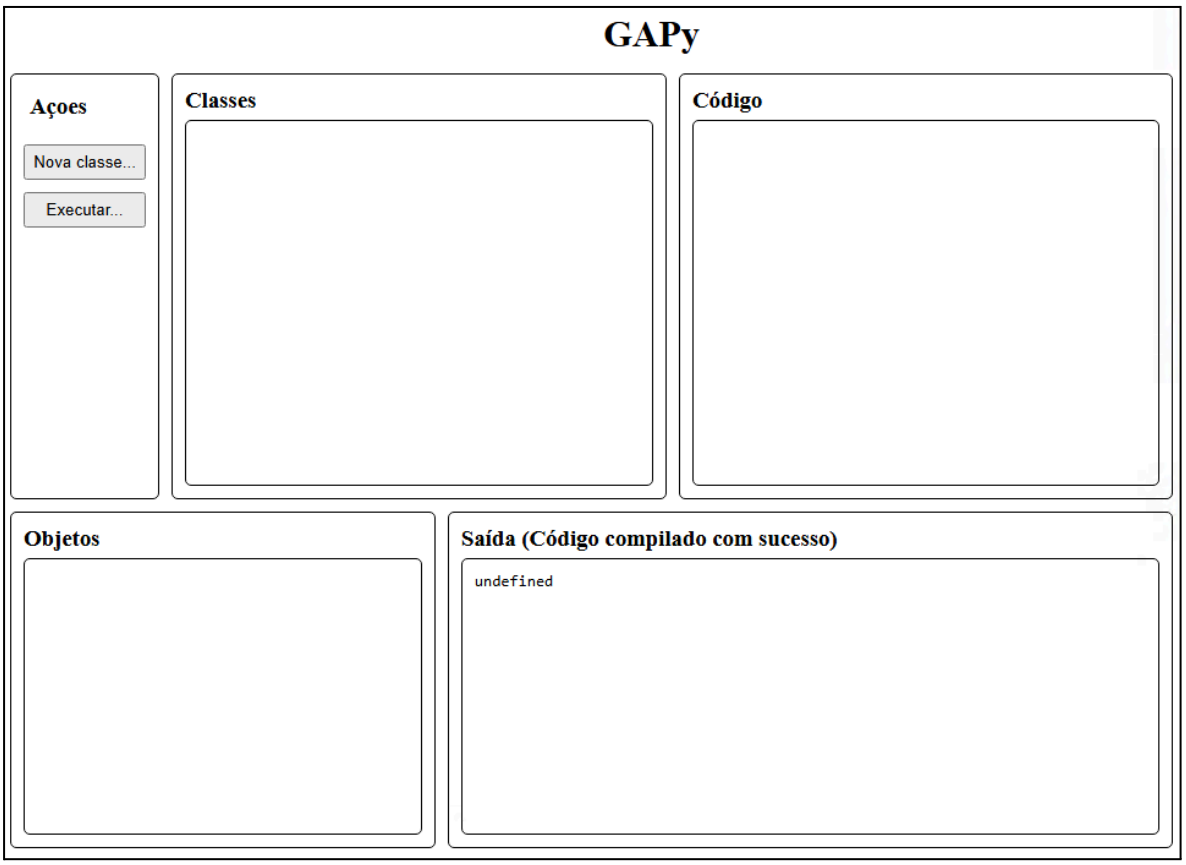
5.1 Interface do usuário

A interface do usuário, representado como componente 1, é responsável pela interação com o usuário, tratando a entrada e saída de dados. Ela permite ao usuário a inserção dos dados de classes e objetos e reflete ao usuário com representações visuais de objetos e classes.

Esse componente é responsável por enviar os dados de entrada do usuário aos respectivos componentes de processamento para que possam processá-los e retornar dados de visualização para exibir ao usuário.

Para detalhar o fluxo da aplicação pela interação do usuário, tem-se a seguir a Figura 6.

Figura 9 - Interface do usuário com a aplicação



Fonte: elaborado pelo autor

A seção da interface de “Ações” permite ao usuário fazer a ação “Nova classe…” e “Executar” . Já na seção de “Classes” é apresentada todas as classes criadas, onde cada classe possui a ação de “Criar objeto”. A seção “Código” apresenta todo o código gerado pelas classes criadas. Na seção de “Objetos” exibe todos os objetos criados com sua devida identificação de classe e cada representação de objeto e a seção “Saída” representa a saída do compilador com apontamento de erros caso haja.

5.1.1 Ações

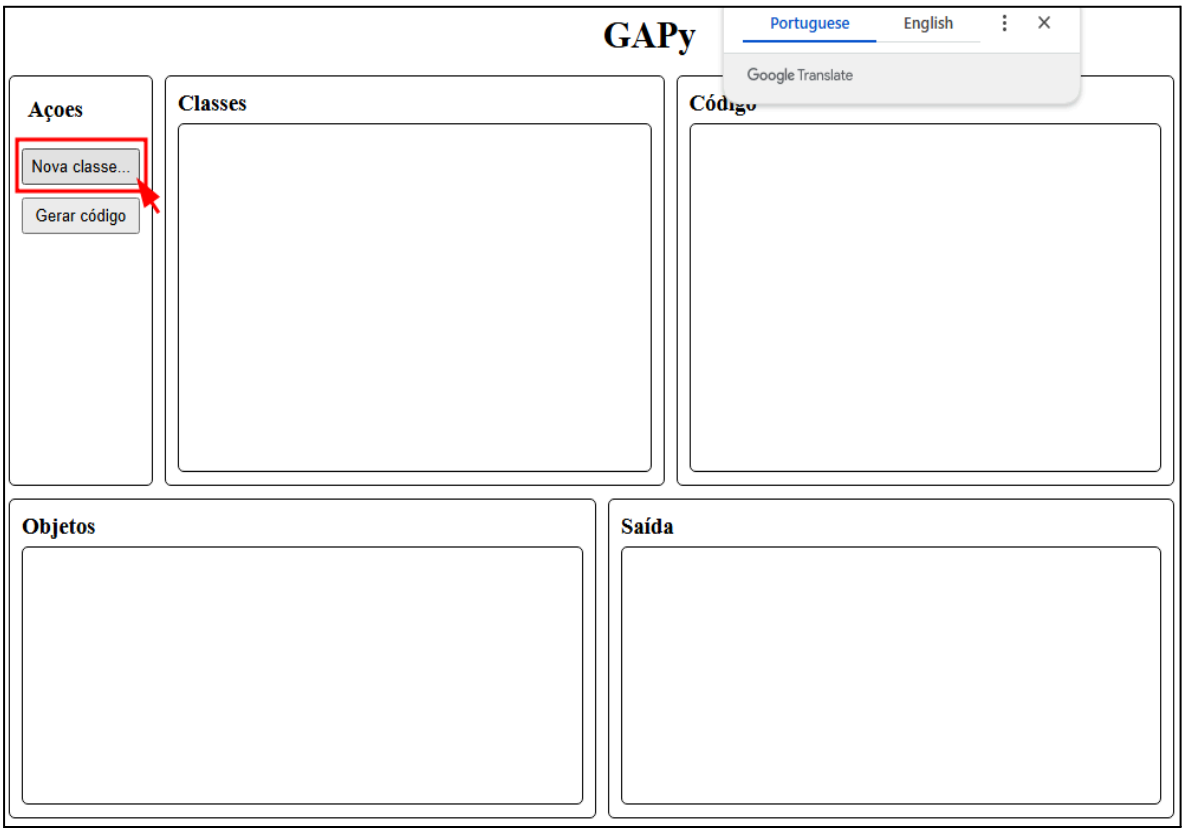
Esta seção descreve as características visuais de duas funcionalidades: A primeira é a “Nova Classe” que é responsável por solicitar ao usuário os dados para criação da classe, enviar esses dados ao componente “Gerador de Classe” descrito no item 5.1, que possui como retorno os dados necessários para exibir a representação da classe e o código-fonte gerado automaticamente. A segunda

funcionalidade é a “Gerar código” que invoca o componente responsável em gerar o código-fonte.

5.1.1.1 Nova Classe

Esta funcionalidade gera na interface a representação visual de uma classe a partir dos dados de entrada do usuário. O fluxo inicia ao clicar no botão “Nova Classe” localizado ao lado esquerdo da tela, como grifado abaixo na figura 7.

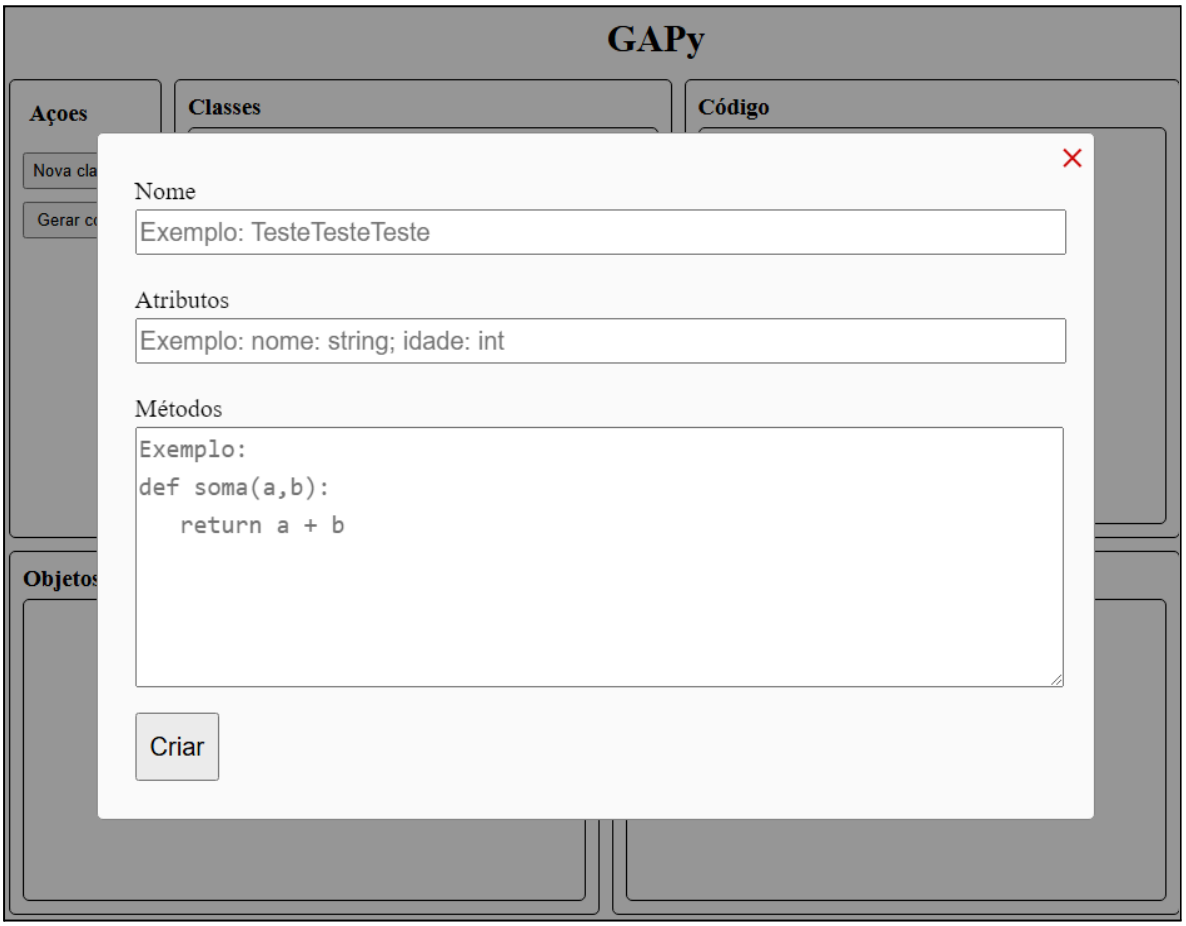
Figura 10 - Ação para criação de nova classe



Fonte: elaborado pelo autor

Em seguida, uma modal (janela flutuante sobre a aplicação) abrirá contendo os campos de texto “Nome”, “Atributos” e “Métodos” e o botão “Criar”, abaixo a Figura 8 demonstra a disposição destes elementos.

Figura 11 - Modal de criação de classe aberto



Fonte: elaborado pelo autor

Ao clicar em “Criar”, a interface se comunica com o componente de “Gerador de classe” descrito na subseção 5.2 deste trabalho.

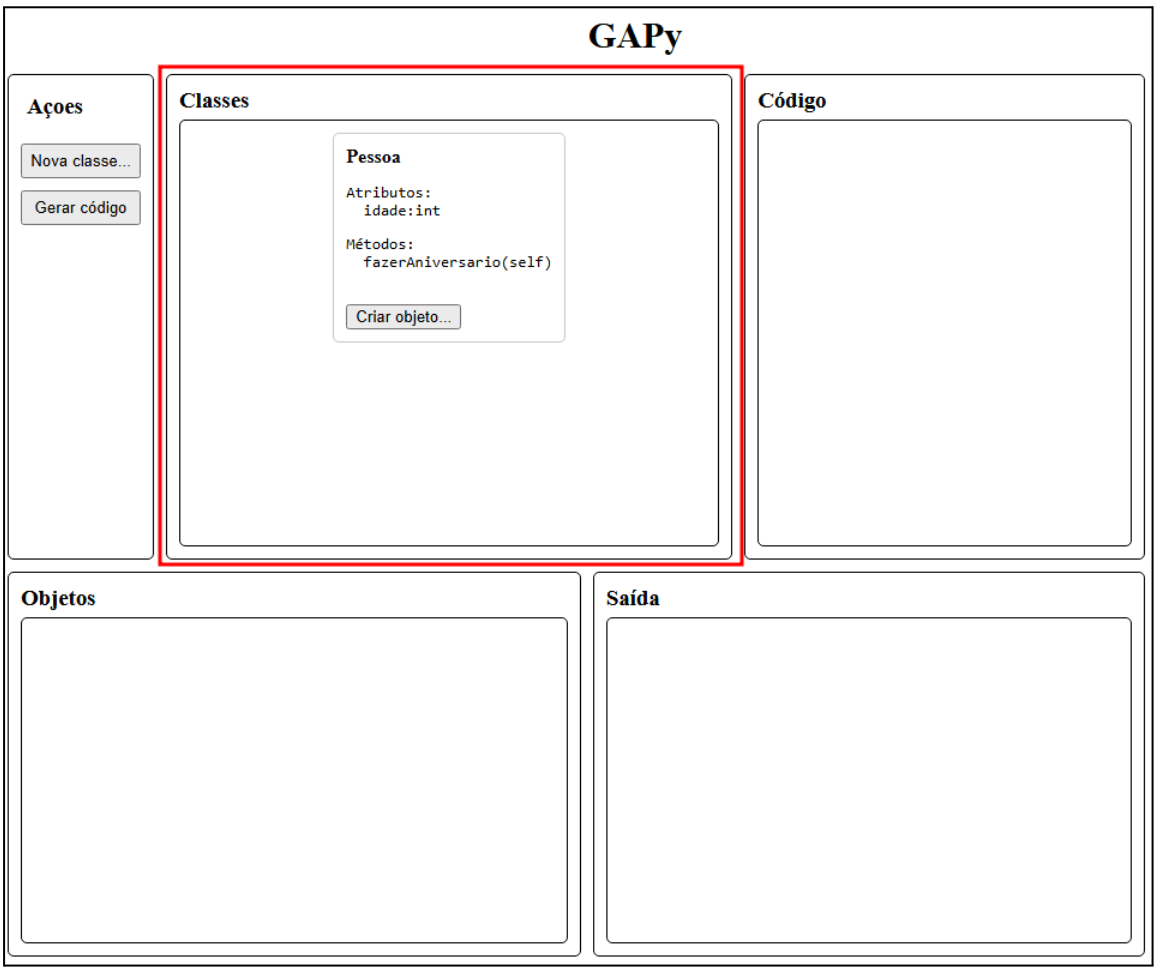
5.1.1.2 Gerar código

O botão “Gerar código”, ao ser acionado, faz a invocação componente “Gerador de código fonte” (descrito na subseção 5.4) e repassa o fluxo para os componentes gráficos “Código” e “Saída”

5.1.2 Classes

Esta seção de interface exibe todas representações visuais de classes criadas a partir da ação “Nova classe” e destacado na Figura 9. A intenção deste componente visual é mostrar de maneira mais abstrata e simplificada as propriedades das classes.

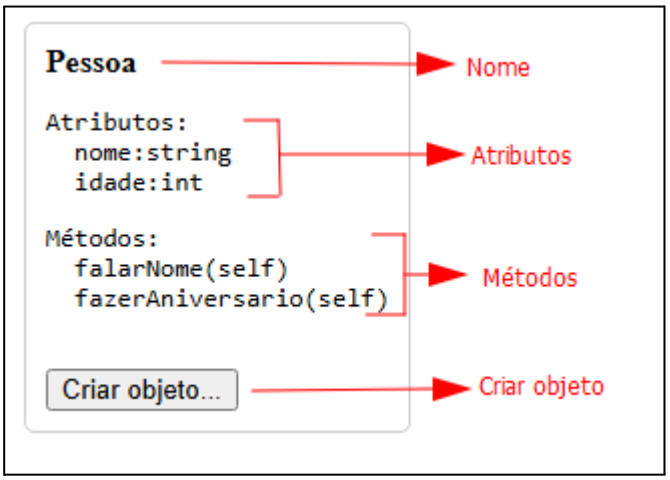
Figura 12 - Componente gráfico de classe em destaque



Fonte: elaborado pelo autor

As propriedades de classes exibidas são o nome da classe, atributos e métodos e a função de criar objetos, como pode ser identificado na Figura 10.

Figura 13 - Representação de classe



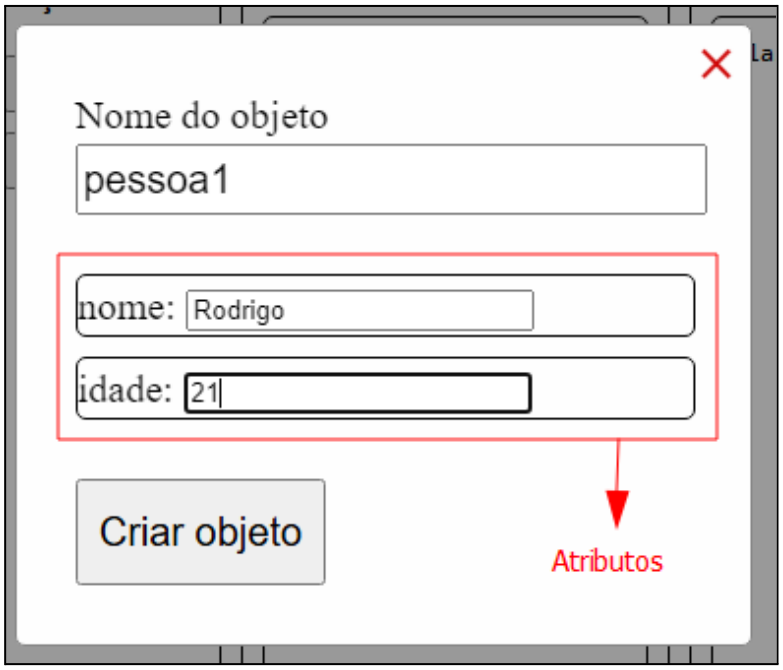
Fonte: elaborado pelo autor

O nome da classe fica destacado no topo em negrito, em seguida estão dispostos os atributos mais abaixo estão dispostos os métodos e na parte inferior tem o botão para “Criar objeto…”.

5.1.2.1 Criar objeto

O botão “Criar objeto” inicia o fluxo de criação de objeto de uma classe, abrindo uma *modal* para entrada de dados pelo usuário conforme mostrado na Figura 11.

Figura 14 - Modal de criação de objeto



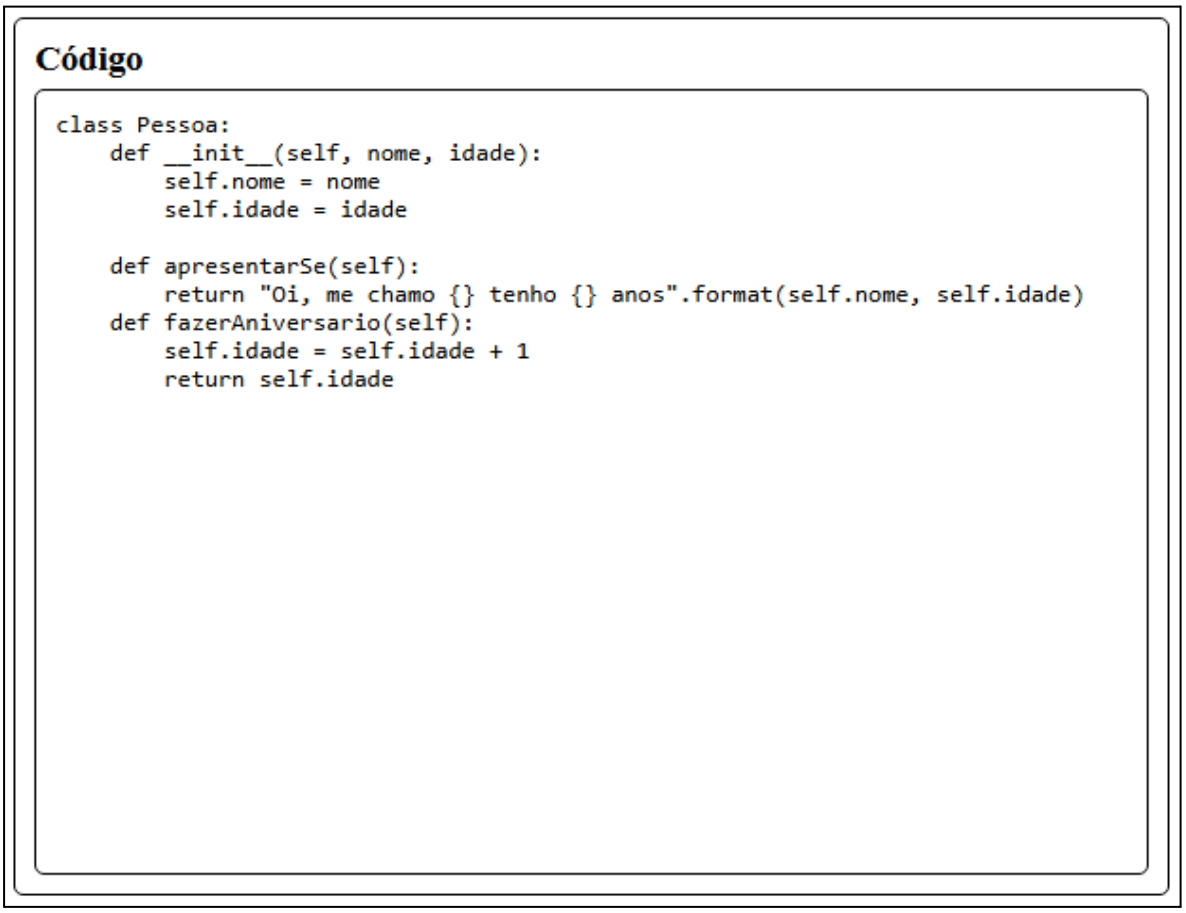
Fonte: elaborado pelo autor

Todos os campos desta *modal* devem ser preenchidos. O campo “Nome do objeto” é o identificador visual do objeto. Os campos de “Atributos” grifados na Figura 15 são os mesmos da classe a ser instanciada. O botão “Criar objeto” encaminha os dados desta *modal* para os componentes “Gerador de objeto” descrito em 5.5 deste trabalho.

5.1.3 Código

A Figura 12 exemplifica o componente visual “Código”, que é responsável somente por exibir o código-fonte gerado a partir do botão “Gerar código”, descrito no tópico 5.1.1.2. Este componente visual não permite a edição do código-fonte pelo usuário.

Figura 15 - Componente gráfico de código-fonte

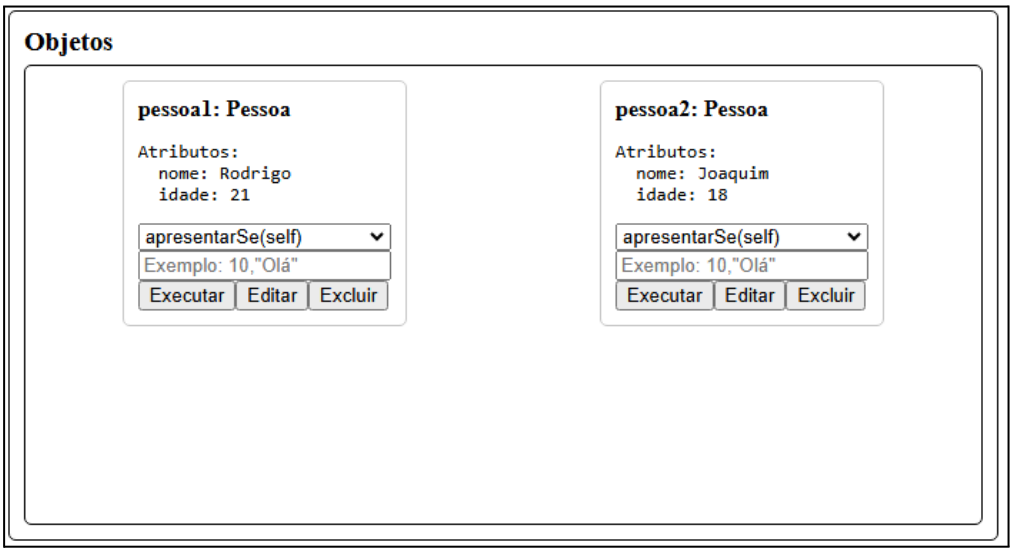


Fonte: elaborado pelo autor

5.1.4 Objetos

O componente visual “Objetos” tem a função de exibir os objetos criados pelo usuário como ilustrado na Figura 13.

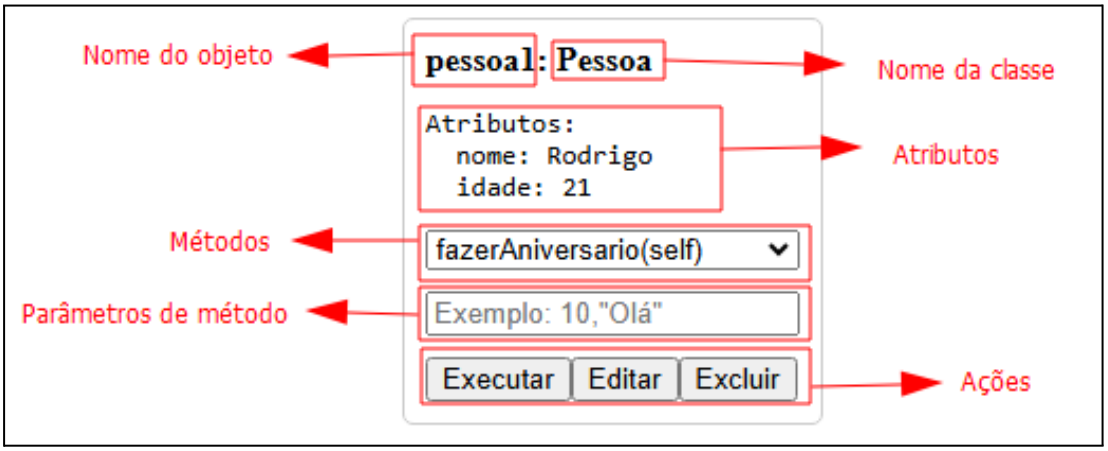
Figura 16 - Componente gráfico de objetos



Fonte: elaborado pelo autor

As representações de objeto desse componente possuem um padrão de exibição com a intenção de fornecer ao usuário uma visão geral de cada objeto como observado na Figura 18.

Figura 17 - Padrão de exibição de um objeto



Fonte: elaborado pelo autor

O “Nome do objeto” é usado para identificar o objeto e distingui-lo dos demais objetos. O elemento destacado na Figura 14 como “Nome da classe” aponta a classe que o objeto se baseia para ser instanciado. Os atributos com os seus respectivos valores estão indicados em “Atributos”. Os métodos a serem executados pelo objeto são apresentados em um elemento de seleção em destaque como “Métodos”, se o método selecionado houver necessidade de receber parâmetros devem ser inseridos no item “Parâmetros de método”. Em “Ações” estão dispostas

as ações serem exercidas em relação ao objeto. O botão “Excluir” deleta o objeto da aplicação e o botão “Executar” será detalhado na subseção 5.1.4.1.

5.1.4.1 Executar método

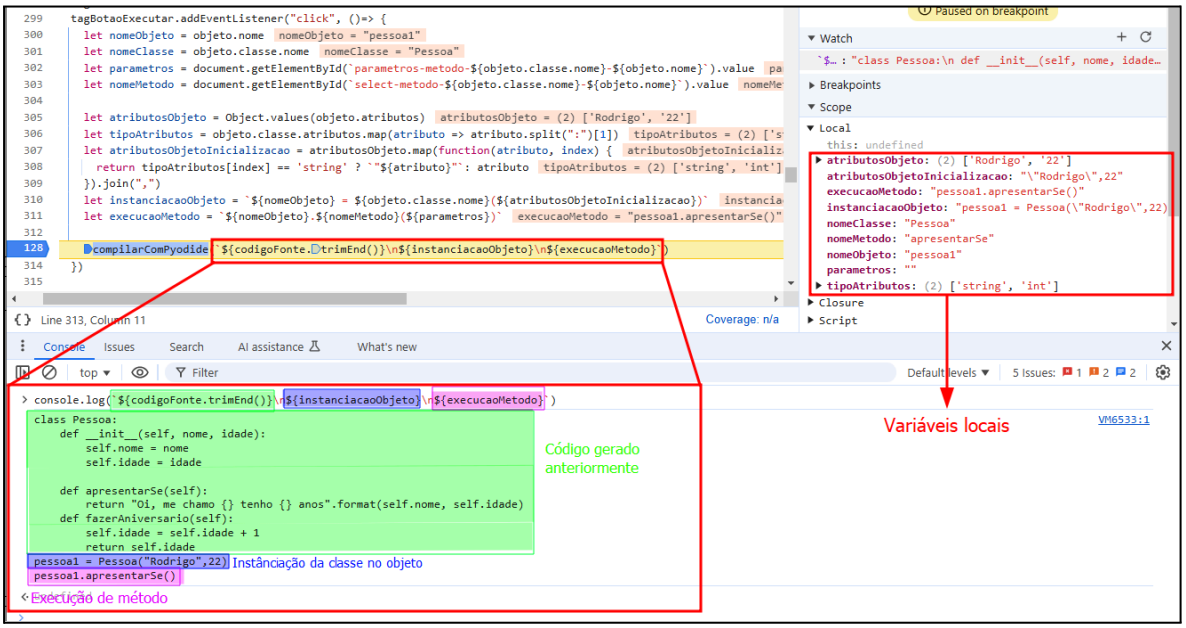
O botão “Executar”, ao ser acionado, obtém o método selecionado na caixa de seleção de métodos, obtém os parâmetros de método do campo abaixo da caixa de seleção e exibe o resultado no componente visual “Saída”.

A função atribui o nome do objeto na variavel “nomeObjeto”, o nome da classe do objeto na variável “nomeClasse”, obtém os parâmetros para executar o método armazena-o na variável “parametro” e obtém o método selecionado na caixa de seleção de métodos atribuindo-o a variável “nomeMetodo”.

Em seguida, extrai o valor do atributos do objeto para a variavel “atributosObjeto” como uma lista de *string*. Em “tipoAtributos” armazena uma lista de *string* referente o tipo de cada variável. Na variável “atributosObjetoInicializacao” é atribuido um *string* a partir da iteração de cada valor de atributo contido em “atributosObjeto”, a iteração trata se o tipo do atributo é *string,* se sim atribui aspas duplas no início e no fim do valor, caso contrário só retorna o valor, após o tratamento a lista resultante é concatenada com uma vírgula entre cada valor.

O próximo passo utiliza informações tratadas, a “instanciacaoObjeto” recebe uma com composição de *string* que representa o código da instanciação do objeto que executou o método, logo após em “execucaoMetodo” é armazenado uma composição de *string* equivalente ao código de execução do método conforme ilustra a Figura 18.

Figura 18 - Execução Javascript de método por objeto



Fonte: elaborado pelo autor

5.1.5 Saída

O componente visual de saída é responsável por exibir o resultado da execução de um método por um objeto e da interpretação do código gerado, não permitindo edição ao usuário. A Figura 19 exibe o retorno do interpretador Pyodide.

Figura 19 - Componente visual de saída

Saída (Código compilado com sucesso)

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def apresentarSe(self):
        return "Oi, me chamo {} tenho {} anos".format(self.nome, self.idade)
    def fazerAniversario(self):
        self.idade = self.idade + 1
        return self.idade
pessoa1 = Pessoa("Rodrigo",22)
pessoa1.apresentarSe()
# SAIDA:Oi, me chamo Rodrigo tenho 22 anos
```

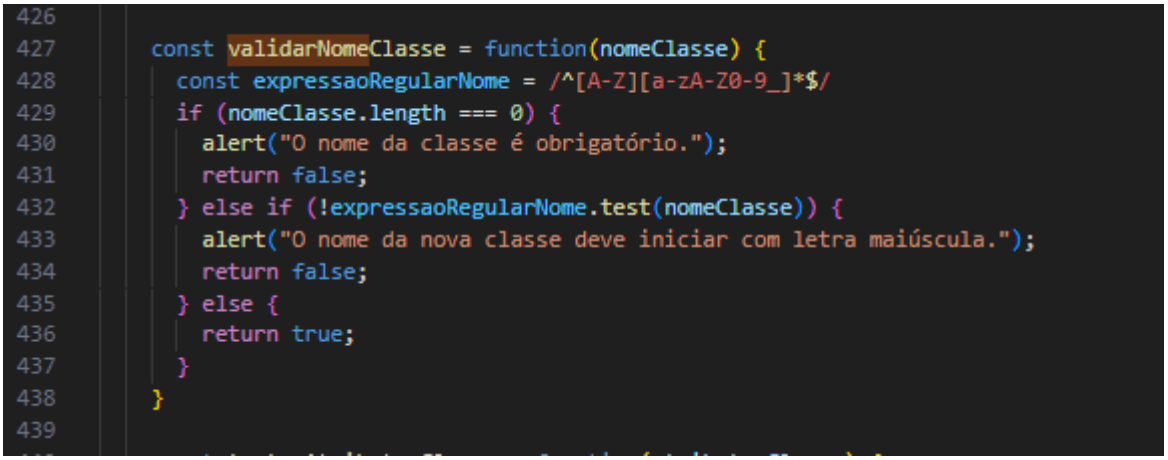
Fonte: elaborado pelo autor

5.2 Gerador de classe

O “Gerador de Classe” é responsável por criar os dados estruturados de classes através dos dados de entradas fornecidos pelo usuário. Este componente valida as entradas recebidas, identifica as definições de classe, métodos e atributos e organiza esses dados em uma estrutura que representa a arquitetura das classes no programa.

Primeiramente é recuperado da interface, respectivamente, o nome da classe, atributos e métodos. Em seguida na linha 448, é feita a validação do nome da classe pelo método “validarNomeClasse” conforme implementação demonstrada na Figura 18.

Figura 20 - Trecho do código-fonte Javascript de validação de nome de classe



Fonte: elaborado pelo autor

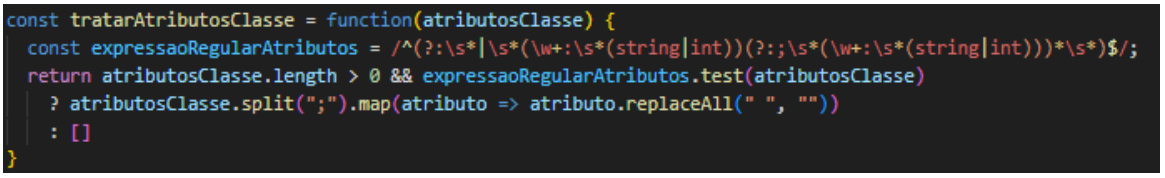
A validação avalia o nome da classe, considerando que não pode ser vazia e deve passar pelo teste da expressão regular para retornar valor verdadeiro, caso contrário retorna falso.

A expressão regular de classe avalia se é um nome válido para, o nome da classe deve começar com uma letra maiuscula por meio do trecho ““^[A-Z]” seguido por “[a-zA-Z0-9_]*” que analisa a validade do demais caracteres do nome pode conter letras (maiusculas ou minúsculas), números ou sublinhado e nome da classe deve terminar aqui “$”(não permite caracteres extras após o nome da classe).

No caso que o método “validarNomeClasse” retorne verdadeiro, o fluxo segue para a criação da estrutura de classe no sistema. Essa estrutura é armazenada em uma variável local, possui as propriedades que representam o nome, atributos e métodos da class. Para formação dessa estrutura é necessário tratar os campos de atributos e métodos

O método “tratarAtributosClasse” trata a propriedade de atributos, recebe como parâmetro o “atributosClasse”, que é do tipo texto contendo todos os atributos informados concatenados. A Figura 21 mostra que o método retorna uma lista vazia se o parâmetro for vazio ou não for válido para a expressão regular de atributos, caso contrário os atributos são separados do parâmetro ao encontrar o caractere “;”, retirada espaço de atributos e retorna uma lista de string, do tipo texto ,contendo os atributo concatenados pelo caractere “:”.

Figura 21 - Método de tratamento em Javascript de atributos de classe



```
const tratarAtributosClasse = function(atributosClasse) {
  const expressaoRegularAtributos = /^(?:\s*|\s*(\w+:\s*(string|int))(?:;\s*(\w+:\s*(string|int)))*\s*)$/;
  return atributosClasse.length > 0 && expressaoRegularAtributos.test(atributosClasse)
    ? atributosClasse.split(";").map(atributo => atributo.replaceAll(" ", ""))
    : []
}
```

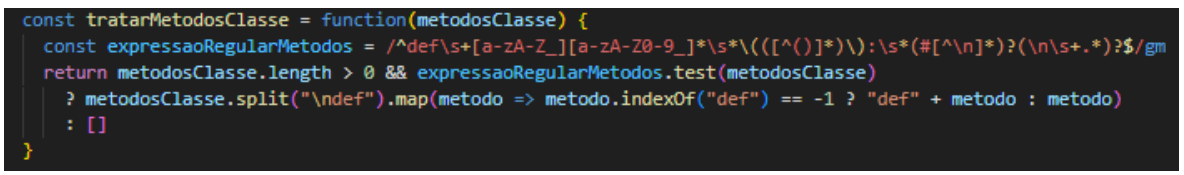
Fonte: elaborado pelo autor

A expressão regular de atributos avalia se parâmetro possui atributos bem formados conforme esperado pela aplicação. A formação do campo atributos pode ser dado vazio (sem atributos), possuir um atributo com nome (pode conter qualquer caractere alfanumérico sem acento) e tipo (string para expressar textos e int para expressar números).

Já o método “tratarMetodosClasse” recebe como parâmetro o “metodosClasse”, que é do tipo texto contendo todos os métodos informados concatenados. A Figura 21 mostra que o método retorna uma lista vazia se o parâmetro for vazio ou não for válido para a expressão regular de métodos, caso contrário os métodos são separados do parâmetro ao encontrar o caractere especial de quebra de linha “\n” juntamente com o conjunto de caracteres “def” que caracteriza um método.

A expressão regular de método avalia se o parâmetro é válido. A expressão pode ser dividida em três parte: “def\s+[a-zA-Z_][a-zA-Z0-9_]*\s*\(([^()]*)\):” para validação do cabeçalho do método, “(\n\s+.+)*” verifica se tem múltiplas linhas de corpo.

Figura 22 - Método de tratamento em Javascript de métodos de classe



```
const tratarMetodosClasse = function(metodosClasse) {
  const expressaoRegularMetodos = /^def\s+[a-zA-Z_][a-zA-Z0-9_]*\s*\(([^()]*)\):\s*(#[^\n]*)?(\n\s+.*)?$/gm
  return metodosClasse.length > 0 && expressaoRegularMetodos.test(metodosClasse)
    ? metodosClasse.split("\ndef").map(metodo => metodo.indexOf("def") == -1 ? "def" + metodo : metodo)
    : []
}
```

Fonte: elaborado pelo autor

Após os tratamentos de atributos e métodos, a estrutura de representação da classe é criada com as propriedades: “nome” (tipo texto), atributos e métodos (ambos tipo lista) e é armazenado na variável de escopo de bloco do método

“novaClasse”. Essa variável é passada por parâmetro para o componente “Gerador de representação de Classe”, que será detalhado na subseção 5.3, em seguida encerra seu comportamento limpando os campos de entrada e voltando para a tela inicial.

5.3 Gerador de representação de classe

O “Gerador de Representação de Classe” é o componente responsável por criar representações visuais das classes definidas pelo usuário. Este componente recebe os dados estruturados de classe do “Gerador de Classe” e gera representação visual da classe. A visualização das classes facilita o entendimento da arquitetura do programa, tornando o aprendizado de POO mais acessível e intuitivo.

As quatro primeiras linhas da figura são subfunções usadas apenas no escopo da função “gerarRepresentacaoClasses”, em seguida há a primeira chamada de função para “gerarDivClasse” que usa “novaClasse” como parâmetro e seu resultado é armazenado em “representacaoVisualClasse” A função “gerarDivClasse” utiliza as propriedades da “novaClasse” para criar um elemento visual que represente essa classe.

As subfunções destacadas em vermelho e verde na figura são responsáveis de unir respectivamente os atributos e métodos de uma classe, na Figura 23 exibe a implementação que cada uma das funções grifadas.

Figura 23 - Subfunções Javascript “tratarAtributos” e “tratarMetodos”

```
const tratarAtributos = function(atributos) {
  return atributos.length !== 0
    ? atributos.join("\n"+indentacao)
    : "(SEM ATRIBUTOS)"
}
const tratarMetodos = function(metodos) {
  let existemMetodos = metodos.length !== 0
  return existemMetodos
    ? metodos.map(metodo => metodo.substring(metodo.indexOf("def ")+4, metodo.indexOf(":")))
             .join("\n"+indentacao)      You, 49 minutes ago • Uncommitted changes
    : "(SEM MÉTODOS)"
}
```

Fonte: elaborado pelo autor

A subfunção “tratarAtributos” recebe como parâmetro a lista de atributos de uma classe, se a lista de atributos estiver vazia, então o método retorna a *string* “(SEM ATRIBUTOS)” evidenciando para o usuário que não há atributos, se houver atributos eles serão concatenado com “\n” (para quebrar em nova linha) + “indentacao” (constante com valor de quatro espaços) para retornar um único *string.*

Em seguida “gerarDivClasse” usa o retorno de “tratarAtributos” e “tratarMetodos” para gerar o elemento HTML como retorno deste método que é armazenado na variável “representacaoVisualClasse” e invoca a função “adicionarRepresentacaoVisual” com “representacaoVisualClasse” como parâmetro, refletindo assim na representação visual apresentada na Figura 10 da subseção 5.1.2.

5.4 Gerador de código fonte

O “Gerador de Código Fonte” é responsável por converter a estrutura de classe, em código Python formatado e pronto para execução. A estrutura de classe é no formato JSON (JavaScript Object Notation), como observado na Figura 25.

Figura 25 - Estrutura de classe no formato JSON

```
{
    nome: string,
    atributos: Array<string>,
    metodos: Array<string>
}
```

Fonte: elaborado pelo autor

A função “gerarCodigoFonteClasse” usa as propriedades do parâmetro “classe” para gerar o código fonte da classe que contém três elementos principais: cabeçalho de classe, método construtor com atributos e métodos personalizados pelo usuário como evidenciado pela Figura 26.

Figura 26 - Implementação da subfunção “gerarCodigoFonteClasse” em Javascript

```
const gerarCodigoFonteClasse = function(classe) {
  const gerarConstrutorClasse = function(atributos) {
    let codigo = '';
    if (atributos.length > 0) {
      let nomesAtributos = atributos.map(attr => attr.split(":")[0])
      codigo += `${indentacao}def __init__(self, ${nomesAtributos.join(", ")}):\n`;
      nomesAtributos.forEach(nomeAtributo => {
        codigo += `${indentacao+indentacao}self.${nomeAtributo} = ${nomeAtributo}\n`;
      });
    }
    return codigo;
  };
  let metodosIndentados = indentacao + classe.metodos.map(metodo => metodo.replaceAll("\n", `\n${indentacao}`)).join(`\n${indentacao}`)
  let cabecalho = `class ${classe.nome}:`
  let metodoConstrutor = gerarConstrutorClasse(classe.atributos)
  const codigoClasse = `${cabecalhoClasse}\n${metodoConstrutor}\n${metodosIndentados}\n`;
  return codigoClasse;
};
```

Fonte: elaborado pelo autor

A primeira atividade da função é indentar os métodos e atribuí-los à variável “metodosIndentados”. Em seguida, gerar o cabeçalho da classe concatenando a palavra reservada *class,* o nome da classe e o caractere “:”(dois pontos). E logo após segue para a montagem do método construtor da classe que inicia com o *string* “def __init__(self, ” unido com uma lista dos atributos concatenado soma a *string* “):\n” para fechar corretamente a assinatura do método e quebra para próxima linha, por fim itera sobre os atributos e a cada atributo gera o código para a atribuição coerente com os parâmetros no formato: duas indentações unido com a palavra “self.” juntamente com o nome do atributo, o caractere “=” (igual) e o nome do atributo novamente.

O cabeçalho de classe, método construtor com atributos e métodos personalizados são concatenados e tem como resultado o código fonte da classe, que são atribuídos a variável “codigoClasse” que será retornada.

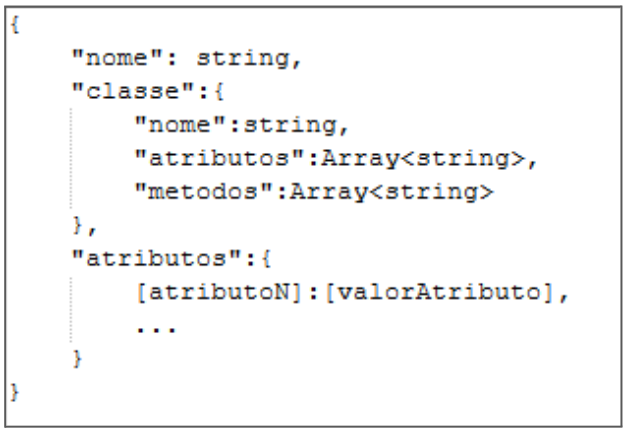
Após gerado o código fonte de cada classe, essa lista de códigos de classe são concatenadas com “\n” (para quebrar em nova linha) para ser exibida conforme subseção 5.1.3 Figura 15.

5.5 Gerador de objeto

O Gerador de Objeto é responsável pela criação das instâncias das classes definidas pelo usuário. Após o usuário definir as classes e métodos por meio da Interface do Usuário, este componente permite que ele crie objetos a partir dessas classes.

O fluxo de geração de objeto é iniciado pela interface do usuário como descrito na Figura 11 da subseção 5.1.2.1. Este componente recebe da interface a classe, nome do objeto, valores dos atributos, cria a estrutura do objeto no sistema, em formato *JSON*, e invoca o "Gerador de representação de objeto" que será descrito na subseção 5.6. A implementação do componente gera uma estruturação do objeto no formato *JSON*, conforme exibido na Figura 27.

Figura 27 - Estrutura de objeto no formato JSON



Fonte: elaborado pelo autor

Primeiramente, o componente invoca o método que valida o nome do objeto que deve retornar *true* para o fluxo de geração de objetos dar continuidade. A implementação detalhada validado está exposta na Figura 28.

Figura 28 - Função de validação do nome do objeto em Javascript

```
219  const validarNomeObjeto = function(nomeObjeto) {
220    let regexNomeObjeto = /^[a-z][a-zA-Z0-9_]*$/gm
221    if (nomeObjeto === "") {
222      alert("O nome do objeto é obrigatório.");
223      return false;
224    } else if (expressaoRegularNomeObjeto.test(nomeObjeto)) {
225      alert("Um objeto com esse nome já existe.");
226      return false;
227    } else {
228      return true;
229    }
230  }
```

Fonte: elaborado pelo autor

A função retorna *false*, quando o nome do objeto é vazio (avaliação feita na linha 221) ou se o nome do objeto recebido por parâmetro for inválido para expressão regular atribuída na variável "regexNomeObjeto" testado na linha 224.

A expressão regular contida em “regexNomeObjeto” avalia que o nome do objeto deve iniciar com letras minúsculas sem acento, pelo trecho “^[a-z]”, podendo ser precedido e terminado com letras minúsculas e maiusculas sem acento, números e caractere “_” expresso pelo trecho “[a-zA-Z0-9_]*$”.

Se ambas avaliações forem validadas a função “validarNomeObjeto” como verdadeiro a função retorna *true* e o fluxo volta para a função “criarObjeto” na linha 401 ilustrado na Figura 27. Neste ponto da implementação a estrutura do objeto é criada em formato JSON.

A estrutura do objeto possui três propriedades: “nome” do tipo *string* que recebe o nome da classe, “classe” do tipo JSON, tendo como valor a estrutura da classe a ser instanciada e na propriedade “atributos” é armazenada os valores dos atributos no formato JSON usando o modelo estrutura dados chave-valor onde chave usada será o nome do atributo e o valor será o valor dado pelo usuário. Esta estrutura de objeto criada é atribuída à variável local “novoObjeto”.

Na linha 402 descrito na Figura 27, da função “criarObjeto”, o “novoObjeto” é adicionado na variável global “objetos” que é uma lista de objetos criados pelo usuário. Na linha seguinte, invoca o componente “Gerador de representação de objeto” através da função “gerarRepresentacoesObjetos” abordada na subseção 5.6.

5.6 Gerador de representação de objeto

O “Gerador de Representação de Objeto” é o componente responsável por criar representações visuais dos objetos definidas pelo usuário. Este componente recebe os dados estruturados do objeto do “Gerador de Objeto” e gera representação visual do objeto.

O componente transforma a estrutura de objeto criado pelo “Gerador de objeto” em uma representação gráfica do objeto como pode ser visto na Figura 16 da subseção 5.1.4.

O elemento gráfico do objeto criado visa simplificar a compreensão da relação entre classe e objeto. Exibindo o nome identificador do objeto juntamente com o nome da classe instanciada separado pelo caractere “:”, lista de atributos com seus valores conforme definidos na criação do objeto e lista de métodos definidos na classe instanciada para serem executadas pelo objeto.

Em seguida, agrega os botões de Executar, Editar e Excluir apresentados pela implementação da Figura 32

Figura 32 - Funcão da ação dos Botões em Javascript

```
const criarBotaoExecutar = function(objeto) {
  const tagBotaoExecutar = document.createElement("button")
  tagBotaoExecutar.textContent = "Executar"
  tagBotaoExecutar.addEventListener("click", ()=> {
    let nomeObjeto = objeto.nome
    let nomeClasse = objeto.classe.nome
    let parametros = document.getElementById(`parametros-metodo-${objeto.classe.nome}-${objeto.nome}`).value
    let nomeMetodo = document.getElementById(`select-metodo-${objeto.classe.nome}-${objeto.nome}`).value

    let atributosObjeto = Object.values(objeto.atributos)
    let tipoAtributos = objeto.classe.atributos.map(atributo => atributo.split(":")[1])
    let atributosObjetoInicializacao = atributosObjeto.map(function(atributo, index) {
      return tipoAtributos[index] == 'string' ? `"${atributo}"`: atributo
    }).join(",")
    let instanciacaoObjeto = `${nomeObjeto} = ${nomeClasse}(${atributosObjetoInicializacao})`
    let execucaoMetodo = `${nomeObjeto}.${nomeMetodo}(${parametros})`

    interpretarComPyodide(`${codigoFonte.trimEnd()}\n\n${instanciacaoObjeto}\n${execucaoMetodo}`)
  })
  return tagBotaoExecutar
}
const criarBotaoEditar = function(objeto) {
  const tagBotaoEditarObjeto = document.createElement("button");
  tagBotaoEditarObjeto.textContent = 'Editar';
  tagBotaoEditarObjeto.addEventListener("click", () => abrirModalEditarObjeto(objeto));
  return tagBotaoEditarObjeto
}
const criarBotaoExcluir = function(objeto) {
  const tagBotaoExcluirObjeto = document.createElement("button");
  tagBotaoExcluirObjeto.textContent = 'Excluir';
  tagBotaoExcluirObjeto.addEventListener("click", () => excluirObjeto(objeto.nome));
  return tagBotaoExcluirObjeto
}
```

Fonte: elaborado pelo autor

Ao criar o botão “Executar” tem a função de buscar as informações de nome de objeto, nome de classe, parâmetros, método selecionado e os atributos do objeto, o tipo dos atributos, prepará-los para serem passados como parâmetro na instanciação do objeto, montagem do método a ser executado e envio com o interpretador com Pyodide que será abordado em 5.7.

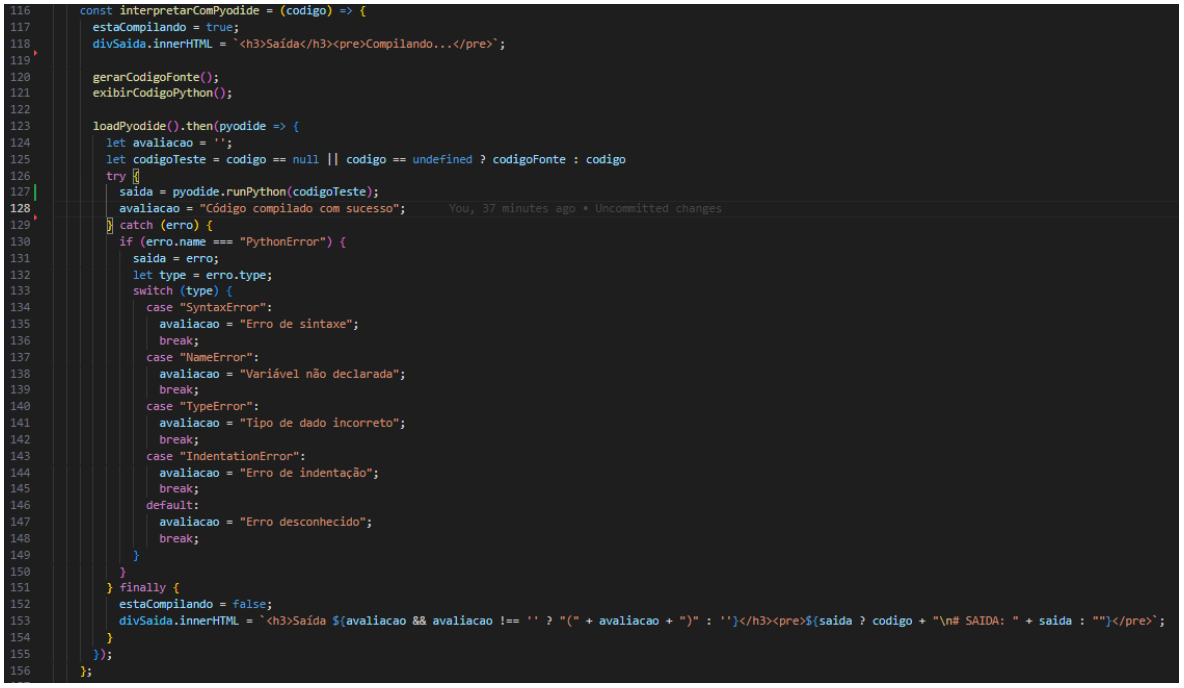
O botão “Editar” abre uma modal para edição dos atributos dos objetos com a intenção de permitir maior interatividade com os usuários e compreensão do escopo de cada objeto.

O botão “Excluir” permite apagar um objeto com o intuito que o estudante que estiver utilizando a ferramenta tenha maior controle sobre os objetos criados durante o processo de aprendizado.

5.7 Interpretador Pyodide

O Interpretador do Pyodide é o núcleo da execução de código Python na aplicação. Esse componente, baseado em WebAssembly, permite ao código Python ser executado diretamente no navegador, eliminando a necessidade de configuração de ambiente local. O Pyodide interpreta o código gerado, executando-o e retornando os resultados para a Interface do Usuário. Essa abordagem permite à aplicação oferecer uma experiência interativa, com feedback imediato ao usuário. Este componente é implementado pela função “interpretarComPython”, conforme observado na Figura 33.

Figura 33 - Implementação do componente interpretador Pyodide em Javascript



```
116  const interpretarComPyodide = (codigo) => {
117    estaCompilando = true;
118    divSaida.innerHTML = `<h3>Saída</h3><pre>Compilando...</pre>`;
119
120    gerarCodigoFonte();
121    exibirCodigoPython();
122
123    loadPyodide().then(pyodide => {
124      let avaliacao = '';
125      let codigoTeste = codigo == null || codigo == undefined ? codigoFonte : codigo
126      try {
127        saida = pyodide.runPython(codigoTeste);
128        avaliacao = "Código compilado com sucesso";      You, 37 minutes ago • Uncommitted changes
129      } catch (erro) {
130        if (erro.name === "PythonError") {
131          saida = erro;
132          let type = erro.type;
133          switch (type) {
134            case "SyntaxError":
135              avaliacao = "Erro de sintaxe";
136              break;
137            case "NameError":
138              avaliacao = "Variável não declarada";
139              break;
140            case "TypeError":
141              avaliacao = "Tipo de dado incorreto";
142              break;
143            case "IndentationError":
144              avaliacao = "Erro de indentação";
145              break;
146            default:
147              avaliacao = "Erro desconhecido";
148              break;
149          }
150        }
151      } finally {
152        estaCompilando = false;
153        divSaida.innerHTML = `<h3>Saída ${avaliacao && avaliacao !== '' ? "(" + avaliacao + ")" : ''}</h3><pre>${saida ? codigo + "\n# SAIDA: " + saida : ""}</pre>`;
154      }
155    });
156  };
```

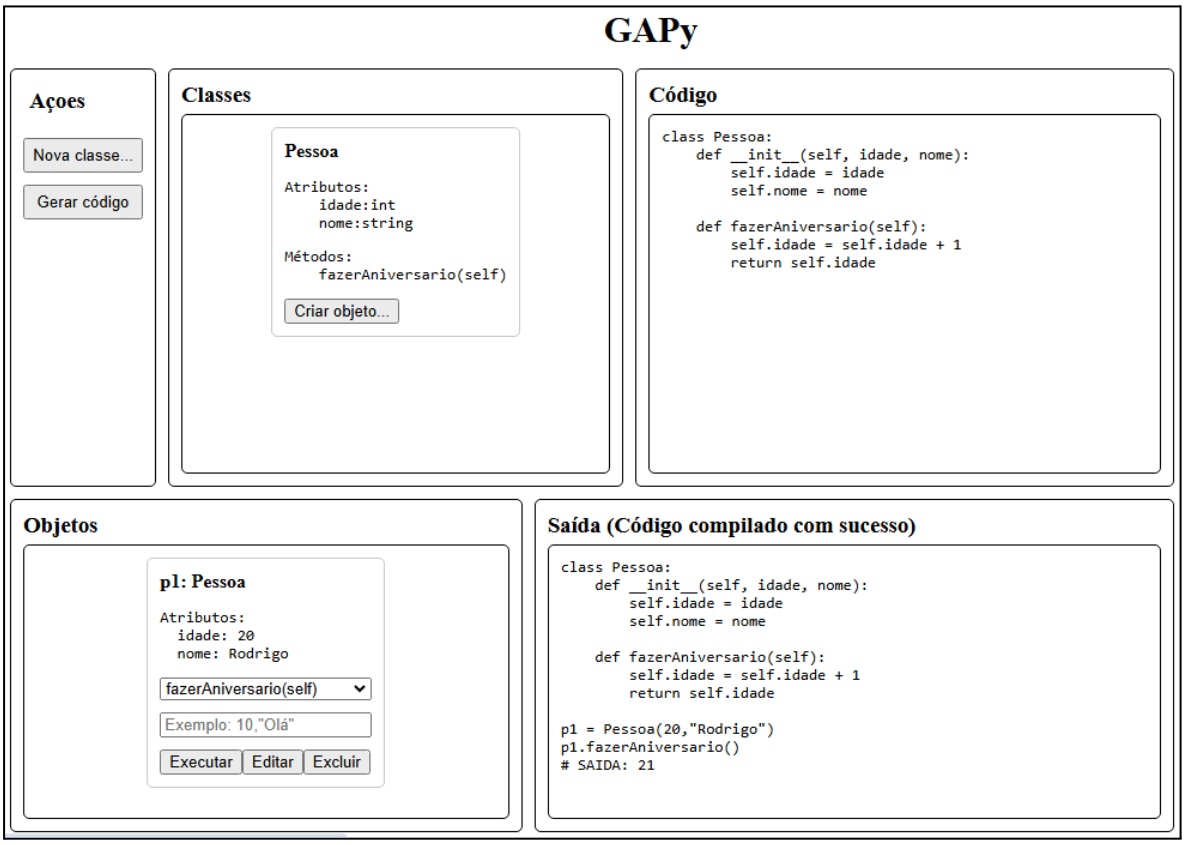
Fonte: elaborado pelo autor

Na linha 117 e 118, atualiza a interface sobre interpretação de código para fornecer um feedback ao usuário. A linha 120 e 121, respectivamente, invoca o componente “Gerador de código fonte” (descrito na subseção 5.4) e atualiza a interface com o código gerado. Em seguida na linha 123 é feito o carregamento do Pyodide usando a variável “pyodide” como representante da biblioteca. Na linha 124 é inicializado a variável “avaliacao” e na linha seguinte, verifica se o parâmetro é vazio armazenando o valor na variável “codigoTeste”.

Na linha 127 o “codigoTeste” começa a ser interpretado, esta linha é a seguinte estão envolvidas em um *try* pois a função *runPython* lança exceção caso haja falha na interpretação e desviando o fluxo para o *catch* onde há os erros já mapeados

Por fim, nas linhas 152 e 153 a interface é notificada das atualizações para que o componente visual “Saída” seja atualizado. Na figura 34 é apresentado um exemplo do funcionamento correto do componente.

Figura 34 - Exemplo de execução do GAPy



Fonte: elaborado pelo autor

5.8 Avaliação qualitativa comparativa de funcionalidades

A avaliação qualitativa demonstrada na tabela 2 compara a ferramenta desenvolvida Gapy com BlueJ e Thonny. Com foco nos objetivos, linguagens e funcionalidades diferentes. As ferramentas jGrasp e DrJava não serão levadas em consideração nesta avaliação devido suas restrições apontadas no critério de Suporte a Metodologias de Avaliação da tabela.

Tabela 2 - Avaliação qualitativa comparativa

Funcionalidade	BlueJ	GAPy	Thonny	Greenfoot
Linguagem	Java	Python	Python	Java
Tipo de Interface	Gráfica, com interação visual de objetos e classes.	Gráfica, com interação visual de objetos e classes.	Textual, voltada para o código e execução direta.	Gráfica, com interação visual de objetos e classes,
Criação de Objetos	Focado na manipulação de objetos, com suporte a visualizações.	Suporte com visualização e manipulação interativas	Ssem recursos visuais para POO.	Suporte com visualização e manipulação interativas em simulações.
Exploração de Classes	Visualização gráfica das interações entre classes.	Visualização gráfica das interações	Não possui visualização específica de classes.	Visualização gráfica em uma simulação.
Execução de Métodos	Execução direta de métodos em objetos	Execução direta de métodos em objetos.	Execução linear de código com suporte a breakpoints.	Execução direta de métodos em simulações visuais.
Facilidade de Instalação	Fácil de instalar, mas exige recursos do sistema.	Não é necessário instalação, rodar em qualquer navegador	Fácil de instalar, mas exige recursos do sistema.	Fácil de instalar, mas exige recursos do sistema.

Fonte: elaborada pelo autor

Cada uma das ferramentas apresenta pontos fortes que as tornam adequadas para diferentes cenários educacionais. GAPy combina a simplicidade de Python com elementos gráficos facilitadores do entendimento dos conceitos essenciais da POO, sendo uma excelente escolha para quem deseja explorá-los de forma prática e interativa. BlueJ é uma referência no ensino de POO em Java, proporcionando uma abordagem consolidada para instituições de ensino e estudantes avançados. Thonny é a opção mais acessível para iniciantes absolutos, mas seu uso pode ser limitado em contextos que exijam o ensino aprofundado de POO, além de ser ausente elementos visuais.

6. Conclusão

O presente trabalho propôs a criação do GAPy, um sistema educacional Web voltado para o ensino de programação orientada a objetos com Python, utilizando tecnologias como WebAssembly e Pyodide para execução de código diretamente no navegador.

O desenvolvimento do GAPy foi inspirado em ferramentas como BlueJ e Greenfoot, buscando integrar a interatividade e os elementos gráficos dessas plataformas com as vantagens de um ambiente Web. Essa abordagem foi projetada para minimizar as barreiras técnicas para os estudantes e maximizar a acessibilidade, eliminando a necessidade de instalação de software local.

A interface gráfica do GAPy permite a criação, manipulação e visualização de classes e objetos de forma interativa. Essa experiência prática reforça o aprendizado e promove o envolvimento dos estudantes com os conceitos fundamentais de POO.

Comparações qualitativas realizadas com BlueJ, Greenfoot e Thonny indicam que o GAPy apresenta vantagens significativas em relação à simplicidade de uso e à integração de recursos gráficos e pedagógicos. Enquanto o BlueJ se destaca como uma ferramenta consolidada para o ensino de POO em Java, e o Thonny como uma IDE acessível para iniciantes em Python, o GAPy combina características positivas de ambas as plataformas, adaptadas ao ensino de POO com Python em um ambiente Web.

A utilização de tecnologias recentes como WebAssembly permite uma execução eficiente do código e uma experiência fluida para o usuário. A integração com Pyodide também assegura a flexibilidade do sistema para interpretar código Python, mantendo a interface amigável e interativa.

As análises realizadas com o GAPy indicam que o sistema atende aos objetivos propostos, fornecendo uma plataforma robusta e acessível para o ensino de POO. O sistema destaca-se pela interação intuitiva e o suporte visual.

O GAPy representa um avanço significativo no desenvolvimento de ferramentas educacionais para programação. Sua abordagem prática, aliada à utilização de tecnologias modernas, oferece um ambiente inovador que facilita a aprendizagem de conceitos complexos de programação orientada a objetos.

Contudo, mesmo com resultados positivos, algumas limitações foram identificadas, como a dependência do Pyodide para interpretação e a ausência de recursos de tutoriais interativos. Essas questões oferecem oportunidades para evoluir a solução e ampliar sua aplicabilidade no ambiente educacional.

Para trabalhos futuros, recomenda-se expandir o GAPy com novos recursos, como depuração linha a linha, para melhor entendimento do funcionamento do interpretador, melhoria de interface de usuário para tornar o do sistema mais intuitivo, como alteração da forma de inserção de métodos e atributos na criação de classe. Essas funcionalidades poderiam melhorar o acompanhamento do progresso dos estudantes e fornecer dados úteis para análises educacionais.

Referências

ALLEN, E.; CARTWRIGHT, R.; STOLER, B. DrJava: A lightweight pedagogic environment for Java. SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education), v. 34, 2002. DOI: 10.1145/563340.563395.

BELANI, G. Programming Languages You Should Learn in 2020. Disponível em: https://www.computer.org/publications/tech-news/trends/programming-languages-you-should-learn-in-2020. Acesso em: 01 set. 2022.

BENNEDSEN, J.; CASPERSEN, M. E. Failure rates in introductory programming. ACM SIGCSE Bulletin, v. 39, n. 2, p. 32-36, 2007.

BLACK, C. WebAssembly: Opportunities and Challenges. In: Proceedings of the ACM Symposium on Web Technologies, p. 12-20, 2017

BROWN, N. C.; KOLLING, M.; MCCALL, D.; UTTING, I.; WARREN, I. Blackbox: A large scale repository of novice programmers' activity. In: Proceedings of the 45th ACM technical symposium on Computer science education, p. 223-228, 2014

DOE, J. Security Considerations in WebAssembly Development. IEEE Computer Society, p. 45-52, 2018

FARINELLI, F. Conceitos básicos de programação orientada a objetos. Instituto Federal Sudeste de Minas Gerais, 2007

FEENBERG, A. O que é a filosofia da tecnologia. Andrew Feenberg: racionalização democrática, poder e tecnologia, v. 3, p. 39-51, 2010. Disponível em: https://www.sfu.ca/~andrewf/books/Portug_O_que_e_a_Filosofia_da_Tecnologia.pdf

GARCIA, R. Exploring WebAssembly for High-Performance Computing Applications. In: Proceedings of the International Conference on Computer Science, p. 102-115, 2022

GOMES, A.; HENRIQUES, J.; MENDES, A. J. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. Educação, Formação & Tecnologias, v. 1, n. 1, p. 93-103, 2008.

GOOGLE. Disponível em: https://workspace.google.com/intl/pt-BR/features/. Acesso em: 2022

HAAS, A. et al. Bringing the web up to speed with WebAssembly. In: Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, p. 185-200, 2017

HAGAN, D. E.; MARKHAM, S. C.; SWITZER, M. J. Engaging Students in Computing through Contextualized Problems and Experiential Learning. Journal of Computing Sciences in Colleges, v. 35, n. 3, p. 87-95, 2020.

HATTIE, J.; TIMPERLEY, H. The Power of Feedback. Review of Educational Research, v. 77, n. 1, p. 81-112, 2007.

HENRIKSEN, P.; KÖLLING, M. Greenfoot: Combining object visualization with interaction. In: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, p. 108-112, 2004.

JADUD, M. C. Methods and tools for exploring novice compilation behaviour. In: Proceedings of the second international workshop on Computing education research, p. 73-84, 2006.

JOHNSON, B. Challenges in Adopting WebAssembly for Web Applications. Journal of Web Development, v. 30, n. 2, p. 112-125, 2022.

JOHNSON, B.; RAASCH, C.; SEFFAH, A. Teaching Object-Oriented Programming to Novices: Towards a Problem-Solving Taxonomy. ACM Transactions on Computing Education (TOCE), v. 16, n. 3, p. 1-25, 2016.

JOHNSON, T. Programming environments in computer science education: A systematic review. ACM Computing Surveys, v. 51, n. 3, Article 56, 2018.

JONES, R.; BROWN, S. WebAssembly: A New Paradigm in Web Development. IEEE Transactions on Web Technologies, v. 18, n. 1, p. 55-68, 2020.

KÖLLING, M. The Greenfoot programming environment. ACM Transactions on Computing Education (TOCE), v. 10, n. 4, p. 1-21, 2010.

KÖLLING, M.; ROSENBERG, J. Guidelines for teaching object orientation with Java. In: Sixth Conference on Innovation and Technology in Computer Science Education (ITiCSE 2001), Canterbury, UK, 2001.

KÖLLING, M.; QUIG, B.; PATTERSON, A.; ROSENBERG, J. The BlueJ System and its Pedagogy. Computer Science Education, v. 13, n. 4, p. 249–268, 2003. DOI: 10.1076/csed.13.4.249.17496.

PAPERT, S. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, 1980.

SPECTRUM. Top Programming Languages 2024. Disponível em: https://spectrum.ieee.org/top-programming-languages-2024. Acesso em: 8 dez. 2024.

YADAV, N.; DEBELLO, J. E. Recommended practices for Python pedagogy in graduate data science courses. In: 2019 IEEE Frontiers in Education Conference (FIE). IEEE, 2019. p. 1-7. DOI: https://doi.org/10.1109/FIE.2019.123456.

ZANETTI, H.; BORGES, M.; RICARTE, I. Pensamento computacional no ensino de programação: Uma revisão sistemática da literatura brasileira. In: Brazilian Symposium on Computers in Education (SBIE), p. 21, 2016.

APENDICE A - Repositório da aplicação

A aplicação desenvolvida neste trabalho pode ser encontrada no seguinte repositório do GitHub: https://github.com/rjnunesdev/rjnunesdev.github.io

APENDICE B - Artigo do TCC

Sistema educacional Web para ensino de programação orientada a objeto com Python

Rodrigo Joaquim Nunes[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.010-970 – Florianópolis – SC – Brazil

rodrigo.j.nunes@grad.ufsc.br

Abstract. This work proposes the development of GAPy, a web-based educational system designed to facilitate the teaching of Object-Oriented Programming (OOP) using the Python programming language. The system combines modern technologies such as WebAssembly and Pyodide to enable code execution directly in the browser, eliminating the need for local software installation and providing an interactive and accessible learning experience. GAPy's graphical interface allows students to intuitively create, manipulate, and visualize classes and objects, reinforcing the learning of fundamental OOP concepts. Qualitative comparisons with traditional OOP teaching tools, such as BlueJ, Greenfoot and Thonny, reveal that GAPy offers significant advantages in terms of accessibility and pedagogical resources. This paper describes the development and evaluation of GAPy, discussing its main features, benefits, and limitations.

Resumo. Este trabalho propõe o desenvolvimento do GAPy, um sistema educacional Web projetado para facilitar o ensino de programação orientada a objetos (POO) utilizando a linguagem Python. O sistema combina tecnologias modernas como WebAssembly e Pyodide para permitir a execução de código diretamente no navegador, eliminando a necessidade de instalação de software local e proporcionando uma experiência de aprendizado interativa e acessível. A interface gráfica do GAPy permite que os estudantes criem, manipulem e visualizem aulas e objetos de maneira intuitiva, reforçando a aprendizagem dos conceitos fundamentais de POO. Comparações qualitativas com ferramentas tradicionais de ensino de POO, como BlueJ, Greenfoot e Thonny, revelam que o GAPy oferece vantagens significativas em termos de acessibilidade e recursos pedagógicos. Este artigo descreve o desenvolvimento e a avaliação do GAPy, discutindo suas principais funcionalidades, benefícios e limitações.

1. Introdução

O ensino de programação tem se expandido em diferentes níveis educacionais, destacando-se nos cursos tecnológicos, onde conceitos básicos e avançados são progressivamente incluídos,

apesar das dificuldades iniciais enfrentadas por muitos estudantes (BELANI, 2020; GOMES, HENRIQUES & MENDES, 2008).

Paradigmas como o orientado a objetos, que abstraem elementos do mundo real de forma estruturada, e linguagens como Python, reconhecidas por sua simplicidade e aplicabilidade pedagógica, têm papel central nesse processo (FARINELLI, 2007; YADAV & DEBELLO, 2019; BELANI, 2020) .

No âmbito do desenvolvimento Web, o WebAssembly surge como uma solução eficiente e segura para substituir sistemas on-premise, reforçando a necessidade de ferramentas educacionais especializadas (HAAS et al., 2017). Com base nesse contexto, este artigo propõe a criação de um sistema educacional em WebAssembly para auxiliar no ensino de programação orientada a objetos com Python, buscando atender estudantes universitários iniciantes na área.

O GAPy (Sistema Educacional Web para Ensino de POO com Python) foi desenvolvido com o objetivo de superar essas barreiras, oferecendo uma plataforma baseada em navegador que elimina a necessidade de instalação de software e oferece uma experiência de aprendizado intuitiva. Utilizando tecnologias como WebAssembly e Pyodide, o GAPy permite a execução do código Python diretamente no navegador, oferecendo uma solução acessível, eficiente e interativa para o ensino de POO.

Este artigo apresenta a proposta e o desenvolvimento do GAPy, abordando as tecnologias utilizadas, a estrutura do sistema, a avaliação qualitativa realizada e os benefícios que o sistema oferece no contexto educacional.

2. Objetivos

Este trabalho tem como objetivo desenvolver um sistema educacional em ambiente Web, focado no ensino de programação orientado a objetos com a linguagem Python, utilizando WebAssembly para execução de código diretamente no navegador. Para isso, será realizada uma análise do estado da arte sobre ensino de programação, ambientes de aprendizagem e a aplicabilidade do WebAssembly. O estudo também buscará identificar as abordagens pedagógicas mais adequadas para o ensino de Python e, a partir dessa base teórica, desenvolver um protótipo do sistema educacional. Por fim, será realizada uma análise qualitativa comparativa entre a ferramenta desenvolvida e outras como BlueJ, Greenfoot e Thonny, avaliando sua usabilidade, eficácia no ensino e desempenho no contexto de aprendizagem de programação orientada a objetos.

2. Metodologia

O desenvolvimento do GAPy contou com etapas claras para garantir sua eficácia educacional. Primeiramente, foi realizada uma análise do estado da arte sobre ensino de programação, paradigmas pedagógicos e tecnologias de suporte como WebAssembly e Pyodide.

A interface do GAPy foi projetada para permitir a criação, manipulação e visualização de classes e objetos em um ambiente interativo. Utilizando WebAssembly, o desempenho do sistema garante alto desempenho na execução de código, enquanto o Pyodide garante flexibilidade à interpretação do Python diretamente no navegador.

Além disso, o GAPy foi avaliado qualitativamente em comparação com ferramentas educacionais como BlueJ, Greenfoot e Thonny, considerando aspectos como simplicidade de

O GAPy foi desenvolvido com o objetivo de criar uma plataforma educacional acessível e interativa para o ensino de POO com Python. A estrutura do sistema foi planejada para facilitar a criação, manipulação e visualização de aulas e objetos, proporcionando uma experiência de aprendizado prático e intuitivo. O sistema é composto por diversas funcionalidades interativas, como a criação de classes e objetos, a execução de métodos e a exibição de resultados de forma visual e intuitiva.

3.1 Arquitetura do Sistema

O GAPy é composto por vários módulos, cada um responsável por uma parte específica do processo de ensino. A arquitetura do sistema inclui os seguintes componentes principais:

1. **Interface de Usuário** : Responsável pela interação com os usuários, permitindo a entrada de dados e a exibição de resultados. A interface é baseada em HTML, CSS e JavaScript.
2. **Gerador de Classe** : Cria e valida as definições de classe a partir dos dados fornecidos pelos usuários.
3. **Gerador de Código Fonte** : Converte as definições de classe em código Python repetidamente.
4. **Gerador de Representação de Classe** : Gera representações visuais das aulas para facilitar a compreensão.
5. **Gerador de Objeto** : Permite a criação de instâncias de classes com atributos e métodos definidos pelos usuários.
6. **Gerador de Representação de Objeto** : Cria representações visuais de objetos instanciados.
7. **Interpretador Pyodide** : Interpreta o código Python gerado e executado diretamente no navegador.
8. **Saída** : Exibe os resultados da execução do código, incluindo erros e saídas geradas pelos métodos executados.

3.2 Fluxo de Execução

O fluxo de execução do sistema começa com o usuário definindo uma nova classe, informando seu nome, atributos e métodos. Esses dados são validados e utilizados para gerar uma estrutura de classe e o código Python correspondente. A partir da criação da classe é permitido criar um novo objeto. O sistema gera representações visuais de classes e objetos, que são exibidas ao usuário, permitindo uma compreensão clara das interações entre os elementos.

Após a criação de classes e objetos, o usuário pode executar métodos sobre os objetos instanciados. O sistema gera o código Python correspondente, que é executado e executado diretamente no navegador usando o Pyodide. A execução é acompanhada por feedback visual e textual, permitindo que os estudantes vejam os resultados de maneira clara e imediata.

4. Resultados e Discussão

A avaliação qualitativa do GAPy foi realizada por meio de uma comparação com ferramentas tradicionais de ensino de POO, como BlueJ, Greenfoot Thonny. O GAPy se destaca em diversas áreas, especialmente em termos de acessibilidade e recursos gráficos. A capacidade de rodar diretamente no navegador, sem a necessidade de instalação, foi apontada como um grande diferencial.

Embora BlueJ, Greenfoot e Thonny sejam eficazes para o ensino de POO, ambos desativam instalação local e configurações complexas, o que pode ser um desafio para alunos com pouca experiência técnica. O GAPy, por sua vez, oferece uma solução mais acessível, que pode ser utilizada em qualquer dispositivo com acesso à internet.

Os resultados das avaliações indicaram que o GAPy oferece uma interface intuitiva, com recursos gráficos que facilitam a compreensão dos conceitos de POO. A interação visual com as classes e objetos, combinada com o feedback imediato sobre a execução do código, foi vista como um recurso pedagógico poderoso, especialmente para iniciantes de programação.

3. Conclusão

O GAPy representa uma inovação significativa no campo das ferramentas educacionais para o ensino de programação orientada a objetos. Sua abordagem prática, aliada ao uso de tecnologias como WebAssembly e Pyodide, fornece um ambiente interativo e acessível que facilita o aprendizado de conceitos complexos de POO. Comparado a outras ferramentas tradicionais, o GAPy se destaca pela sua simplicidade de uso e pela capacidade de executar código diretamente no navegador, sem a necessidade de instalação de software local.

Embora os resultados sejam promissores, o sistema apresenta algumas limitações, como a falta de tutoriais interativos e a dependência do Pyodide para a execução do código. Melhorias futuras, como a inclusão de depuração linha a linha e o aprimoramento da interface do usuário, poderiam aumentar ainda mais a eficácia do GAPy como ferramenta educacional.

Em trabalhos futuros, recomendamos expandir as funcionalidades do sistema, incluindo a possibilidade de depuração em tempo real e o suporte a conceitos avançados de POO, como herança e polimorfismo. Além disso, seria interessante realizar uma avaliação prática com estudantes de diferentes níveis de conhecimento para medir a eficácia do sistema em contextos educacionais reais.

4. Referências

- BELANI, G. Programming Languages You Should Learn in 2020. Computer Society, 2020.
- GARCIA, R. Exploring WebAssembly for High-Performance Computing Applications. International Conference on Computer Science, 2022.
- HAAS, A. et al. Bringing the Web up to Speed with WebAssembly. Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, 2017.
- HAGAN, D. E., MARKHAM, S. C., & SWITZER, M. J. Engaging Students in Computing through Contextualized Problems and Experiential Learning. Journal of Computing Sciences in Colleges, 2020.
- JOHNSON, B., RAASCH, C., & SEFFAH, A. Teaching Object-Oriented Programming to Novices: Towards a Problem-Solving Taxonomy. ACM

Transactions on Computing Education, 2016.

- KAFAI, Y. B., & RESNICK, M. Constructionism in Practice: Designing, Thinking, and Learning in a Digital World. Lawrence Erlbaum Associates, 1996.
- KOELLING, M. The BlueJ System and its Pedagogy. Computer Science Education, 2003.
- LEA, M., et al. Enhancing Web Application Performance with WebAssembly. ACM Transactions on Internet Technology, 2021.
- YADAV, N.; DEBELLO, J. E. Recommended practices for python pedagogy in graduate data science courses. 2019 IEEE Frontiers in Education Conference (FIE), IEEE, 2019.