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

Fran Adalid Cardoso Alvarez

FERRAMENTA WEB DE AUXÍLIO À AVALIAÇÃO DE APRENDIZAGEM ONLINE.

Florianópolis
2019

Fran Adalid Cardoso Alvarez

FERRAMENTA WEB DE AUXÍLIO À AVALIAÇÃO DE APRENDIZAGEM ONLINE.

Trabalho de conclusão de curso submetido à banca examinadora da Universidade Federal de Santa Catarina para a obtenção do título de bacharel em Sistemas da Informação.
Orientador: Prof. José Eduardo De Lucca

Florianópolis
2019

Fran Adalid Cardoso Alvarez

FERRAMENTA WEB DE AUXÍLIO À AVALIAÇÃO DE APRENDIZAGEM ONLINE.

O presente trabalho em nível de bacharel foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. José Eduardo De Lucca
Orientador

Prof. Dra. Fabiane Barreto Vavassori Benitti.
Membro da banca

Prof. José Francisco Danilo de Guadalupe Correa Fletes
Membro da banca

Florianópolis, 26 de Novembro de 2019.

Este trabalho é dedicado a minha família, namorada, amigos, e a todos que me fizeram ser quem eu sou.

AGRADECIMENTOS

A minha mãe que me apoiou desde o começo da minha existência. Deixou muitas vezes de ter para dar a mim e aos meus irmãos. Se um dia puder ser metade daquilo que você sonhou para mim, me considero realizado.

A meu pai que apesar de distante algumas vezes, sempre amou aos filhos e fez o possível para nos dar a melhor vida.

Aos meus irmãos que sempre estiveram do meu lado e vivem o significado de família.

A minhas tias, em especial minha madrinha Ana, que sempre apoiou meus estudos e me fez adquirir tantos bons hábitos, como o da leitura.

A minha namorada Catherine que sempre me apoiou em todos os momentos e é possivelmente a maior torcedora do meu sucesso, assim como eu sou do dela.

E finalmente a todos meus amigos que me fazem sorrir e dão bons conselhos e vivências durante minha caminhada.

“A educação é a arma mais poderosa
que você pode usar para mudar o mundo.”
(Nelson Mandela, 2003)

RESUMO

O mundo e suas relações estão em constantes mudanças, a educação acadêmica é uma delas. Inúmeros cursos a distância tem surgido para suprir a demanda de aulas que não requerem a presença física em uma sala de aula tradicional, com quadro na parede, apresentação de slides, lista de chamada, provas e testes. Isto tudo já pode ser feito através de tecnologias em rede. A oferta de cursos a distância que tem sua grade curricular totalmente via internet tem aumentado muito, e também cursos presenciais têm se beneficiado de tecnologias de educação à distância para dinamizar ou melhorar algum processo de ensino. Inserido nesse contexto, este trabalho foi iniciado com a proposta de trazer estudos sobre uma das etapas de ensino, a avaliação. Pesquisar sobre aplicativos de avaliação no mercado que tem uma proposta educacional e ao final desenvolver uma ferramenta com características levantadas que visam aumentar o engajamento e performance dos alunos em sua vida acadêmica durante seus anos de formação. A ferramenta foca na parte de avaliação, ser interativa e com design atrativo. Também procurou-se dar uma experiência boa ao professor, que será o responsável por alimentar um banco de questões para que seus alunos utilizem de maneira a se preparar melhor para as aulas. Por fim, espera-se ter se desenvolvido uma aplicação que ajude no processo de avaliação, que pode ser usada tanto para testes preparatórios às tradicionais provas escritas, ou por que não, ser o próprio método de avaliação adotado pelos professores.

Palavras-chave: Educação à distância. Ferramenta online. Avaliação. Experiência do usuário.

ABSTRACT

The world and its relations are constantly changing, academic education is one of them. Numerous distance learning courses have sprung up to meet the demand for classes that do not require physical presence in a traditional classroom, with wall-mounted chalkboard, slideshow, call list, exams and tests. This can all be done through online technologies. The offer of distance learning courses that have their curriculum entirely via the internet has increased a lot, and also in-person courses have benefited from distance education technologies to streamline or improve some teaching process. Inserted in this context, this work started with the proposal to bring studies about one of the teaching stages, the evaluation. Understand how university assessment is done and ultimately develop a tool with raised characteristics that aim to increase the engagement and performance of students in their academic life during their formative years. The tool focus on the evaluation, be interactive and attractive design. We also tried to give a good experience to the teacher, who will be responsible for feeding a bank of questions for his students to use in order to better prepare for classes. Finally, it is expected to have developed an application that helps in the assessment process, which can be used for both preparatory tests and traditional written tests, or why not be the very evaluation method adopted by teachers.

Keywords: Distance Education. Website. Evaluation. User experience.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AICC	Aviation Industry Computer-based training Committee
API	Application Programming Interface ou Interface de Programação de Aplicativos
AVA	Ambiente Virtual de Aprendizagem
EaD	Educação a Distância
INEP	Instituto Nacional de Estudos e Pesquisas
LMS	Learning Management Systema
PDA	Personal Digial Assistences

SUMÁRIO

1 INTRODUÇÃO

Segundo dados do censo da educação superior divulgado pelo INEP (2017), mais de 46% das matrículas de cursos tecnológicos no Brasil já são a distância, que pelo estudo é explicado principalmente pelo aumento na oferta dessa modalidade nos últimos anos, que entre 2007 e 2017 cresceu 586% em relação ao também positivo aumento de número de matrículas de cursos presenciais. A educação a distância "trata-se de um domínio da educação em que as tecnologias são fundamentais pois, quer a transmissão de conteúdos quer a própria relação pedagógica, têm que ser mediatizadas de forma a ultrapassar as barreiras do espaço e do tempo, que separam professor e alunos (formador e formandos)" (GOMES, 2005, p. 233). Existem muitas facetas e conceitos atribuídos à educação a distância , que ao longo deste trabalho serão apresentados a modo de melhor conceituar esse método de ensino que é tão amplo.

Podemos relacionar a expansão da EaD com os benefícios que ela proporciona, que segundo Valentine (1997) podem ser classificados em três principais categorias: Custo-benefício, maior impacto por meio de mídias interativas atualizadas em tempo real e maior variedade de canais de comunicação entre conteúdo e aluno. Conforme a população cresce e fica mais conectada gera tanto a demanda em ter mais opções e disponibilidade de cursos de graduação, como o meio de suprir essa procura. E os dispositivos para isso podem ser os mais variados, desde computadores em um laboratório de informática, até um smartphone com dados móveis. O que ocorre é que a sociedade atualmente está sempre conectada, trocando mensagens, acessando algum recurso como jornal online ou estudando. "O ensino a distância é um sistema tecnológico de comunicação bidirecional, que pode ser de massa e que substitui a interação pessoal entre professor e aluno na sala de aula, como meio preferencial do ensino, pela ação sistemática e conjunta de diversos recursos didáticos e pelo apoio de uma organização e tutoria que propiciam a aprendizagem autônoma dos estudantes" (IBÁNEZ, 1996, p. 10).

EaD é uma inovação, contudo existem pontos bons e ruins nesse modelo de ensino. Entre os bons Rover (2003), Bruno (2001) e Takahashi (2000) destacam:

- Aprendizado personalizado.
- Variedade nas opções de troca de conhecimento.
- Flexibilidade no uso do tempo disponível pelo usuário.
- Critérios de avaliação e aproveitamento variados e alguns processos de avaliação automatizados.
- Menor custo com o uso da internet.

- Acesso universal.
- Centralização da administração da qualidade do conteúdo.
- Menor chance de erro na entrega do material digital.
- Aumento do público alcançado e possibilidade de troca de informação entre instituições de ensino.
- Maior independência de locais e horários.
- Maior facilidade de se trabalhar em equipe.

É importante salientar que estes pontos necessitam um uso adequado da tecnologia envolvida que proporcione o melhor aproveitamento da troca de conhecimento entre os alunos e professores. Contudo, existem problemáticas e desafios relacionados a EaD nos aspectos teóricos e práticos. Nos teóricos a maioria advém do próprio preconceito por parte desse modelo de ensino e Rover (2003), Vargas (1994) e Benakouche (2000) destacam os seguintes pontos:

- Desconhecimento do significado do modelo.
- Desconhecimento das características atuais da sociedade.
- Desconhecimento das possibilidades do EaD.
- Não possuir a cultura de EaD, ou ter experienciado de maneira burocrática.

E sobre os aspectos práticos da dificuldade do uso da EaD, Rover (2003) e Rodrigues (2002) pontuam:

- Não se ter um planejamento da atividade de produção e/ou da transmissão do conteúdo, ou seja, não dar o real valor à tecnologia e a necessidade de adaptação de uso de novas mídias.
- Falta de preparação de seu segmento.
- Falta de dimensão de custos.
- Inexistência de estruturas de gerenciamento de conteúdos e métodos de avaliação.
- Falta de pessoal que tenha qualificações técnicas que saibam gerenciar as exigências e potências da EaD.

O processo de avaliação na educação é onipresente, existe em qualquer sistema de ensino, seja ele a distância ou não. Segundo Piletti (1987, p. 190): “Avaliação é um processo contínuo de pesquisas que visa interpretar os conhecimentos, habilidades e atitudes dos alunos, tendo em vista mudanças esperadas no comportamento, propostas nos objetivos educacionais, a fim de que haja condições de decidir sobre alternativas do planejamento do trabalho do professor e da escola como um todo”. O processo de avaliação tem como fim melhorar a qualidade do processo de aprendizagem, podendo ter basicamente três funções: Função diagnóstica, função formativa e função somática (PILETTI, 1987; HAYDT, 2002).

Por função diagnóstica da avaliação entende-se o intuito de identificar o nível de conhecimento a quem é aplicada, para se ter ideia dos níveis de conhecimento individuais e grupais dos alunos. É aquela avaliação geralmente feita no começo do curso ou unidade de ensino para identificar se os alunos possuem conhecimentos básicos para a aprendizagem de novos conteúdos, podendo servir também para estimar problemas de aprendizagens e suas causas (HAYDT, 2002).

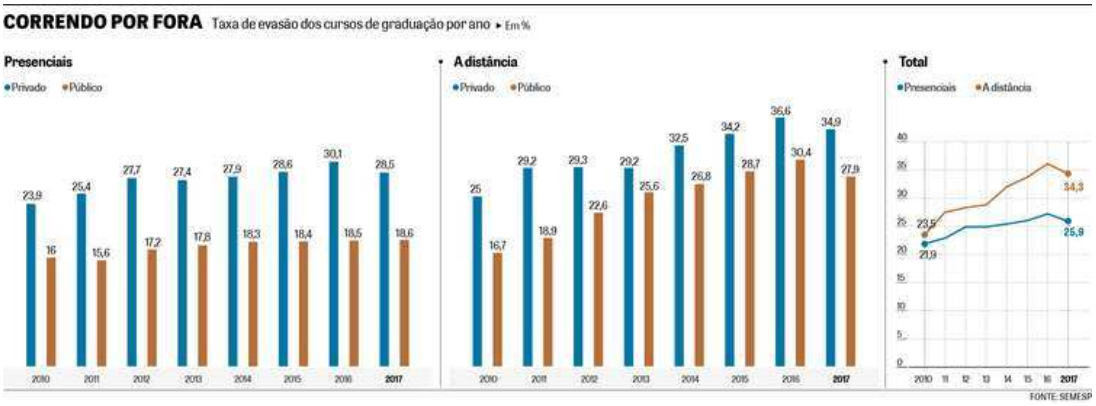
A função formativa da avaliação é aquela aplicada durante o processo de ensino, como o semestre letivo de uma disciplina. É uma forma de controle que visa informar o professor do rendimento do aluno, possíveis deficiências na organização do ensino e também potenciais ajustes que podem ser feitos para atingir os objetivos estipulados pelo plano de ensino (ALMEIDA, L., 2001). “Esse tipo de avaliação é basicamente um orientador dos estudos e esforços dos professores e alunos no decorrer desse processo, pois está muito ligada ao mecanismo de retro-alimentação (feed- back) que permite identificar deficiências e reformular seus trabalhos, visando aperfeiçoá-los em um ciclo contínuo e ascendente.” (SANTOS, 2005, p. 2).

Por fim a função somativa é aquela efetuada ao final do curso, período letivo ou unidade de ensino, que tem como intuito de classificar os alunos quanto ao nível de conhecimento adquiridos durante tal período. Comumente tem o fim de promover o aluno de um nível a outro (HAYDT, 2002).

1.1 JUSTIFICATIVA

A evasão de cursos superiores é um número que cresce a cada ano tanto na modalidade presencial como a distância, mesmo com o aumento de instituições que oferecem ingressos a cursos de graduação. Entender o porquê dessas desistências e o que pode ser feito para diminuir esse comportamento é de extrema importância para construir uma sociedade educada e com acesso ao conhecimento.

Figura 1 – Taxa de evasão de cursos graduação 2010-2017.



Fonte: (CRISTIANA, 2019)

Segundo Silva Filho *et al.* (2007, p. 643) “As expectativas do aluno em relação à sua formação e a própria integração do estudante com a instituição constituem, na maioria das vezes, os principais fatores que acabam por desestimular o estudante a priorizar o investimento de tempo ou financeiro, para conclusão do curso. Ou seja, ele acha que o custo benefício do “sacrifício” para obter um diploma superior na carreira escolhida não vale mais a pena.” Em outra ponta, que podem até mesmo disputar o tempo disponível pelos estudantes, existem os aplicativos e sites dos mais diferenciados gêneros e conteúdos (comédia, jogos, chats, notícias, etc.) que são objetos de acesso de milhões de usuários todos os dias. Os conteúdos na maioria das vezes são curtos e podem ser consumidos em poucos minutos, o fácil acesso também é um fator que explica o grande acesso dessas mídias. O desafio é saber aliar estas tecnologias a conteúdos relevantes de para auxiliarem na formação acadêmica.

O processo de avaliação é foco do presente trabalho, que pretende disponibilizar uma ferramenta que auxilie nessa etapa do processo de ensino-aprendizagem que segundo Gonçalves (1997, p. 7-16) “A avaliação, não importa a missão que se lhe proponha cumprir, parece ter o dom de despertar nas pessoas suas defesas mais escondidas. É, na educação, um processo revestido de rituais complexos, que resulta por torná-la um mito. No caso da avaliação da aprendizagem, tal mitificação ao invés de possibilitar às pessoas maior consciência de como está se desenvolvendo internamente o processo de construção do conhecimento, termina por confundi-las, tornando-as dependentes de algum veredicto externo que determine se estão aprendendo ou não”.

Inspirado no desafio que a educação superior enfrenta em motivar e engajar seus alunos, e os aplicativos da internet que ganham mais e mais usuários a cada dia, foi que esse trabalho se motivou. Se beneficiar da tecnologia para propor uma ferramenta que auxilie no engajamento e evolução dos alunos em disciplinas variadas.

Utilizando estudos que já se propuseram a analisar os motivos da evasão em cursos do ensino superior, e também estudos que analisam características de aplicativos educacionais que levam ao maior número de usuários satisfeitos e motivados e terem um uso cada vez maior. Utilizando mais uma vez a tecnologia para tentar curar uma dor, nesse caso, no contexto acadêmico. “A proliferação da mineração de dados software e desenvolvimentos na educação on-line, sistemas móveis de aprendizado e gerenciamento de aprendizado estão se unindo para ambientes de aprendizado que alavancam softwares de análise e visualização para retratar os dados de aprendizagem de forma multidimensional e maneira portátil. Nos cursos online e combinados, os dados podem revelar como as ações dos alunos contribuem para o seu progresso e ganhos específicos de aprendizagem” (BECKER *et al.*, 2017, p. 14).

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver uma aplicação web que possibilite a criação de atividades interativas a fim de avaliar instantaneamente o conhecimento dos alunos quanto a conteúdos vistos em sala de aula.

1.2.2 Objetivos Específicos

- Pesquisar e analisar aplicativos educacionais de sucesso no mercado.
- Implementar uma aplicação web de atividades que se baseie na simplicidade e boa experiência de uso.
- Oferecer subsídios aos professores para analisarem o desempenho dos alunos.

2 LEVANTAMENTO BIBLIOGRÁFICO

Neste capítulo serão trazidos os principais conceitos que englobam o assunto do trabalho e servem como pilar para o desenvolvimento da ferramenta que é o produto final a ser desenvolvido. Serão abordados conceitos desde educação a distância, até tópicos mais específicos como web desenvolvimento e tecnologias relacionadas, usabilidade de aplicativos, e subcategorias da EaD.

2.1 EDUCAÇÃO A DISTÂNCIA

A educação a distância é um recurso de incalculável importância como modo apropriado para atender a grandes contingentes de alunos de forma mais efetiva que outras modalidades e sem riscos de reduzir a qualidade dos serviços oferecidos em decorrência da ampliação da clientela atendida (NUNES, 1993). É um termo que já é usado há bastante tempo, mas foi no começo do século XXI que tomou mais notoriedade, exatamente por ter um aumento no número de praticantes e institutos que oferecem essa metodologia de ensino.

A história da EaD já tinha vestígios no século XVIII com as aulas de carta por correspondência, e hoje é marcada pelo uso de computadores e internet que são capazes de conectar alunos e professores em todos os lugares do mundo. Houve um grande desenvolvimento nessa modalidade, que hoje em dia já conta com faculdades totalmente a distância, onde as aulas são transmitidos por vídeos pré gravados das aulas, junto de um material de apoio como os slides da aula ou algum texto complementar. As avaliações são feitas através das plataformas conhecidas como LMS (Learning Management System) disponibilizam os conteúdos de aula bem como atividades de reforço e avaliação das disciplinas. As notas podem ser disponibilizadas em tempo real assim que o aluno termina a prova, ou também existe a possibilidade de provas dissertativas que são corrigidas pelo professor ou um tutor encarregado e entregue as notas posteriormente.

Educação a distância é um termo amplo e que ramifica outros grupos, alguns que condizem com o tema deste presente trabalho serão apresentados a seguir.

2.1.1 E-learning

O e-Learning é uma modalidade da educação a distância que faz uso da internet para distribuição de conteúdos e interação entre alunos e professores (ALMEIDA, M. E. B. de, 2003). Apesar de serem termos comumente utilizados juntos, educação a distancia e e-learning têm diferenças. A educação a distância pode ser utilizada por diferentes meios como correspondência postal, rádio, televisão, telefone, fax, e outras ferramentas para trazer o uso de técnicas de comunicação entre alunos e professores

distantes geograficamente.

E-learning é uma especialização EaD que utiliza a internet como meio de comunicação entre os agentes envolvidos. Beneficia-se também das chamadas TIC's (Tecnologias da informação e comunicação) para distribuir conteúdos interativos disponibilizados pelo acesso de computadores e outros dispositivos conectados a internet.

2.1.2 Microlearning

Microlearning é a entrega de uma informação de forma curta e rápida. Diferente do seu contraponto, o macrolearning, que propõe iniciativas maiores para formação de conhecimento, os microlearnings são como pílulas de conhecimento, com seu conteúdo mais encurtado para justamente serem consumidos em tempos em torno de 5 a 10 minutos.

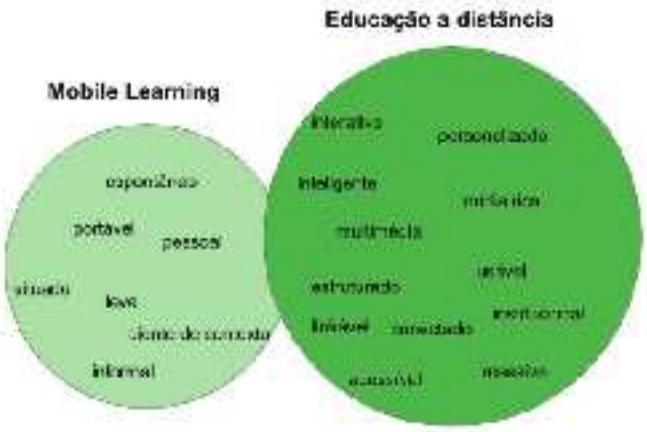
Criar, publicar e compartilhar micro conteúdos na internet abriu novas possibilidades para formas de aprendizagem informais, implícitas e incidentais, como o microlearning, o termo sendo referenciado a pequenas atividades de aprendizagem com micro conteúdos (LINDNER, 2006; ROBES, 2009; HUG, 2010).

Segundo Buchem e Hamelmann (2010), exemplos de microlearning incluem podcasts, conteúdos de blog, paginas de wikis, ou até mesmo pequenas mensagens no Facebook e Twitter.

2.1.3 Mobile learning

A sociedade está em constante movimento, aprendemos e adquirimos conhecimento em um lugar e aplicamos esse conhecimento em outros tantos (SHARPLES *et al.*, 2009). Por esse e outros motivos que os celulares móveis ganharam um importante papel no desenvolvimento de novas tecnologias para a educação a distância. Segundo Traxler (2005, p. 262) "Mobile Learning pode talvez ser definido como 'qualquer fornecimento educacional onde a única ou dominante tecnologia são dispositivos de mão'. Essa definição pode significar que mobile learning inclui dispositivos móveis, smartphones, PDA, tablets e similares". O autor comenta também que existe uma confusão com mobile learning e educação a distância, que apesar de terem aspectos em comum, mobile learning é um fragmento menor do aspecto total que seria a educação a distância, como a imagem abaixo demonstra:

Figura 2 – mobile vs e-learning



Fonte: (TRAXLER, 2005)

Segundo um artigo publicado pela empresa de tecnologia Cisco (2016), 70% da população mundial farão o uso de dispositivos móveis no ano de 2020, o que equivale a quase 5,5 bilhões de pessoas. Uma análise feita em pelo site especializado em estatísticas de uso da web StatCounter (2019) apoia esse fator, atestando que cerca de 60% da população mundial já navega na internet por meio de dispositivos móveis. Este é um setor que merece o olhar atento, pois oferece um enorme campo para áreas como a educação se beneficiarem direcionando conteúdos e encontrando soluções otimizadas para estes dispositivos. Um estudo feito em uma universidade sul-coreana, mostrou que estudantes que trabalham em tempo integral tem 48% a mais de chance de usarem dispositivos móveis do que aqueles que não trabalham, possibilitado pela facilidade de acesso a conteúdos de aula enquanto estão em trânsito (HAN; SHIN, 2016).

2.1.4 Blended learning

O ensino hibrido, que é mais comumente conhecido pelo seu termo no inglês Blended learning, é justamente a mistura de técnicas do ensino presencial com o e-learning. Staker e Horn (2012) definem este tipo de ensino como um programa de educação formal onde existem momentos que os alunos acessam conteúdos e instruções online, e outros que estão em sala de aula e interagem entre si e o professor.

2.1.5 Gamificação

Gamificação, ou no inglês gamification, apesar de ter a palavra game (jogo) como raiz, e de fato possuir relação com o termo, não tem o mesmo significado (CAMPOS, 2018). O processo de Gamificação é utilizar dinâmicas, mecânicas, e outros

elementos de jogos em outros ambientes, atividades e processos com o objetivo de engajar, motivar, emocionar e mudar o comportamento das pessoas envolvidas (VIANNA *et al.*, 2013; ZICHERMANN; CUNNINGHAM, 2011; KIM, 2015). Dois elementos importantíssimos que a técnica de Gameficação utiliza são o sistema de recompensas e o feedback. (ZICHERMANN; CUNNINGHAM, 2011; KIM, 2015). Aplicativos que no decorrer deste trabalho serão apresentados como estado da arte, utilizam aprofundadamente estas características para garantirem a satisfação dos usuários e que estes venham a utilizar o sistema recorrentemente, de forma a ampliar e fixar os conhecimentos adquiridos pelo uso do mesmo.

2.2 FERRAMENTAS DA EDUCAÇÃO A DISTÂNCIA

2.2.1 Ava's

Ambientes virtuais de aprendizagem (AVA) são softwares na internet ou intranet que possibilita a interação de alunos e professores com a possibilidade de recursos virtuais disponibilizados pela plataforma, como fóruns, quizzes, e slides. Podem ser usados para dar suporte a atividades presenciais, dando assim um aumento nas interações em sala de aula, oferecem também suporte a comunicação e troca de informação entre alunos e professores (RIBEIRO; ARAÚJO MENDONÇA; MENDONÇA, 2007). Uma analogia que pode ser feita é que AVA's são como sala de aulas, onde tem a lista de presença, provas e apresentação de conteúdos. Os alunos possuem acesso a recursos que os professores previamente disponibilizaram para eles. É importante lembrar que existem diversos tipos de AVA's no mercado, alguns são gratuitos e de livre acesso como o Moodle[1] e outros pagos como o Cornerstone[2].

2.2.2 Aplicativos educacionais

Atualmente estudantes experimentam ambientes digitais de maneira tátil e pessoal através de uma grande variedade de dispositivos móveis como smartphones e tablets, no qual o uso pode ser convertido para práticas educacionais colaborativas (VÁZQUEZ-CANO, 2014). O uso de aplicativos tem permeado a vida social, profissional e agora também acadêmica. Por isso diversos países têm apostado em aplicativos educacionais no processo de aprendizagem em diferentes métodos e perspectivas (UNESCO, 2013; JOHNSON *et al.*, 2014).

Cochrane e Bateman (2010), e Dyson *et al.* (2009) concordam que os benefícios do uso de aplicativos educacionais passam por portabilidade, flexibilidade, contextualizando tecnologias educacionais que facilitam o aprendizado, promovem colaboração

[1] https://moodle.org/
[2] https://www.cornerstoneondemand.com/

entre alunos, e encorajam tanto aprendizado independente como colaborativo para a vida.

Um estudo feito por Silva Melo e Carvalho (2014) fez um levantamento de aplicativos educacionais encontrados na loja de aplicativos Android distribuídos de maneira livre e gratuita. O estudo passou desde aplicativos de educação básica até o ensino superior. Dos aplicativos de ensino superior destacam-se os seguintes:

Tabela 2 – Aplicativos Educacionais Livres para o Ensino Superior

Aplicativo	Área	Proposta Didática	Licença	Tamanho
Angulo	Física	Medidor de gravidade	GPLv3	27 kiB
Antikythera	Interdisciplinar	Calculadora científica	FreeBSD	534 KiB
CEToolbox	Medicina	Calculadora de parâmetros de eletroforese capilar	Apache 2	47 KiB
DIYgenomics	Biotecnologia	Medidor de saúde e condicionamento físico	BSD	194 KiB
EP Mobile	Medicina	Tabela periódica de elementos	GPLv3	875 KiB
Sage	Matemática	Calculadora Matemática	GPLv3	245 KiB
NFKmol	Bioquímica	Visualizador de moléculas	LGPL	448 KiB
Ohm Cal	Física	Lei de Ohm	AGPL	596 KiB
Type and Speach	Liguística	Aprendizado de idiomas	Apache2	252 KiB

Fonte: Silva Melo e Carvalho (2014).

Para Silva Melo e Carvalho (2014) os aplicativos funcionam de forma a ser um ferramental de apoio a ações desenvolvidas em sala de aula, o que se faz necessário.

2.2.3 AICC, SCORM e Tin Can API

Um ponto essencial para o gerenciamento da educação a distância é o rastreamento das interações com o conteúdo feitas pelos alunos, isto é, saber como estes interagem com os materiais da plataforma de ensino. Os básicos comumente rastreados são: nota, localização até onde o aluno visualizou os conteúdos disponíveis e completude.

Como o rastreamento de informações é um fator recorrente, alguns padrões surgiram para padronizar o salvamento e comunicação destes dados. Os principais são AICC, SCORM e Tin Can. Esse rastreamento ajuda na avaliação e diagnóstico do engajamento e progresso dos estudantes.

AICC

Comitê de treinamento baseado em computadores da indústria de aviação (AICC em inglês), foi um associação que existiu de 1998 a 2014 e que definiu diversas especificações tecnologias educacionais (AICC, 2014). Tais especificações são utilizadas por muitas ferramentas de criação de conteúdos para educação a distância, tais como Adobe captivate, Skillsoft e Articulate. Dentre as especificações duas se destacam: (1) Protocolo HTTP para comunicação do conteúdo com o AVA, possibilitando

rastreamento do progresso do aluno, e (2) a forma que o conteúdo deve ser sudio no AVA (AICC, 2014).

SCORM

É um padrão de comunicação e execução de objetos de aprendizagem em cursos online (RODRIGUES *et al.*, 2009), ele define como recursos devem ser empacotados e os protocolos de comunicação (API) com os principais AVA's. Foi criado pela ADL, iniciativa americana de educação da Secretaria de Defesa para padronização de conteúdos técnicos. Sua primeira versão foi publicada em 2000, fortemente inspirada pelos padrões previamente estabelecidos pelo AICC. É um padrão bastante disponível em AVA's.

Tin Can API

Para suprir lacunas dos dois padrões anteriores, a Tin Can API surgiu por desenvolvimento da Rustici Software (XAPI, 2019) que posteriormente foi renomeada de Experience Api (xAPI). Estendeu funcionalidades como reportar múltiplas avaliações ao invés de uma, melhoramento na segurança, transição de plataformas, e rastreamento de vários tipos de objetos de aprendizagem e recursos (mobile learning, simulações, mundos virtuais, serious games, atividades presenciais, educação experiencial, educação social, educação offline, e educação colaborativa) (BEHRINGER, 2013).

2.2.4 Ferramentas de comunicação

A comunicação na EaD se faz importante tendo em vista que os contatos são mais limitados em comparação com a educação presencial. Esta comunicação se dá pelo uso de tutoriais, orientações, observações de trabalhos, autoavaliações a avaliações finais (NUNES, 1993).

Segundo Maia e Neto (2007) as comunicações na EaD podem se dar de maneiras síncronas e assíncronas. Para Oliveira *et al.* (2017, p. 81): "Na comunicação síncrona são usados aqueles meios de comunicação que ocorrem em tempo real, ou seja, marca-se um horário para que todos participem ao mesmo tempo, sendo eles:

- Videoconferências;
- Chat;
- Audioconferência;
- Teleconferência;

Já na comunicação assíncrona são utilizados aqueles que não dependem do tempo real, podem ser realizadas a qualquer momento, tanto pelos alunos, quanto pelos tutores, sendo:

- Portfólio;
- Blog;
- Fórum;
- E-mail;"

2.3 UI E UX

A interface do usuário (do inglês user interface, comumente abreviada por UI), é parte do sistema que comunica-se diretamente com o usuário, sendo aquela visível a com quem este interagem para dar comandos ao sistema. Para Griffin e Baston (2014, p. 2) “uma interface é o conjunto de meios pelos quais as pessoas (os usuários) interagem com uma determinada máquina, dispositivo, programa de computador ou outra ferramenta complexa (o sistema)”.

Já a experiência do usuário (do inglês user experience, comumente abreviado por UX), termo criado por Norman (1990) é aquela que responsável pelo sentimento do usuário a usar determinado software, segundo Garrett (2011) é quando alguém te questiona como é usar determinado sistema, essa pergunta refere-se a sua UX. Sua principal função é tornar o produto útil, habitual e acessível, que a sua interface seja usável e desejável, e que a informação que veicula seja credível (MORVILLE, 2004). Abarentos (2017) tenta diferenciar os dois conceitos de UI e UX pelos seguintes pontos:

UX:

- É a totalidade do produto ou sistema, como esse é estruturado e organizado.
- É como o usuário navega de A a B, e como ele se sente fazendo isto.
- O processo muitas vezes envolve pesquisa, entrevistas, criação de personas, examinar a usabilidade e acessibilidade.

UI:

- É apenas a interface: O layout, as cores, botões, ilustrações, e etc. Tudo que é visual.
- É a parte que o usuário de fato vê e interage.

Se preocupar com UI/UX ao se desenvolver um software além de ser um diferencial, é extremamente importante para projetar o êxito do produto, que segundo Garrett (2011) se dá conforme quanto de atenção é dado a este assunto.

2.4 DESENVOLVIMENTO WEB

Desenvolvimento web refere-se a aplicações e sistemas desenvolvidos para serem usado pelo uso da internet ou intranet. São rodados sem a necessidade da instalação de um programa a não ser um navegador Web, e podem ser simples páginas estáticas até aplicações completas como redes sociais e comércios eletrônicos. Segundo um artigo retirado do site de tecnologia especializado Techopedia , o processo de desenvolvimento web incluí web design, desenvolvimento de conteúdo web, código cliente/servidor e configuração de rede segura, além de outras tarefas.

2.4.1 World Wide Web

A World Wide Web (comumente abreviada por WWW) é um sistema que disponibiliza páginas web e seus recursos através de URLs como por exemplo http://sin.inf.ufsc.br, que são acessadas por navegadores como Google Chrome, Mozilla Firefox, Microsoft Edge e outros. Segundo Berners-Lee *et al.* (1994, p. 298) "A Web é construída pelo paradigma cliente/servidor. O usuário senta na frente de um cliente em rede (usualmente um terminal gráfico) rodando um Web Browser. O Browser instrui um servidor (usualmente outro computador localizado remotamente) a providencia um documento hipertexto específico. O documento é entregue e visualizado na tela do cliente. Referências nesse documento hipertexto, chamando de links, são representados como porções de texto destacados. O usuário clica no mouse em um link específico, e o documento referenciado, que pode ou não estar no mesmo servidor do documento original, é recuperado e visualizado no cliente".

Um aspecto notório da Web é que ela possui um consócio web, o W3C, que segundo o próprio se auto designa como "O World Wide Web Consortium (W3C) é uma comunidade internacional que desenvolve padrões abertos para garantir o crescimento a longo prazo da Web". Possui cerca de 400 membros que agrega empresas, órgãos governamentais e organizações independentes que tem o objetivo de estabelecer padrões que ajudem o crescimento saudável da web. Além disso possuem diversos cursos e documentos em seu site para auxiliar usuários e desenvolvedores web a utilizarem e construírem conteúdos para a internet.

2.4.2 Hypertext Transfer Protocol

Protocolo de transferência de Hipertexto (Hypertext Transfer Protocol, ou HTTP), é um protocolo de troca de mensagens da camada de aplicação, é a base de troca de

documentos da World Wide Web. A sua primeira versão foi disponibilizada em 1991 pelo seu criador Tim Berners-Lee, mesmo criador do WWW e fundador da W3C. É um protocolo leve e rápido necessários para a distribuição de sistemas da informação com hipermedia colaborativa (BERNERS-LEE, 1990). A versão mais atual desde a publicação deste trabalho é a versão 3.0, disponibilizada em 2018.

No protocolo encontra-se especificações para sintaxe de mensagem e roteamento, semânticas e conteúdo, requests condicionais, requests de intervalo, cacheamento, e finalmente autenticação.

Tabela 3 – Principais métodos do protocolo HTTP

Method	Description
GET	Request for resource from server
POST	Submit data to the server
HEAD	Same as GET but does not return de body
PUT	The data within the request must be stored at the URL supplied, replacing any existing data
DELETE	Delete a resource
OPTIONS	Return de HTTP methods supported by the server
CONNECT	Clients requests the HTTP proxy to forward a TCP connection to some destination. Used to create a TCP/IP tunnel for secure connections using HTTP proxies

Fonte: Chemvura (2017).

2.4.3 Front-end

A parte do sistema com o que a usuário lida diretamente é conhecida por front-end. Para Amaral e Almeida Neris (2015) o front-end é responsável em coletar as informações de entrada do usuário, processá-la e adequá-la de maneira que possa ser repassada para o back-end utilizar. O front-end é uma espécie de interface entre usuário e back-end, onde ambos podem estar distribuídos em um ou mais sistemas (MORRISON, 2008). No front-end web as principais tecnologias utilizadas são HTML, CSS, e Javascript.

HTML

Hypertext Markup Language (linguagem de marcação de hipertexto, ou apenas HTML) e a linguagem que delimita e estrutura páginas e aplicações web. Providência nomes (tags) que descrevem diferentes tipos de conteúdo, como <audio>, , <header>. O navegador web consegue ler estas tags, e renderiza conteúdo visual para o usuário acessando o página.

CSS

Cascading style sheets (folhas de estilo cascateadas, ou apenas CSS) são os arquivos que dão estilo a arquivos HTML. Estes estilos permitem adicionar cor,

modificar tamanhos, adicionar imagens de fundo,e fazer um design responsivo que funcione tanto em mobile como em desktops.

JavaScript

JavaScript é uma linguagem de programação interpretada criada pela Netscape em 1995, inicialmente funcionava apenas no navegador, porém atualmente está presente em diversas aplicações, podendo ser usado de ponta a ponta ao desenvolver um programa. Javascript é usado especialmente para criar aplicações interativas que rodam na internet, permite a interação e renderização de documentos HTML dependendo da entrada do usuário ou de dados advindos de um servidor.

2.4.4 Back-end

O Back-end é onde estão as regras de negócio da aplicação, é parte do sistema que valida os dados vindos do front-end e faz persistência dos mesmos (AMARAL; ALMEIDA NERIS, 2015). Geralmente está alocado em um servidor remoto onde o usuário da aplicação não tem acesso.

NodeJS

"NodeJs é Javascript rodando em tempo de execução criado em cima do motor Javascript V8 do Chrome"[3]. Isso significa que Node possibilita a execução de Javascript fora do navegador (lado do cliente), e roda direto no servidor. Node é open-source e foi originalmente desenvolvido por Ryan Dahl em 2009, quando ele foi criado queria resolver problemas como limitação de múltiplas requisições simultâneas e código síncrono que bloqueia o processo ou implica em múltiplas camadas de execução, gastando mais hardware e software (Garbar, 2017).

Servidor Web

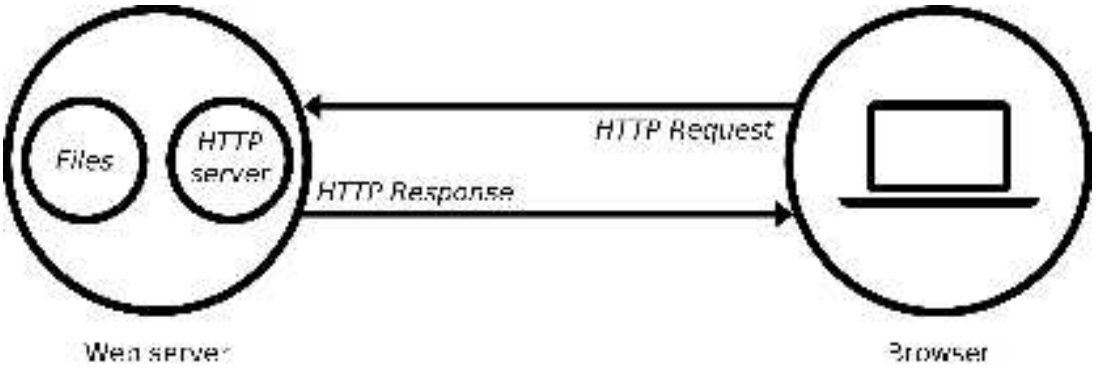
Um servidor que hospeda arquivos e os disponibiliza por requisições HTTP para seus clientes. Segundo a MDN (2019), organização de desenvolvedores da Mozilla, "o servidores web podem ser no hardware, software ou ambos trabalhando junto:

1. Referente ao hardware, um servidor web é um computador que armazena arquivos que compõem os sites (por exemplo, documentos HTML, imagens, folhas de estilo, e arquivos JavaScript) e os entrega para o dispositivo do usuário final. Está conectado a Internet e pode ser acessado através do seu nome de domínio (DNS), como por exemplo mozilla.org.

[3] https://nodejs.org/en/

2. Referente ao software, um servidor web inclui diversos componentes que controlam como os usuários acessam os arquivos hospedados (armazenados para disponibilização), no mínimo um servidor HTTP. Um servidor HTTP é um software que compreende URLs (endereços web) e HTTP (o protocolo que seu navegador utiliza para visualizar páginas web."

Figura 3 – Funcionamento básico de um servidor web



Fonte: MDN (2019)

Banco de dados

Um banco de dados são dados inter-relacionados que estes representam informações de uma determinado sistema ou domínio. “A coleção de dados, normalmente conhecida como banco de dados, contém informações relevantes para uma empresa. O principal objetivo de um SGBD é proporcionar uma forma de armazenar e recuperar informações de um banco de dados de maneira conveniente e eficiente” (KORTH; SILBERSCHATZ; SUDARSHAN, 1994, p. 55).

As principais operações que podem ser solicitadas pelos usuários ao SGBD são:

- Acrescentar novos arquivos no banco de dados.
- Inserir dados em arquivos existentes.
- Buscar dados de arquivos existentes.
- Excluir dados de arquivos existentes.
- Alterar dados de arquivos existentes.
- Remover arquivos do banco de dados.

3 FERRAMENTAS CORRELATAS

Neste capítulo encontra-se uma pesquisa e revisão do que se há de mais novo e popular sobre sistemas educacionais que se propõe avaliar alunos, sendo assim um ferramental de apoio a educação em geral.

Segundo Vázquez-Cano (2014) os resultados de estudos (ALLY, 2009; CHEN; DENOYELLES, 2013; DAHLSTROM, 2012) que exploraram os padrões de uso de aplicativos educacionais ilustram que conteúdos otimizados para dispositivos móveis são amplamente utilizados e os estudantes têm desejo cada vez maior de tais recursos disponibilizados neste formato, incluindo informações administrativas de universidades.

Para cada ferramenta correlata se definiu alguns critérios para analise que visam servir de base de comparação com a ferramenta desenvolvida, sendo eles: Atividades com feedback em tempo real, relatórios para professores, versão para celulares e código aberto.

3.1 CODEACADEMY

Codeacademy é uma plataforma de ensino de programação online, que disponibiliza cursos de linguagens como Java, Phyton, SQL e várias outras. Foi fundada em 2011 por Zach Sims e Ryan Bubinski, e no ano de 2019 conta com 90 funcionários que gerenciam e desenvolvem produtos para 25 milhões de usuários pelo mundo todo. No site da plataforma é possível escolher entre cursos livres ou trilhas de aprendizagem de uma determinada área, como desenvolvimento web, desenvolvimento de aplicativos, banco de dados e muitos outros. Todos os cursos tem um design atrativo e sistema de feedback instantâneo, além de ter provas de proficiência e tutores disponíveis para tirar dúvidas personalizadas.

Figura 4 – Um exemplo de curso no Codeacademy

Fonte: Site oficial do Codeacademy. Disponível em: www.codecademy.com Acesso em: 13 maio 2019

A tela principal do Codeacademy é como na figura acima, do lado esquerdo algum texto de contextualização, ao centro a parte onde o aluno insere linhas de código para visualizar o resultado compilado no lado direito.

A aplicação visa ensinar programação por meio de atividades práticas numa linha de dificuldade crescente, onde o usuário avança as fazes assim que obtém a resposta correta. Além disso monta trilhas de aprendizagem para aprender uma competência completa como programação web por exemplo, disponibilizando em sequencia que se relacionam.

Tabela 4 – Critérios atendidos Codeacademy.

Feedback instantaneo	Relatórios	Versão Mobile	Código aberto
Sim	Não	Sim	Não

Fonte: Criação própria.

3.2 KAHOOT

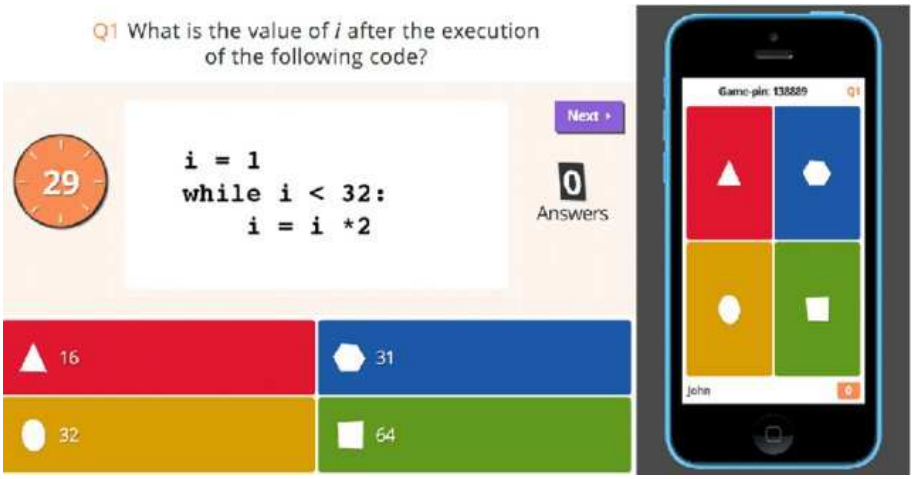
Segundo a página inicial do site[1], “Kahoot! torna fácil criar, compartilhar e se divertir com jogos educativos ou quizzes de trivia em minutos”. O jogo educacional teve sua primeira versão lançada em setembro de 2013 pelos colegas da universidade de tecnologia e ciência da noruega Johan Brand, Jamie Brooker and Morten Versvik.

O aplicativo possui versão desktop e mobile, e em seu modo principal funciona como um questionário interativo, que segundo artigo (SOUZA; NEIVA, 2018) publicado no IV seminário de extensão e pesquisa da CES/JF:

[1] https://kahoot.com/

> O Kahoot é uma plataforma de aprendizagem e ensino online, que busca trazer elementos de gamificação para criação de quizzes que podem ser utilizados em ambientes empresariais, salas de aula e ambientes sociais. Existe uma série de quizzes prontos e compartilhados sobre diversos assuntos na plataforma, além de haver a opção de criar o seu próprio quiz, privado ou público, e com ranking e pontuação ou não, em 4 modos de jogo. O professor pode optar para realizar o quiz em modo competitivo ou não durante as atividades ministradas e pode utilizar o Kahoot como ferramenta de auxílio na realização de atividades em sala, fixação de conteúdo e elaboração de lista de exercícios para as provas. (SOUZA; NEIVA, 2018, p. 713)

Figura 5 – Exemplo de quizz feito no Kahoot!



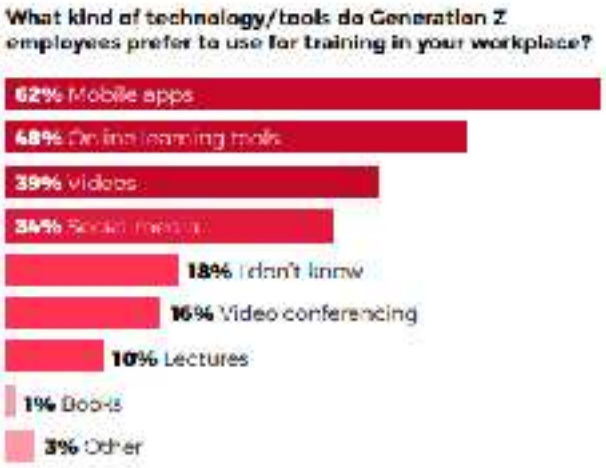
Fonte: Fotaris *et al.* (2016)

No seu site oficial e no relatório desenvolvido pela própria empresa EdTrends em 2018, aparecem alguns dados interessantes, como os listados abaixo:

- 2 bilhões de jogadores desde o lançamento.
- 97% das empresas listadas no Fortune[2] 500 jogam Kahoot.
- 5 milhões de professores usaram nos últimos 12 meses.
- Presente em mais de 200 países.
- Obteve 60 milhões de dólares em investimento até o momento.
- Metade dos estudantes e alunos interagiram com Kahoot nos últimos 12 meses.

No relatório também foi feito um questionário com 392 usuários do Kahoot, e uma das perguntas foi "Que tipos de tecnologia/ferramentas a geração Z[3] de empregados prefere usar para treinamento no local de trabalho". O resultado foi que 62% dos respondentes relataram a preferência do uso de aplicativos móveis.

Figura 6 – Ferramentas preferidas da geração Z para treinamento corporativo



Fonte: Corporate Trainers (2018)

[2] Lista anual das maiores corporações americanas
[3] Pessoas nascidas, em média, entre meados dos anos 1990 até o início do ano 2010

A ferramenta utiliza perguntas curtas e com tempo de resposta limitado para estimular o pensamento rápido para a prática de conhecimentos idealmente já conhecidos pelo usuário. Além de utilizar da gamificação e competição para gerar motivação ao responder os questionários.

Tabela 5 – Critérios atendidos Kahoot!

Feedback instantaneo	Relatórios	Versão Mobile	Código aberto
Sim	Sim	Sim	Não

Fonte: Criação própria.

3.3 DUOLINGO

Um dos aplicativos mais populares de aprendizagem de idiomas é o Duolingo, aplicativo lançado em novembro de 2011 nos Estados Unidos da América pelo professor universitário Luis von Ahn e seu aluno de graduação Severin Hacker. A inspiração para o desenvolvimento do Duolingo veio quando von Ahn que é originário da Guatemala, percebeu que os cursos de linguística eram muito caros para a comunidade. Numa entrevista que deu para a Forbes em 2014, von Ahn e Hacker atestaram que “Educação gratuita realmente irá mudar o mundo”.

Figura 7 – Tela principal do Duolingo

Fonte: Site Oficial do Duolingo. Disponível em: https://pt.duolingo.com/ Acesso em: 21 ago. 2019

O aplicativo se utiliza de técnicas de gamificação ao máximo para engajar os usuários, além de ter um sistema de recompensas onde os alunos ganham os chamados "lingots" que são trocados por customizações que podem ser feitas no avatar do usuário. Possui também um sistema de rankeamento onde os alunos podem ver a onde ficam postados em relação aos seus amigos ou mesmo com o resto do mundo.

No próprio site[4] do sistema é listado uma série de características sobre o mesmo:

- Aprendizagem personalizada: As aulas e exercícios são customizados conforme você utiliza o aplicativo, de modo a aprender e revisar o vocabulário de maneira eficaz
- Feedback imediato: O sistema de avaliação do Duolingo é instantâneo, o aluno sabe na hora se acertou uma questão, e se errou, o aplicativo mostra onde

[4] https://pt.duolingo.com/

melhorar

- Sistema de recompensas: O aluno ganha moedas virtuais (lingotes), desbloquear novos níveis conforme aumenta seu nível.
- Aprendizagem rápida: 34 horas de Duolingo equivalem a um semestre numa escola universitária de línguas.

O aplicativo recentemente lançou o “Duolingo para escolas”, onde professores podem utilizar o sistema em sala de aula para rastrearem o progresso do aluno. Segundo o site “Muitos professores e até mesmo governos inteiros em todo o mundo já veem o Duolingo como o companheiro de aprendizado combinado perfeito para suas salas de aula. As aulas do Duolingo dão a cada aluno feedback e prática personalizados, preparando-os para tirar o máximo proveito da instrução em sala de aula.”.

Um fato interessante, é que o Duolingo disponibiliza dados abertos de mais de 300 milhões de usuários que utilizam a plataforma. A empresa utiliza esse site para fazer estudos de comportamento e inteligência artificial para criar novos produtos e métodos inovadores de aprendizagem.

Na data de publicação deste presente trabalho o Duolingo conta com 32 cursos de diferentes idiomas, cerca de 300 milhões de usuários ativos e ganhou 4 grandes prêmios, sendo um dos mais notórios o prêmio de “melhor aplicativo do ano” da Apple no ano de 2013, primeira vez que um aplicativo educacional esteve no pódio na competição.

Tabela 6 – Critérios atendidos Duolingo.

Feedback instantaneo	Relatórios	Versão Mobile	Código aberto
Sim	Não	Sim	Não

Fonte: Criação própria.

4 PROPOSTA E DESENVOLVIMENTO DO TRABALHO

Com base nas referências levantadas e sistemas analisados nos capítulos anteriores, o presente trabalho implementou uma ferramenta web de auxílio a avaliação de alunos de cursos de graduação, para ser acessado nos mais modernos navegadores de internet.

O sistema é, no seu cerne, um criador e disponibilizador de atividades, onde existem dois principais papéis, o de professor e o de aluno. Os professores poderão criar questões de única ou múltipla escolha a serem respondidas pelos alunos. Os alunos responderão questões interativas e terão o feedback se responderam corretamente em tempo real. Os professores terão um relatório geral e individual dos acertos e erros dos alunos em cada atividade.

Para engajar e motivar a utilização o desenvolvimento focou na simplicidade de uso, portabilidade, e interface amigável de modo a deixar o design da aplicação o mais atrativo possível e tornar o seu uso fácil e rápido, bastante inspirado nos aplicativos levantados no capítulo do estado da arte, de maneira onde o número reduzido de funções deixe o foco no objetivo principal que é a fixação dos conteúdos vistos em sala de aula.

4.1 PROPOSTA DE FORMAS DE USO DA FERRAMENTA

Um professor organiza suas aulas, criando vários planos de aula. Cada plano pode conter, segundo critérios do professor, diversos elementos, tais como:

- Identificação/Tema
- Objetivos/Finalidade/Duração
- Conteúdo
- Estratégia/procedimentos didáticos
- Recursos didáticos
- Métodos de avaliação

Como o foco deste trabalho é de reforçar o engajamento e permitir uma avaliação do aprendizado do estudante, mas também da própria atividade (do plano de aula e de seus elementos: estratégias e recursos adotados por exemplo), recomenda-se que o professor utilize essa ferramenta (Apprenda) como apoio à avaliação, nas seguintes situações:

a) tanto para fazer uma observação do conhecimento/habilidade prévia dos alunos.

b) quanto para avaliar o quanto a atividade didática projetada melhorou (potencialmente) o conhecimento do aluno sobre o tema.

c) e ainda, permitir avaliar (pelo conjunto de resultados), se o plano de aula atingiu os objetivos propostos (se houve uma melhora generalizada do conhecimento sobre o tema, pode-se avaliar que o plano de aula foi adequado, caso contrário, o professor deverá procurar adotar outras práticas/estratégias em atividades futuras).

4.1.1 Preparação da avaliação

Como discutido acima, há várias formas de utilizar a ferramenta Apprenda, dependendo de qual tipo de avaliação o professor deseja realizar:

a) para avaliar o conhecimento prévio (e com isso, organizar a atividade didática a partir do conhecimento prévio demonstrado pelos alunos): Para elaboração da avaliação do conhecimento prévio, o professor deve levantar os pré-requisitos daquele plano de aula e selecionar perguntas simples que representariam os pré-requisitos daquela atividade. As respostas dos estudantes permitem identificar a necessidade de reforços ou de redirecionamento da atividade proposta para revisar o conhecimento prévio necessário.

b) para avaliar se os objetivos da atividade didática foram atingidos (com o reforço do novo conhecimento construído ao longo da atividade):

1. O professor deve identificar, dentro do conteúdo e objetivos da atividade didática, os elementos-chave que gostaria de reforçar com os estudantes, ao final da atividade.

2. Para isso, o professor elabora perguntas simples para receber feed-back dos estudantes. Essas perguntas permitem que os próprios alunos identifiquem quais são os elementos-chave da atividade e se compreenderam os elementos discutidos, ainda no contexto da atividade.

3. Os relatórios a respeito das questões apresentadas ajudam ao professor a identificar pontos a reforçar, o que pode ser realizado ainda naquele momento ou em um futuro encontro.

c) para avaliar o próprio plano de aula, ou seja, avaliar se a própria atividade foi do gosto dos estudantes: O professor pode utilizar também a ferramenta para avaliar se as estratégias e recursos adotados foram adequados, na perspectiva dos estudantes. Para tal, ele deve elaborar questões simples (sem feedback) que o ajudem a paulatinamente ajustar as perspectivas dos estudantes às metas de aprendizagem definidas e os recursos disponíveis.

Figura 8 – Protótipo da tela da visualização de perguntas do aluno.

Fonte: Criação própria

Figura 9 – Protótipo da tela de feedback para o aluno.

Fonte: Criação própria.

4.2 A FERRAMENTA

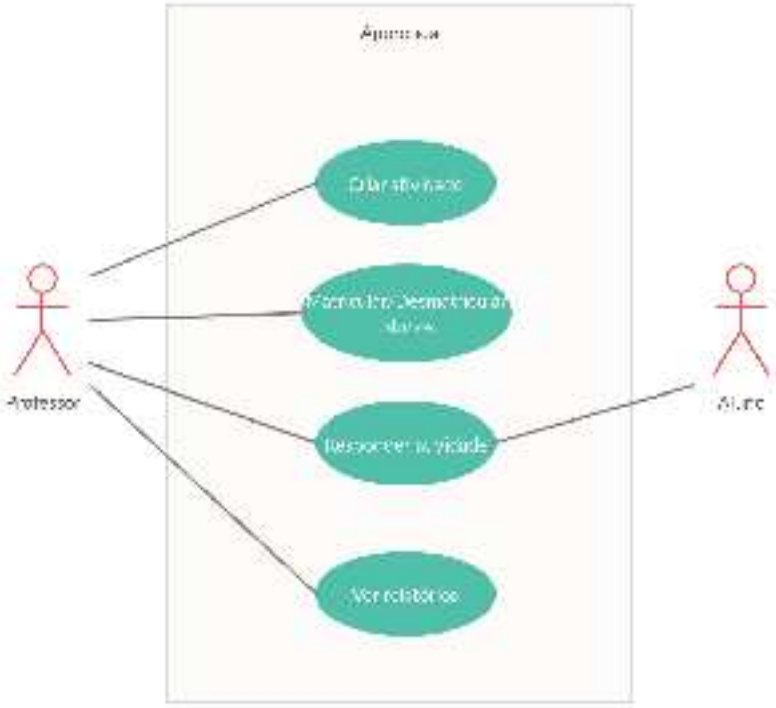
Tendo em vista os conceitos definidos acima, partiu-se para a construção de uma ferramenta como prova de conceito em que alguns elementos poderiam ser usados e testados, sendo inicialmente sendo nomeada de Apprenda.

A aplicação final foi construída imaginando o seu uso por dois tipos de usuários: Aluno e professor. O professor tem como ações principais a criação de atividades, disponibilização dessas atividades aos alunos e análise dos resultados obtidos pelos alunos.

O aluno tem uma atividade apenas, que é a de responder as atividades que foram designadas para ele. Podendo rever suas respostas nas atividades a qualquer momento, contanto que o professor as deixe disponíveis.

O professor é um papel que estende o perfil de aluno, portanto, ele assume as funções de responder as atividades que ele mesmo pode se matricular.

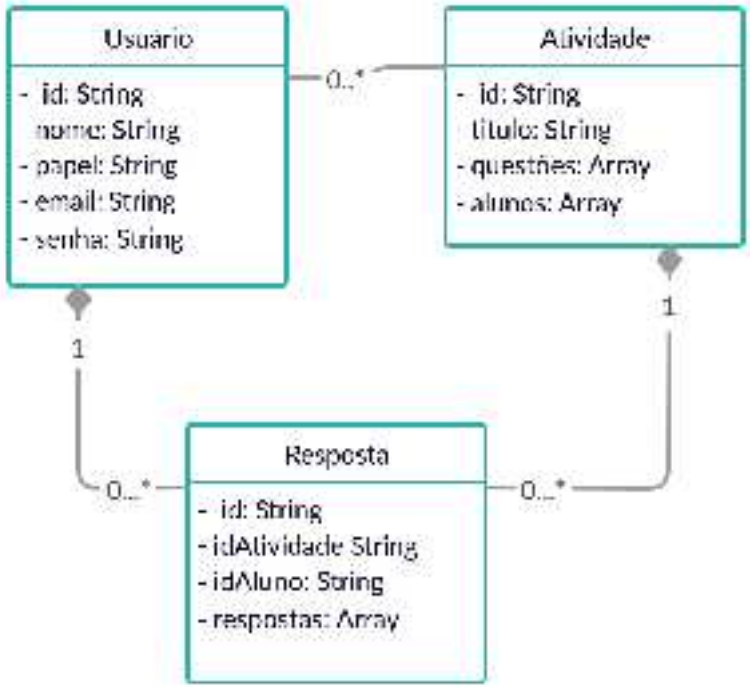
Figura 10 – Diagrama de casos de uso da aplicação



Fonte: Criação própria

As entidades da aplicação que estão armazenadas no banco de dados são apenas três: Usuário, Atividade e Respostas. Os atributos de cada entidade estão descritos no diagrama de classe abaixo.

Figura 11 – Diagrama de classes da aplicação



Fonte: Criação própria.

4.2.1 Restrições e requisitos

Por se tratar de uma aplicação web, um requisito básico para seu uso é a utilização de um navegador. Existem muitos navegadores no mercado, porém alguns são mais difundidos e estão em constatante atualizações, suportando assim as mais novas tecnologias e API's que as aplicações podem vir utilizar, como por exemplo animações robustas, requisições HTTP e suporte a funções assíncronas.

Os navegadores que atendem estes demandas e são os indicados para utilizar a ferramentas são o Google Chrome, Mozilla Firefox e o Microsoft Edge. Todos em suas versões mais atualizadas. Não foi dado suporte e nem analisado o comportamento da ferramenta em outros navegadores ou de versões antigas.

4.2.2 Tecnologias utilizadas e arquitetura

A escolha da aplicação ter sido desenvolvida para um ambiente web se deu por alguns pontos, entre eles a facilidade de uso tendo em vista que não é necessário a instalação de nenhum programa além de um navegador para utilizar a ferramenta, alta demanda e abrangência de aplicações web, portabilidade otimizada por poder rodar em qualquer ambiente que possua um navegador web, como celulares, tablets, e televisores.

A linguagem de programação que permeia toda a aplicação é o JavaScript, o que possibilita transitar entre o lado do servidor e cliente de maneira simples. No servidor é utilizado o NodeJS um interpretador de JavaScript assíncrono que permite leitura e escrita de arquivos, acesso a banco de dados, criação de requisições HTTP e muitas outras características. Junto com o NodeJS foi utilizado o Express, um framework que facilita a criação de servidores web. O servidor disponibiliza uma API para receber chamadas do cliente e executar alguma ação no banco de dados e retornar um resposta quando finalizada a operação.

O banco de dados utilizado foi o MongoDB, um banco não relacional que utiliza notação JSON[1] para armazenar os documentos, possibilitando a utilização de funções comuns a linguagem para fazer consultas e escritas no banco. Ele é flexível de maneira que é recomendado para sistemas que não tem muito bem definido esquemas e entidades que vão ser utilizadas no futuro, facilitando a adição de atributos e coleções a medida que se faz necessário.

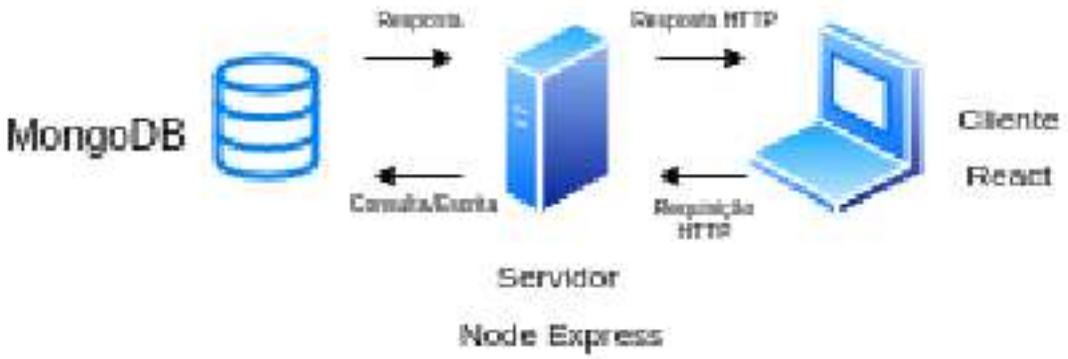
Na parte do cliente foi utilizado o framework React, desenvolvido pelo Facebook. O React simplifica a criação de interfaces de usuários por utilizar o conceito de componentização que incentiva a criação de componentes genéricos que possam ser reutilizados pela aplicação. Uma característica também notável, é que ele otimiza o processo de renderização(compilação) de páginas HTML, por técnica que atualizam

[1] Javascript Object Notation ou notação de objeto Javascript

apenas a menor porção necessária, otimizando muito a memória gasta, e agilizando o carregamento das páginas. As páginas são criadas com HTML e estilizadas com CSS, e as chamadas e funções interativas são feitas com Javascript.

A seguir um diagrama que elucida melhor a arquitetura da aplicação.

Figura 12 – Arquitetura geral da aplicação



Fonte: Criação própria.

4.3 UTILIZANDO A FERRAMENTA APPRENDA

A primeira página que o usuário vê é a de login, nela o usuário cadastrado entra com o e-mail e senha para ser autentificado no sistema e assim poder utilizar a ferramenta. É possível ter na mesma instância (cliente) quantos professores e alunos desejar, isso se viu necessário para quando há casos de disciplinas que possuem monitores ou até mesmo são ministradas por mais de um professor, dessa forma os professores e monitores compartilham e veem os relatórios de todos os alunos, além de poderem criar atividades a serem respondidas de maneira paralela, otimizando o processo de avaliação.

Figura 13 – Tela de login

Fonte: Criação própria

4.3.1 Perfil professor

Lista de atividades

Como professor, aparece uma página de atividades e uma de relatórios. Na página de atividades ficam aquelas que foram previamente criadas, onde ao clicar em uma, vai para a parte de edição da mesma. Há também um botão para criação de novas atividades.

Figura 14 – Tela de lista de atividades professor

Fonte: Criação própria.

Criação e edição de atividades

Ao criar ou editar uma atividade, o usuário adiciona questões de múltipla ou única escolha, e então cria as opções de resposta para a questão escolhendo aquela

ou aquelas que são as corretas.

Figura 15 – Criação e edição de atividades

Fonte: Criação própria.

Foram modelados alguns outros tipos de atividades e modelos de feed-back que acabaram não saindo no protótipo inicial como: atividades com mais de uma chance, verdadeiro ou falso, ligar os pontos, respostas dissertativas, feed-back só ao final de toda atividade, sem feed-back, feed-back específico para cada questão.

Matrícula de alunos

Ao selecionar uma atividade tem se a lista de usuários matriculados e aqueles que podem ser matriculados para começar a responder.

Figura 16 – Gerenciamento de alunos de uma atividade

Fonte: Criação própria

Relatórios

A página de relatórios é separada por relatórios gerais onde se tem informações sobre número de atividades cadastradas, números de usuários por perfil, e total de respostas. Além disso, nesta página exite uma lista de usuários que ao selecionar um, redireciona para as informações detalhadas daquele aluno e as respostas dadas em cada atividade. Por fim uma lista de atividades que, selecionando qualquer uma, leva a detalhes da performance dos alunos.

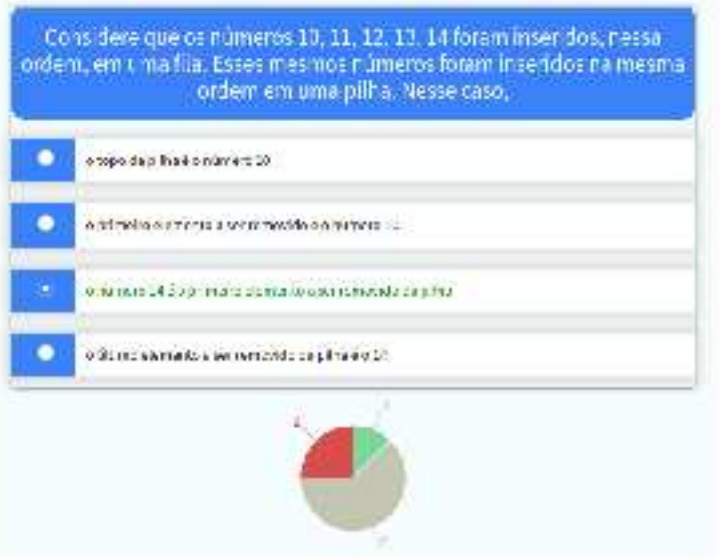
Figura 17 Relatório geral de todas atividades e usuários. Usuários separados por professores(azul) e alunos (verde)

Fonte: Criação própria.

Figura 18 Relatório de uma atividade, alunos matriculados (azul) e que responderam a atividade (verde)

Fonte: Criação própria.

Figura 19 Relatório de uma questão da atividade. Alunos corretos (verde), alunos incorretos (vermelho) e alunos que não responderam (cinza)



Fonte: Criação própria

Figura 20 – Relatório de um aluno especifico nas atividades

Fonte: Criação própria.

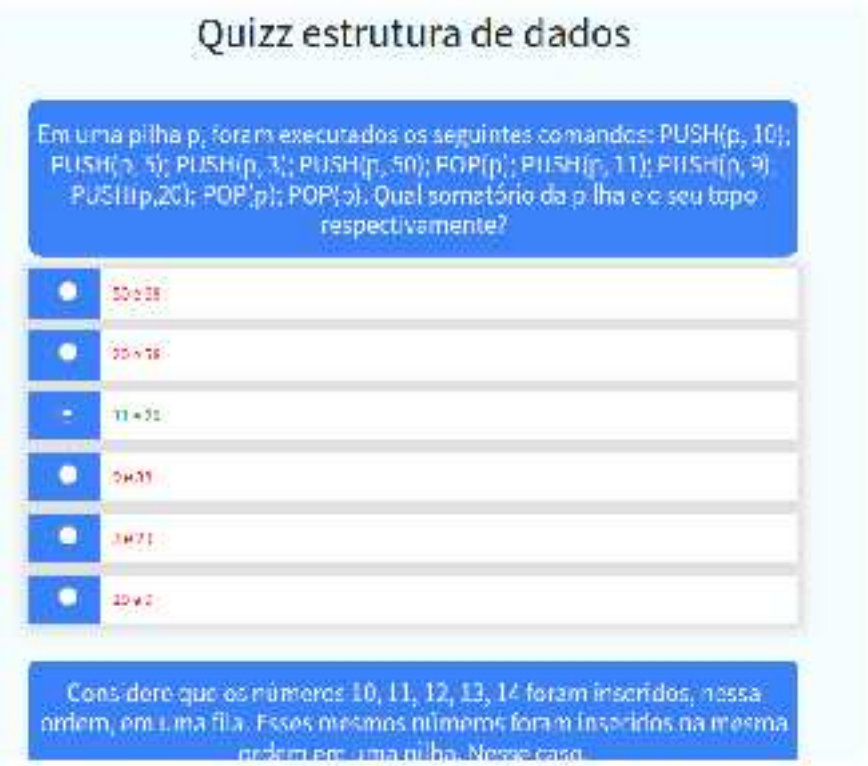
4.3.2 Perfil aluno

O perfil de aluno é o mais simples de propósito, porque o único foco dele é responder as atividades e receber o feedback de acerto ou erro. Na tela principal tem uma lista de atividades que lhe foram atribuidas, e ao selecionar uma ele pode respondê-la. O aluno ainda pode utilizar as respostas dadas como gabarito, de maneira que as respostas ficam salvas para eventuais consultas. Essa é uma maneira de relembrar os conceitos vistos em aula, e gravar na memória por meio da repetição.

Figura 21 – Lista de atividades disponibilizas para o aluno

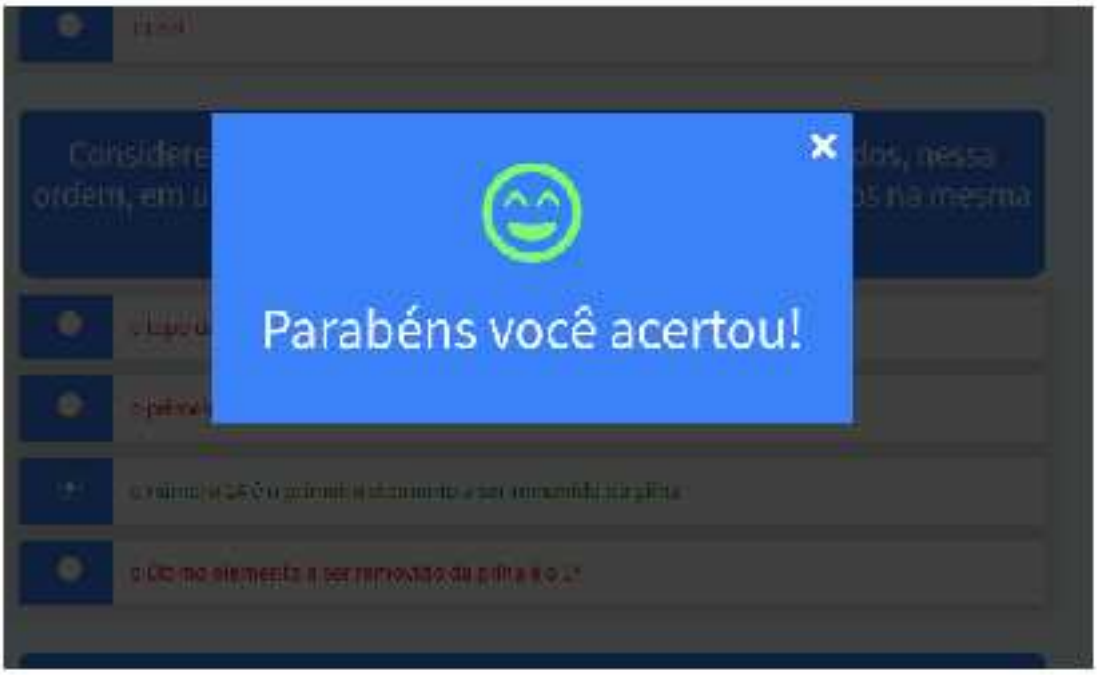
Fonte: Criação própria

Figura 22 – Visualização de uma atividade



Fonte: Criação própria.

Figura 23 Feedback da resposta do aluno



Fonte: Criação própria.

5 CONSIDERAÇÕES FINAIS

Aprendizado não se dá em um evento único. Quando aprendemos um conceito novo, é necessário exercitá-lo para que fique na memória por mais tempo. Neste trabalho foi desenvolvido uma ferramenta que ajuda a reter conhecimento e exercitar conceitos vistos em sala de aula por meio de atividades de única ou múltipla escolha. Além disso a ferramenta consegue compilar dados de respostas dos alunos para que sejam analisados e sirvam de base para tomadas de decisões, como por exemplo, uma sala em que a maioria dos respondentes errou a mesma questão, se faz necessária uma revisão do conteúdo para sanar dúvidas do mesmo. Também se faz possível uma ação individual, uma vez que se tem dados separados por aluno.

A avaliação é uma importante ferramenta para se dar direcionamento aos estudos e também identificar se os conhecimentos foram bem compreendidos pelos receptores. "A avaliação diagnóstica é uma ferramenta fundamental para que as instituições de ensino avaliem o processo de aprendizagem. Por isso, é preciso ser ágil, inovar e buscar soluções que auxiliam gestores, coordenadores e professores a criar a melhor experiência para os alunos"(NICOLETTI, 2019, p. 1).

Foram estudadas ferramentas que são amplamente utilizadas e tem um viés educacional, e delas foi buscado inspiração nas características que as tornam interativas e divertidas de utilizar. Com isto foi desenvolvido a prova de conceito com elementos de aprendizado gamificado, microlearning, totalmente web e com suporte a relatórios para professores e tutores.

Por fim a ferramenta se concebeu na vontade de auxiliar o aprendizado, dando uma opção simples e eficaz de experienciar o conhecimento adquirido previamente por meio das aulas regulares em sala, como é feito pelo bleended learning. Qualquer momento que o aluno tenha disponível em seu dia e/ou que tenha mais vontade, pode utilizar o sistema para relembrar e testar o que aprendeu, tanto em um computador pessoal como no seu smartphone, respondendo questões objetivas criadas pelos professores.

5.1 DIFERENCIAIS EM RELAÇÃO A OUTROS APLICATIVOS

Apesar de ser mais um entre tantos aplicativos com a mesma proposta, o sistema se diferencia principalmente daqueles elencados no estado da arte por alguns pontos chaves.

- É uma ferramenta totalmente gratuita e aberta, onde seu código pode ser reutilizado, aprimorado e alterado por qualquer um que assim deseje. Pode ser inclusive utilizado para o estudo de programação por meio da análise do código desenvolvido.

- Totalmente web, possibilitando o seu uso em qualquer dispositivo que possua um navegador de internet, contando com responsividade de design, onde a aplicação se molda do melhor jeito ao tamanho da tela.
- Não é preciso muitas configurações para começar a usar a ferramenta. A principio só prover questionários e fazer o gerenciamento de alunos que vão estar atrelados a cada um.
- Analise de dados tanto gerais como específicos por questionário ou aluno. Disponibilizando os dados numa interface amigável e simples.

A seguir a comparação das tabelas feitas para cada ferramenta do capitulo 3 com a ferramenta desenvolvida.

Tabela 7 – Comparação das ferramentas correlacionadas com a desenvolvida.

Ferramente	Feedback instantaneo	Relatórios	Versão Mobile	Código aberto
Codeacademy	Sim	Não	Sim	Não
Kahoot!	Sim	Sim	Sim	Não
Duolingo	Sim	Não	Sim	Não
Apprenda	Sim	Sim	Sim	Sim

Fonte: Criação própria.

5.2 TRABALHOS FUTUROS

Se desenvolveu uma ferramenta de avaliação simples e ágil mas que pode servir de base para desenvolvimentos mais robustos no futuro. Algumas funcionalidades que são sugeridas para possíveis implementações são:

- Mais opções de de avaliação como verdade e falso e todos os outros tipos de atividades modeladas.
- Ter um feed-back específico para cada questão, de maneira a clarificar o porque da resposta certa.
- Relatórios mais robustos, podendo conter até algoritmos avançados de mineração de dado.
- Disponibilização da ferramenta em algum servidor na internet para compartilhamento.
- Fazer uma pesquisa de usabilidade do aplicativo para obter pontos bons e de melhoria possíveis.

REFERÊNCIAS

ABARENTOS, Lea Sacdalan. **UX vs UI**. [*S.l.*: *s.n.*], 2017. Disponível em: https://slides.com/leasacdalanabarentos/ux-vs-ui/fullscreen#/0/19. Acesso em: 12 out. 2019.

ALLY, Mohamed. **Mobile Learning**: Transforming the Delivery of Education and Training. Canada: AU Press, 2009.

ALMEIDA, Leandro. **A avaliação da aprendizagem escolar e a função social da escola**. São Paulo: Programa de Pós-Graduação em História e Filosofia da Educação. Pontifícia Universidade Católica de São Paulo, 2001.

ALMEIDA, Maria Elizabeth Bianconcini de. **Educação a distância na internet**: abordagens e contribuições dos ambientes digitais de aprendizagem. São Paulo: Educação e Pesquisa, 2003. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1517-97022003000200010. Acesso em: 3 set. 2019.

AMARAL, Rodrigo Augusto do; ALMEIDA NERIS, Vânia Paula de. **Análise comparativa entre frameworks de frontend para aplicações web ricas visando reaproveitamento do back-end**. São Carlos: Revista T.I.S, 2015. Disponível em: http://www.revistatis.dc.ufscar.br/index.php/revista/article/view/303. Acesso em: 17 set. 2019.

BECKER, Samantha Adams *et al.* **NMC Horizon Report: 2017 Higher Education Edition**. Austin, Texas: The New Media Consortium, 2017. Disponível em: https://www.nmc.org/publication/nmc-horizon-report-2017-higher-education-edition. Acesso em: 21 set. 2019.

BEHRINGER, Reinhold. **Interoperability Standards for MicroLearning**. Austria: International MicroLearning Conference, 2013. Disponível em: https://www.researchgate.net/publication/258449941_Interoperability_Standards_for_MicroLearning. Acesso em: 11 out. 2019.

BENAKOUCHE, Tâmara. **Educação à Distância (EaD): Uma Solução ou um Problema?** Petrópolis, RJ: Centro de Investigação em Sociologia Económica e das Organizações. Instituto Superior de Economia e Gestão. Universidade Técnica de Lisboa, 2000.

BERNERS-LEE, Tim. **Information Management: A Proposal**. [*S.l.*]: W3C, 1990. Disponível em: https://www.w3.org/History/1989/proposal.html. Acesso em: 17 set. 2019.

BERNERS-LEE, Tim *et al.* **Computers in Physics**. 8. ed. [*S.l.*]: AIP Publishing, 1994.

BRUNO, Gilberto Marques. **Considerações sobre a questão do ensino à distância no Brasil e a necessidade de uma legislação voltada ao e-learning diante do crescimento do w.w.w**. [*S.l.*: *s.n.*], 2001. Disponível em: http://www.buscalegis.ufsc.br/revistas/index.php/buscalegis/article/view/3849/3420. Acesso em: 7 set. 2019.

BUCHEM, Ilona; HAMELMANN, Henrike. **Microlearning: a strategy for ongoing professional development**. [*S.l.*]: Creative Education, 2010. Disponível em: https://www.scirp.org/(S(czeh2tfqyw2orz553k1w0r45))/reference/ReferencesPapers.aspx?ReferenceID=1644222. Acesso em: 30 set. 2019.

CAMPOS, Thiago de. **Uma plataforma para integração e validação de regras e elementos de gamificação em sistemas**. Florianópolis: Repositório Institucional da UFSC, 2018. Disponível em: https://repositorio.ufsc.br/handle/123456789/187870. Acesso em: 9 out. 2019.

CHEMVURA, Tinotenda. **LARMAS - Language Resource Management System**. Cape Town: [*s.n.*], 2017. Disponível em: https://www.researchgate.net/publication/323884666_LARMAS_-_Language_Resource_Management_System/figures?lo=1. Acesso em: 17 set. 2019.

CHEN, Baiyun; DENOYELLES, Aimee. **Exploring Students' Mobile Learning Practices in Higher Education**. San Jose,California: EDUCAUSE Review, 2013. Disponível em: https://er.educause.edu/articles/2013/10/exploring-students-mobile-learning-practices-in-higher-education. Acesso em: 6 dez. 2019.

CISCO. **Mobile Forecast Projects 70 Percent of Global Population Will Be Mobile Users**. [*S.l.*: *s.n.*], 2016. Disponível em: https://newsroom.cisco.com/press-release-content?articleId=1741352. Acesso em: 5 out. 2019.

COCHRANE, Thomas; BATEMAN, Roger. **Smartphones give you wings**: Pedagogical affordances of mobile Web 2.0. Melbourne, Australia: Australasian Journal of Educational Technology, 2010. Disponível em: https://ajet.org.au/index.php/AJET/article/view/1098. Acesso em: 22 out. 2019.

CORPORATE TRAINERS, Kahoot! EdTrends Report for. **Gen Z in the workplace**: How to train and inspire them. [*S.l.*]: Kahoot, 2018. Disponível em: https://kahoot.com/files/2018/12/Kahoot-Edtrends-2018-Corporate-Training-report.pdf. Acesso em: 6 dez. 2019.

CRISTIANA, Paula. **Ensino privado e EaD são incapazes de suprir vácuo de universidades públicas**. São Paulo: Diário Comércio Indústria e Serviços, 2019. Disponível em: https://www.dci.com.br/impresso/ensino-privado-e-ead-s-o-incapazes-de-suprir-vacuo-de-universidades-publicas-1.802479. Acesso em: 6 dez. 2019.

DAHLSTROM, Eden. **Student Mobile Computing Practices, 2012**: Lessons Learned from Qatar. [*S.l.*]: EDUCAUSE Review, 2012. Disponível em: https://library.educause.edu/resources/2012/5/student-mobile-computing-practices-2012-lessons-learned-from-qatar. Acesso em: 6 dez. 2019.

DYSON, Laurel Evelyn *et al.* **Advancing the m-learning research agenda for active, experiential learning**: Four case studies. Melbourne, Australia: Australasian Journal of Educational Technology, 2009. Disponível em: https://ajet.org.au/index.php/AJET/article/view/1153. Acesso em: 22 out. 2019.

FOTARIS, Panagiotis *et al.* **Climbing Up the Leaderboard: An Empirical Study of Applying Gamification Techniques to a Computer Programming Class**. [*S.l.*]: Electronic Journal of e-Learning, 2016. Disponível em: https://www.researchgate.net/publication/293816223_Climbing_Up_the_Leaderboard_An_Empirical_Study_of_Applying_Gamification_Techniques_to_a_Computer_Programming_Class. Acesso em: 24 set. 2019.

GARRETT, Jesse James. **The elements of user experience**. [*S.l.*]: AIGA New Riders, 2011. Disponível em: http://www.jjg.net/elements/pdf/elements_ch02.pdf. Acesso em: 12 out. 2019.

GOMES, Maria João. **E-LEARNING: REFLEXÕES EM TORNO DO CONCEITO**. [*S.l.*: *s.n.*], 2005. Disponível em: https://repositorium.sdum.uminho.pt/bitstream/1822/2896/1/06MariaGomes.pdf. Acesso em: 16 ago. 2019.

GONÇALVES, C.T.F. **Quem tem medo do ensino a distância?** Rio de Janeiro: INED/IBASE, 1997. Disponível em: http:/www.%20intelecto.net/ead/consuelo.html.

GRIFFIN, Ben; BASTON, Laurel. **Interfaces**. [*S.l.*: *s.n.*], 2014. Disponível em: https://www.wikizeroo.org/index.php?q=aHR0cDovL3BlYWNlLnNhdW1hZy5lZHUvZmFjdWx0eS9rYXJkYXMvQ291cnNlcy9DUy9JbnRlcmZhY2VzMjAwN19maWxlcy9JbnRlcmZhY2VzMjAwNy5wcHQ. Acesso em: 11 out. 2019.

HAN, Insook; SHIN, Won Sug. **The use of a mobile learning management system and academic achievement of online students**. UK: Journal Computers & Education, 2016. Disponível em: https://dl.acm.org/citation.cfm?id=3030754. Acesso em: 6 out. 2019.

HAYDT, Regina Celia Cazaux. **Avaliação do processo ensino-aprendizagem**. São Paulo: Ática, 2002. Disponível em: https://rcolacique.files.wordpress.com/2010/08/didc3a1tica-texto-13.pdf.

HUG, Theo. **Mikrolernen – konzeptionelle Überlegungen und Anwendungsbeispiele**. [*S.l.*]: VS Verlag für Sozialwissenschaften, 2010. Disponível em: https://link.springer.com/chapter/10.1007/978-3-531-92135-8_12#citeas. Acesso em: 30 set. 2019.

IBÁNEZ, Ricardo Marin. **A educação a distancia**: Suas modalidades e economia. Rio de Janeiro: UCB, 1996.

INEP. **CENSO DA EDUCAÇÃO SUPERIOR**: Notas Estatísticas 2017. Brasil, 2017. Disponível em: http://download.inep.gov.br/educacao_superior/censo_superior/documentos/2018/censo_da_educacao_superior_2017-notas_estatisticas2.pdf. Acesso em: 25 ago. 2019.

JOHNSON, L *et al.* **NMC Horizon Report: 2014 Higher Education Edition**. Austin, Texas: The New Media Consortium, 2014. Disponível em: https://library.educause.edu/-/media/files/library/2014/1/2014hres.pdf. Acesso em: 20 out. 2019.

KIM, Bohyun. **Understanding Gamification**. Chicago, USA: American Library Association, 2015. Disponível em: https://journals.ala.org/ltr/issue/download/502/252. Acesso em: 9 out. 2019.

KORTH, Henry F.; SILBERSCHATZ, Abraham; SUDARSHAN, S. **Sistema de Banco de Dados**. 8. ed. [*S.l.*]: Makron Books, 1994.

LINDNER, Martin. **Use These Tools, Your Mind Will Follow. Learning in Immersive Micromedia & Microknowledge Environments**. [*S.l.*]: Citeseerx, 2006. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.98.7263. Acesso em: 30 set. 2019.

MAIA, Carmem; NETO, João Augusto Mattar. **ABC da EaD**: a educação a distância hoje. [*S.l.*]: Prentice Hall Brasil, 2007.

MORRISON, Michael. **Head First JavaScript**. Sebastopol, CA: O'Reilly Media, 2008. Disponível em: http://www.psu.edu.sa/OfficeCenters/deanoffice/DigitalLibrary/DocumentsLibrary/Documents/head-first-javascript.22.pdf.

MORVILLE, Peter. **User Experience Design**. [*S.l.*: *s.n.*], 2004. Disponível em: http://semanticstudios.com/user_experience_design/. Acesso em: 12 out. 2019.

MOZILLA DEVELOPER NETWORK. **O que é um servidor web (web server)**. [*S.l.*]: MDN, 2019. Disponível em: https://developer.mozilla.org/pt-

BR/docs/Learn/Common_questions/o_que_e_um_web_server. Acesso em: 22 set. 2019.

NICOLETTI, Bruna. **Avaliação diagnóstica**: utilize como ferramenta estratégica. [*S.l.*: *s.n.*], 2019. Disponível em: https://studos.com.br/avaliacao-diagnostica/. Acesso em: 1 nov. 2019.

NORMAN, Don. **The Definition of User Experience (UX)**. [*S.l.*]: Nielsen Norman Group, 1990. Disponível em: https://www.nngroup.com/articles/definition-user-experience/. Acesso em: 12 out. 2019.

NUNES, Marisa Fernandes. **As metodologias de ensino e o processo de conhecimento científico**. Curitiba: Setor de Educação da Universidade Federal do Paraná, 1993. Disponível em: https://revistas.ufpr.br/educar/article/view/36055/22244. Acesso em: 13 set. 2019.

OLIVEIRA, Aline Tatiane Evangelista de *et al.* **Ferramentas e estratégias de interação e comunicação na prática da tutoria em EaD**. Araxá: Evidências, 2017. Disponível em: https://www.uniaraxa.edu.br/ojs/index.php/evidencia/article/view/532. Acesso em: 11 out. 2019.

PILETTI, Nelson. **Didática Geral**. 8. ed. São Paulo: Ática, 1987. Disponível em: https://praxistecnologica.files.wordpress.com/2014/08/piletti_didatica-geral.pdf.

RIBEIRO, Elvia Nunes; ARAÚJO MENDONÇA, Gilda Aquino de; MENDONÇA, Alzino Furtado de. **A importância dos ambientes virtuais de aprendizagem na busca de novos domínios da EAD**. [*S.l.*: *s.n.*], 2007. Disponível em: http://www.abed.org.br/congresso2007/tc/4162007104526AM.pdf. Acesso em: 11 out. 2019.

ROBES, Jochen. **Microlearning und Microtraining**. [*S.l.*: *s.n.*], 2009. Disponível em: https://weiterbildungsblog.de/blog/2009/10/05/microlearning-und-microtraining-flexible-kurzformate-in-derweiterbildung. Acesso em: 30 set. 2019.

RODRIGUES, Alessandra Pereira *et al.* **Autoria e empacotamento de conteúdos**. Porto Alegre, RS, Brasil: Revista Novas Tecnologias na Educação, 2009. Disponível em: https://seer.ufrgs.br/renote/article/view/13503/8839. Acesso em: 24 out. 2019.

RODRIGUES, Gabriel Mário. **Os desafios da Educação a Distância**. [*S.l.*: *s.n.*], 2002. Disponível em: http://buscalegis.ccj.ufsc.br. Acesso em: 7 set. 2019.

ROVER, Aires Jose. **A educação a distância no ensino de graduação: contexto tecnológico e normativo**. Florianópolis, Brasil: [*s.n.*], 2003. Disponível em: http://www.egov.ufsc.br/portal/sites/default/files/publicacao-ead-ensino_de_graduacao_contexto_tecnologico_e_normativo_fragale.pdf. Acesso em: 7 set. 2019.

SANTOS, João Francisco Severo. **Avaliação no Ensino a Distância**. [*S.l.*]: Revista Iberoamericana de Educación (ISSN: 1681-5653), 2005. Disponível em: https://rieoei.org/deloslectores/1372Severo.pdf. Acesso em: 20 set. 2019.

SHARPLES, Mike *et al.* **Mobile learning**: Small devices, big issues. [*S.l.*]: Springer, 2009. Disponível em: https://www.academia.edu/745912/Mobile_Learning. Acesso em: 1 out. 2019.

SILVA FILHO, Roberto Leal Lobo e *et al.* **A evasão no ensino superior brasileiro**. Brasil: Cadernos de Pesquisa Scielo, 2007. Disponível em: http://www.scielo.br/pdf/cp/v37n132/a0737132. Acesso em: 8 out. 2019.

SILVA MELO, Rafaela da; CARVALHO, Marie Jane Soares. **Aplicativos educacionais livres para mobile learning**. [*S.l.*: *s.n.*], 2014. Disponível em: http://www.periodicos.letras.ufmg.br/index.php/anais_linguagem_tecnologia/article/view/5809/5098. Acesso em: 22 out. 2019.

SOUZA, Mazio Bennassi de; NEIVA, Frâncila Weidt. **Uso do kahoot como plataforma de apoio ao ensino em universidades**. Juiz de Fora, Brasil: IV Seminário de extensão e pesquisa, CES/JF, 2018. Disponível em: https://seer.cesjf.br/index.php/ANL/article/viewFile/1803/1148. Acesso em: 24 set. 2019.

STAKER, Heather; HORN, Michael B. **Classifying K–12 blended learning**. [*S.l.*]: Inno Sight Institute, 2012. Disponível em: https://www.christenseninstitute.org/wp-content/uploads/2013/04/Classifying-K-12-blended-learning.pdf. Acesso em: 6 out. 2019.

STATCOUNTER. **Desktop vs Mobile vs Tablet Market Share Worldwide**. [*S.l.*]: Statcounter Global stats, 2019. Disponível em: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet. Acesso em: 5 out. 2019.

TAKAHASHI, Tadao. **Sociedade da Informação no Brasil**. Brasilia: Ministério da Ciência e tecnologia, 2000.

THE EXPERIENCE API. **What is the Experience API?** [*S.l.*: *s.n.*], 2019. Disponível em: https://xapi.com/overview/. Acesso em: 11 out. 2019.

TRAXLER, John. **Defining mobile learning**. Wolverhampton, UK: IADIS International Conference Mobile Learning, 2005. Disponível em: https://www.researchgate.net/profile/John_Traxler/publication/228637407_Defining_mobile_learning/links/0deec51c8a2b531259000000/Defining-mobile-learning.pdf. Acesso em: 1 out. 2019.

UNITED NATIONS EDUCATIONAL, SCIENTIFIC e CULTURAL ORGANIZATION. **Policy guidelines for mobile learning**. Paris, França: UNESDOC Digital Library, 2013. Disponível em: https://unesdoc.unesco.org/ark:/48223/pf0000219641. Acesso em: 11 out. 2019.

VALENTINE, Doug. **Distance Learning: Promises, Problems, and Possibilities**. Oklahoma, USA: [*s.n.*], 1997. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.496.2781&rep=rep1&type=pdf. Acesso em: 17 ago. 2019.

VARGAS, Milton. **Para uma Filosofia da Tecnologia**. São Paulo: Alfa Omega, 1994.

VÁZQUEZ-CANO, Esteban. **Mobile Distance Learning with Smartphones and Apps in Higher Education**. Espanha: Educational Consultancy e Research Center, 2014. Disponível em: https://pdfs.semanticscholar.org/1fe2/79359f9985b13b34f0016cab5597a39363b1.pdf. Acesso em: 11 out. 2019.

VIANNA, Maurício *et al.* **Gamification, Inc.** Como reinventar empresas a partir de jogos. Rio de Janeiro: MJV Press, 2013.

ZICHERMANN, Gabe; CUNNINGHAM, Christopher. **Gamification by Design**: Implementing Game Mechanics in Web and Mobile Apps. Sebastopol, CA: O'Reilly Media, Inc, 2011.

APÊNDICE A – ARTIGO DO TRABALHO

Apêndice A – Artigo do trabalho
Ferramenta web de auxílio à avaliação de aprendizagem online.

Iran A. C. Alvarez[1]

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

fran.cardoso@grad.ufsc.br

Abstract. *The world and its relations are constantly changing, an academic education is one of them. The numbers of distance learning courses are designed to meet a demand for classes that are not in person in a traditional classroom, with a blackboard, slideshow, call list, exams, and tests. This can all be done through network technologies. Inserted in this context, this work started with a proposal to bring studies about one of the teaching stages, an evaluation. Research marker-based assessment applications that have an educational purpose, and ultimately develop a tool with features designed to increase or activate and execute student performance.*

Resumo. *O mundo e suas relações estão em constantes mudanças, a educação acadêmica é uma delas. Inúmeros cursos a distância tem surgido para suprir a demanda de alunos que não requerem a presença física em uma sala de aula tradicional, com quadro na parede, apresentação de slides, lista de chamada, provas e testes. Isto tudo já pode ser feito através de tecnologias em rede. Inserido nesse contexto, este trabalho foi iniciado com a proposta de trazer estudos sobre uma das etapas de ensino, a avaliação. Pesquisar sobre aplicativos de avaliação no mercado que tem uma proposta educacional e ao final se desenvolveu uma ferramenta com características que visam aumentar o engajamento e performance dos alunos.*

1. Introdução

Segundo dados do censo da educação superior divulgado pelo INEP (2017), mais de 46% das matrículas de cursos tecnológicos no Brasil já são a distância, que pelo estudo é explicado principalmente pelo aumento na oferta dessa modalidade nos últimos anos, que entre 2007 e 2017 cresceu 586% em relação ao também positivo aumento de número de matrículas de cursos presenciais.

Podemos relacionar a expansão da EaD com os benefícios que ela proporciona, que segundo Valentine(1997) podem ser classificados em três principais categorias: Custo-benefício, maior impacto por meio de mídias interativas atualizadas em tempo real e maior variedade de canais de comunicação entre conteúdo e aluno.

O processo de avaliação na educação é onipresente, existe em qualquer sistema de ensino, seja ele a distância ou não. Segundo Piletti(1987, p.190): "Avaliação é um

processo contínuo de pesquisas que visa interpretar os conhecimentos, habilidades e atitudes dos alunos, tendo em vista mudanças esperadas no comportamento, propostas nos objetivos educacionais, a fim de que haja condições de decidir sobre alternativas do planejamento do trabalho do professor e da escola como um todo".

O processo de avaliação tem como fim melhorar a qualidade do processo de aprendizagem, podendo ter basicamente três funções: função diagnóstica, função formativa e função somática (PILETTI, 1987; HAYDT, 2002).

Por função diagnóstica da avaliação entende-se o intuito de identificar o nível de conhecimento a quem é aplicada, para se ter ideia dos níveis de conhecimento individuais e grupais dos alunos. É aquela avaliação geralmente feita no começo do curso ou unidade de ensino para identificar se os alunos possuem conhecimentos básicos para a aprendizagem de novos conteúdos, podendo servir também para estimar problemas de aprendizagens e suas causas (HAYDT, 2002).

A função formativa da avaliação é aquela aplicada durante o processo de ensino, como o semestre letivo de uma disciplina. É uma forma de controle que visa informar o professor do rendimento do aluno, possíveis deficiências na organização do ensino e também paliativos ajustes que podem ser feitos para atingir os objetivos estipulados pelo plano de ensino (ALMEIDA, L., 2001). "Esse tipo de avaliação é basicamente um orientador dos estudos e esforços dos professores e alunos no decorrer desse processo, pois está muito ligada ao mecanismo de retro-alimentação (feed- back) que permite identificar deficiências e reformular seus trabalhos, visando aperfeiçoá-los em um ciclo contínuo e ascendente." (SANTOS, 2005, p. 2).

Por fim a função somativa é aquela efetuada ao final do curso, período letivo ou unidade de ensino, que tem como intuito de classificar os alunos quanto ao nível de conhecimento adquiridos durante tal período. Comumente tem o fim de promover o aluno de um nível a outro (HAYDT, 2002).

Inspirado no desafio que a educação superior enfrenta em motivar e engajar seus alunos, e os aplicativos da internet que ganham mais e mais usuários a cada dia, foi que esse trabalho se motivou. Se beneficiar da tecnologia para propor uma ferramenta que auxilie no engajamento e evolução dos alunos em disciplinas variadas. Utilizando estudos que já se propuseram a analisar os motivos da evasão em cursos do ensino superior, e também estudos que analisam características de aplicativos educacionais que levam ao maior número de usuários satisfeitos e motivados e terem um uso cada vez maior. Utilizando mais uma vez a tecnologia para tentar curar uma dor, nesse caso, no contexto acadêmico.

2. Desenvolvimento

O sistema é, no seu cerne, um criador e disponibilizador de atividades, onde existem dois principais papéis, o de professor e o de aluno. Os professores poderão criar questões de única ou múltipla escolha a serem respondidas pelos alunos. Os alunos responderão questões interativas e terão o feedback se responderam corretamente em tempo real.

Os professores terão um relatório geral e individual dos acertos e erros dos alunos em cada atividade.

Para engajar e motivar a utilização o desenvolvimento focou na simplicidade de uso, portabilidade, e interface amigável de modo a deixar o design da aplicação o mais atrativo possível e tornar o seu uso fácil e rápido, bastante inspirado nos aplicativos levantados no capítulo do estado da arte, de maneira onde o número reduzido de funções deixe o foco no objetivo principal que é a fixação dos conteúdos vistos em sala de aula.

2.1. Proposta de formas de uso da ferramenta

Um professor organiza suas aulas, criando vários planos de aula. Cada plano pode conter, segundo critérios do professor, diversos elementos, tais como:

- Identificação/Tema
- Objetivos/Finalidade/Duração
- Conteúdo
- Estratégia/procedimentos didáticos
- Recursos didáticos
- Métodos de avaliação

Como o foco deste trabalho é de reforçar o engajamento e permitir uma avaliação do aprendizado do estudante, mas também da própria atividade (do plano de aula e de seus elementos: estratégias e recursos adotados por exemplo), recomenda-se que o professor utilize essa ferramenta (Apprenda) como apoio à avaliação, nas seguintes situações:

a) tanto para fazer uma observação do conhecimento/habilidade prévia dos alunos.

b) quanto para avaliar o quanto a atividade didática projetada melhorou (potencialmente) o conhecimento do aluno sobre o tema.

c) e ainda, permitir avaliar (pelo conjunto de resultados), se o plano de aula atingiu os objetivos propostos (se houve uma melhora generalizada do conhecimento sobre o tema, pode-se avaliar que o plano de aula foi adequado, caso contrário, o professor deverá procurar adotar outras práticas/estratégias em atividades futuras).

2.2. A ferramenta

A aplicação final foi construída imaginando o seu uso por dois tipos de usuários: Aluno e professor.

O professor tem como ações principais a criação de atividades, disponibilização dessas atividades aos alunos e análise dos resultados obtidos pelos alunos.

O aluno tem uma atividade apenas, que é a de responder as atividades que foram designadas para ele. Podendo rever suas respostas nas atividades a qualquer momento, contanto que o professor as deixe disponíveis.

O professor é um papel que estende o perfil de aluno, portanto, ele assume as funções de responder as atividades que ele mesmo pode se matricular.

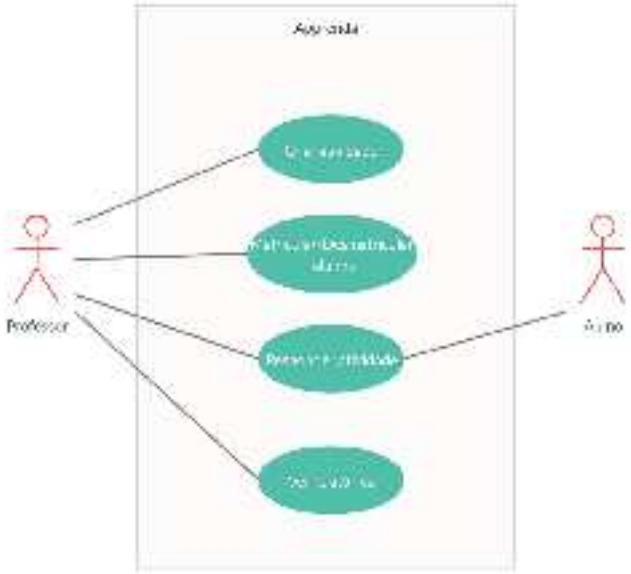


Figura 1. Diagrama de casos de uso da aplicação

As entidades da aplicação que estão armazenadas no banco de dados são apenas três: Usuário, Atividade e Respostas. Os atributos de cada entidade estão descritos no diagrama de classe abaixo.

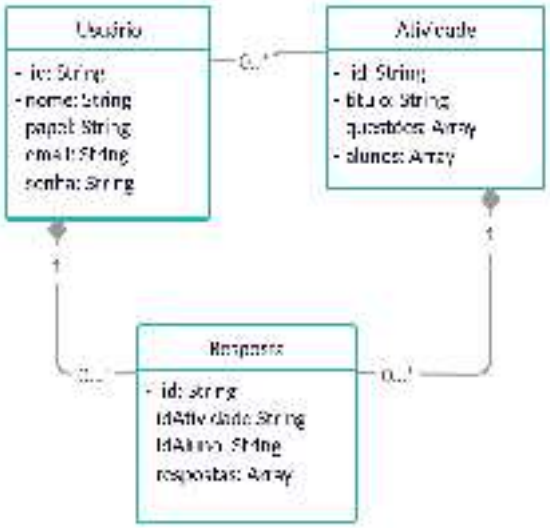


Figura 2. Diagrama de classes da aplicação

2.3. Tecnologias utilizadas e arquitetura

A escolha da aplicação ter sido desenvolvida para um ambiente web se deu por alguns pontos, entre eles a facilidade de uso tendo em vista que não é necessário a instalação de nenhum programa além de um navegador para utilizar a ferramenta, alta demanda e abrangência de aplicações web, portabilidade otimizada por poder rodar em qualquer ambiente que possua um navegador web, como celulares, tablets, e televisores.

A linguagem de programação que permeia toda a aplicação é o JavaScript, o que possibilita transitar entre o lado do servidor e cliente de maneira simples. No servidor é

utilizado o NodeJS um interpretador de JavaScript assincrono que permite leitura e escrita de arquivos, acesso a banco de dados.

O banco de dados utilizado foi o MongoDB, um banco não relacional que utiliza notação JSON para armazenar os documentos, possibilitando a utilização de funções comuns a linguagem para fazer consultas e escritas no banco.

Na parte do cliente foi utilizado o framework React, desenvolvido pelo Facebook. O React simplifica a criação de interfaces de usuários por utilizar o conceito de componentização que incentiva a criação de componentes genéricos que possam ser reutilizados pela aplicação.

A seguir um diagrama que elucida melhor a arquitetura da aplicação.

Figura 3. Arquitetura do sistema

2.4. Utilizando a ferramenta

A primeira página que o usuário vê é a de login, nela o usuário cadastrado entra com o e-mail e senha para ser autentificado no sistema e assim poder utilizar a ferramenta. É possível ter na mesma instância (cliente) quantos professores e alunos desejar, isso se viu necessário para quando há casos de disciplinas que possuem monitores ou até mesmo são ministradas por mais de um professor, dessa forma os professores e monitores compartilham e veem os relatórios de todos os alunos, além de poderem criar atividades a serem respondidas de maneira paralela, otimizando o processo de avaliação.

Figura 4. Tela de login

Como professor, aparece uma página de atividades e uma de relatórios. Na página de atividades ficam aquelas que foram previamente criadas, onde ao clicar em uma, vai para a parte de edição da mesma. Há também um botão para criação de novas atividades.

Figura 5. Lista de atividades professores

Ao criar ou editar uma atividade, o usuário adiciona questões de múltipla ou única escolha, e então cria as opções de resposta para a questão escolhendo aquela ou aquelas que são as corretas.

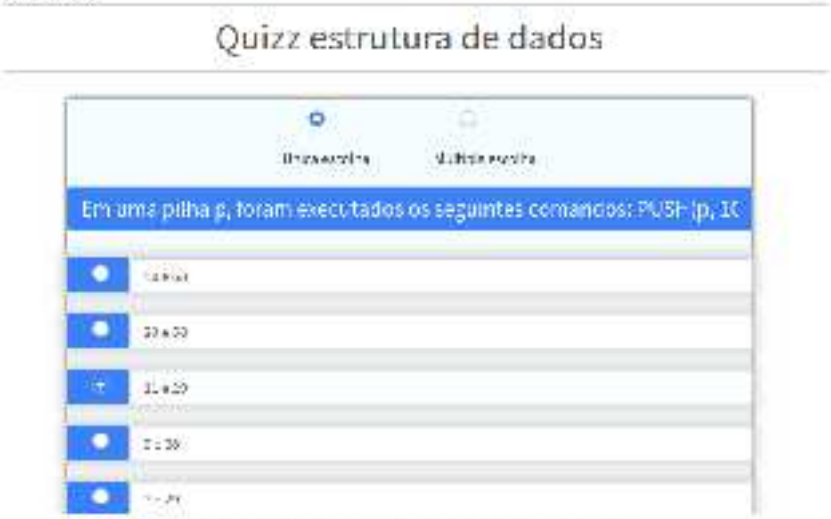


Figura 5. Criando uma nova atividade

Ao selecionar uma atividade tem-se a lista de usuários matriculados e aqueles que podem ser matriculados para começar a responder.

Figura 6. Matricular e desmatricular alunos

A página de relatórios é separada por relatórios gerais onde se tem informações sobre número de atividades cadastradas, números de usuários por perfil, e total de respostas. Além disso, nesta página exite uma lista de usuários que ao selecionar um, redireciona para as informações detalhadas daquele aluno e as respostas dadas em cada atividade. Por fim uma lista de atividades que, selecionando qualquer uma, leva a detalhes da performance dos alunos.

Figura 7. Relatório geral

Figura 8. Relatório de uma atividade

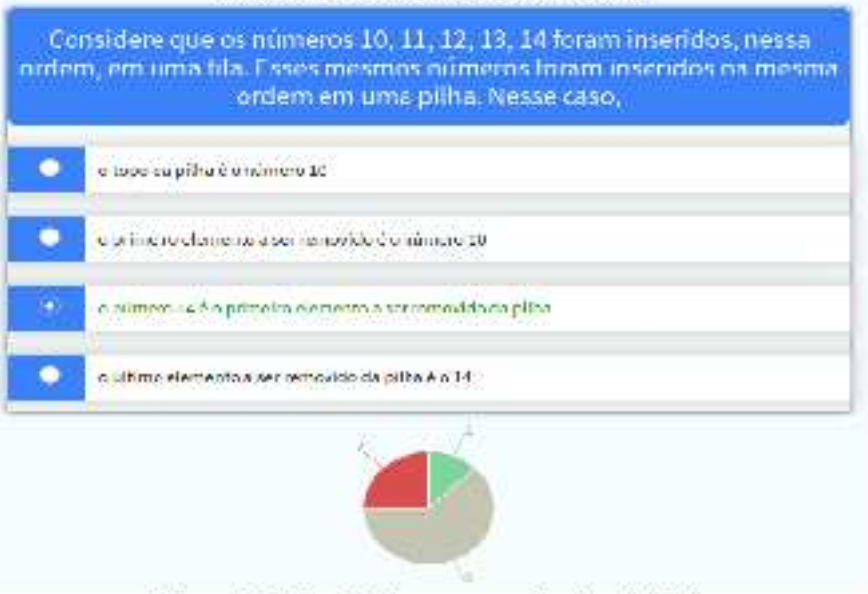


Figura 9. Relatório de uma questão da atividade

Figura 10. Relatório de aluno em uma atividade

O perfil de aluno é o mais simples de propósito, porque o único foco dele é responder as atividades e receber o feedback de acerto ou erro. Na tela principal tem uma lista de atividades que lhe foram atribuídas, e ao selecionar uma ele pode respondê-la. O aluno ainda pode utilizar as respostas dadas como gabarito, de maneira que as respostas ficam salvas para eventuais consultas. Essa é uma maneira de relembrar os conceitos vistos em aula, e gravar na memória por meio da repetição.

Figura 11. Lista de atividades de um aluno

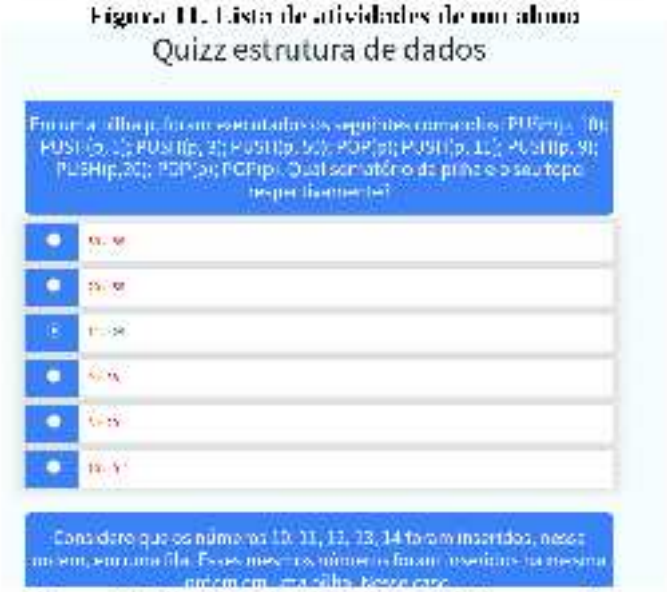


Figura 12. Visualização de uma atividade como aluno

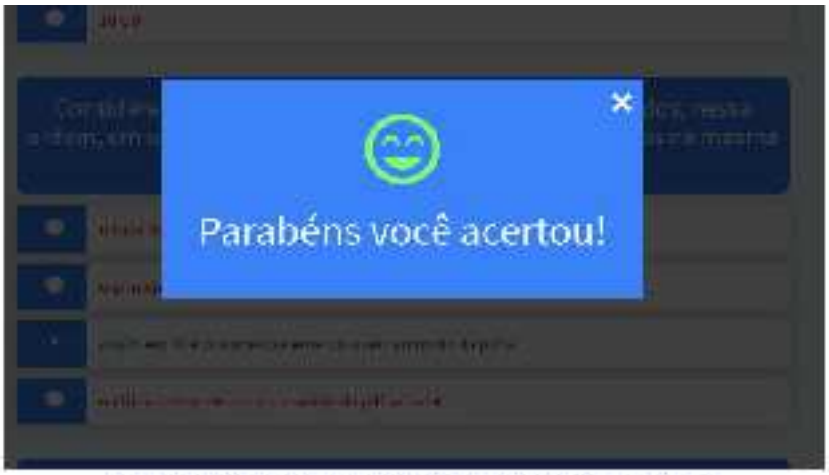


Figura 13. Feedback de uma atividade para o aluno

3. Considerações finais

Aprendizado não se dá em um evento único. Quando aprendemos um conceito novo, é necessário exercitá-lo para que fique na memória por mais tempo. Neste trabalho foi desenvolvido uma ferramenta que ajuda a reter conhecimento e exercitar conceitos vistos em sala de aula por meio de atividades de única ou múltipla escolha. Além disso a ferramenta consegue compilar dados de respostas dos alunos para que sejam analisados e sirvam de base para tomadas de decisões, como por exemplo, uma sala em que a maioria dos respondentes errou a mesma questão, se faz necessária uma revisão do conteúdo para sanar dúvidas do mesmo. Também se faz possível uma ação individual, uma vez que se tem dados separados por aluno. A avaliação é uma importante ferramenta para se dar direcionamento aos estudos e também identificar se os conhecimentos foram bem compreendidos pelos receptores.

Foram estudadas ferramentas que são amplamente utilizadas e tem um viés educacional, e delas foi buscado inspiração nas características que as tornam interativas e divertidas de utilizar. Com isto foi desenvolvido a prova de conceito com elementos de aprendizado gamificado, microlearning, totalmente web e com suporte a relatórios para professores e tutores.

Por fim a ferramenta se concebeu na vontade de auxiliar o aprendizado, dando uma opção simples e eficaz de experienciar o conhecimento adquirido previamente por meio das aulas regulares em sala, como é feito pelo bleended learning. Qualquer momento que o aluno tenha disponível em seu dia e/ou que tenha mais vontade, pode utilizar o sistema para relembrar e testar o que aprendeu, tanto em um computador pessoal como no seu smartphone, respondendo questões objetivas criadas pelos professores.

Apesar de ser mais um entre tantos aplicativos com a mesma proposta, o sistema se diferencia principalmente daqueles elencados no estado da arte por alguns pontos chaves.

- É uma ferramenta totalmente gratuita e aberta, onde seu código pode ser reutilizado, aprimorado e alterado por qualquer um que assim deseje. Pode ser inclusive utilizado para o estudo de programação por meio da análise do código desenvolvido.

- Totalmente web, possibilitando o seu uso em qualquer dispositivo que possua um navegador de internet, contando com responsividade de design, onde a aplicação se molda do melhor jeito ao tamanho da tela.
- Não é preciso muitas configurações para começar a usar a ferramenta. A princípio só prover questionários e fazer o gerenciamento de alunos que vão estar atrelados a cada um.
- Analise de dados tanto gerais como específicos por questionário ou aluno. Disponibilizando os dados numa interface amigável e simples.

4. Referências

INEP. **CENSO DA EDUCAÇÃO SUPERIOR: Notas Estatísticas 2017**. Brasil. 2017. Disponível em: http://download.inep.gov.br/educacao_superior/censo_superior/documentos/2018/censo_da_educacao_superior_2017-notas_estatisticas2.pdf. Acesso em: 25 ago. 2019.

VALENTIN, Doug. **Distance Learning: Promises, Problems, and Possibilities**. Oklahoma, USA: [s.n.], 1997. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.496.2781&rep=rep1&type=pdf. Acesso em: 17 ago. 2019.

PILETTI, Nelson. **Didática Geral**. 5. ed. São Paulo: Ática, 1987. Disponível em: http://praxisacademica.files.wordpress.com/2014/08/piletti-didatica-geral.pdf.

HAYDT, Regina Celia Cazaux. **Avaliação do processo ensino-aprendizagem**. São Paulo: Ática, 2002. Disponível em: https://professorjuliocesar.files.wordpress.com/2010/08/didc3a1tica-texto-1.pdf.

ALMEIDA, Lecrdio. **A avaliação da aprendizagem escolar e a função social da escola**. São Paulo: Programa de Pós-Graduação em História e Filosofia da Educação. Pontifícia Universidade Católica de São Paulo, 2001.

SANTOS, João Francisco Severo. **Avaliação no Ensino a Distância.** S.l.]: Revista Iberoamericana de Educação (ISSN: 1681-5653), 2005. Disponível em: https://rieoei.org/historico/documentos/rie37a02.pdf. Acesso em: 20 set. 2019.

APÊNDICE B – CÓDIGO FONTE

Apendice B – Código fonte

O código fonte também pode ser encontrado em: https://github.com/francardoso/aprenda-front e https://github.com/francardoso/aprenda.

Back-end:

Main:

```
const express = require('express');
const session = require('express-session');
const MongoStore = require('connect-mongo')(session);
const cors = require('cors');
const mongoose = require('mongoose');
const app = express();
const routesInit = require('./routes');
const bodyParser = require('body-parser');
const settings = require('../settings');

mongoose.connect(settings.DB_URL,
    {
        useNewUrlParser: true,
        useFindAndModify: false,
        useUnifiedTopology: true
    }
);

const db = mongoose.connection;

db.on('error', console.error.bind(console, 'connection error:'));

//connect do MongoDB
db.once('open', () =>{
    console.log('connected to DB');
    initServer();
});

function initServer(){
    app.use(cors({
        origin: settings.CORS_ALLOWED_URL,
        credentials:true,
        optionsSuccessStatus: 200
    }));
    // session
    app.set('trust proxy', 1); // trust first proxy
    app.use(session({
        secret: settings.SESSION_SECRET,
        resave: true,
        saveUninitialized: true,
            cookie: {secure:false}, // MAYBE WILL HAVE TO CHANGE ON
PRODUCTION
        store: new MongoStore({
            url: settings.DB_URL
        })
    }));
```

```
    app.use(bodyParser.json());
    // routes debbug
    app.use((req,res,next)=>{
        console.log(req.method, req.url);
        next();
    });
    routesInit(app);
     app.listen(settings.PORT, () => console.log(`Server Listening on
port ${settings.PORT}`));
}
```

Rotas

```
const getUsersController = require('./controllers/getUsers');
const loginController = require('./controllers/login');
const logoutController = require('./controllers/logout');
const addUserController = require('./controllers/addUser');
const deleteUserController = require('./controllers/deleteUser');
const isLoggedController = require('./controllers/isLogged');
const                    getUserByLoginController                    =
require('./controllers/getUserByLogin');
const addLessonController = require('./controllers/addLesson');
const                     getAllLessonsController                    =
require('./controllers/getAllLessons');
const getLessonController = require('./controllers/getLesson');
const                    enrollStudentsController                    =
require('./controllers/enrollStudents');
const                   unenrollStudentsController                   =
require('./controllers/unenrollStudents');
const                  getStudentLessonsController                   =
require('./controllers/getStudentLessons');
const              checkLessonQuestionAnswerController               =
require('./controllers/checkLessonQuestionAnswer');
const                   getStudentAnswerController                   =
require('./controllers/getStudentAnwser');
const editLessonController = require('./controllers/editLesson');

const                    getAnswersTotalController                   =
require('./controllers/reports/getAnswersTotal');
const                   getStudentReportController                   =
require('./controllers/reports/getStudentReport');
const                    getLessonReportController                   =
require('./controllers/reports/getLessonReport');

function routesInit(app){
    app.get('/', (req, res)=>{
        res.send('Hello from Aprenda');
    });

    app.post('/login', loginController);
    app.get('/isLogged', isLoggedController);
    app.post('/logout', logoutController);

    app.post('/addUser', addUserController);
    app.get('/getUserByLogin', getUserByLoginController);
    app.delete('/deleteUser', deleteUserController);
    app.get('/getUsers', getUsersController);
```

```
    app.get('/getLesson', getLessonController);
    app.post('/addLesson', addLessonController);
    app.put('/editLesson', editLessonController);
    app.get('/getAllLessons', getAllLessonsController);
    app.get('/getStudentLessons', getStudentLessonsController);
    app.put('/enrollStudents', enrollStudentsController);
    app.put('/unenrollStudents', unenrollStudentsController);

                                  app.post('/checkLessonQuestionAnswer',
checkLessonQuestionAnswerController);
    app.get('/getStudentAnswer', getStudentAnswerController);

    app.get('/getAnswersTotal', getAnswersTotalController);
    app.get('/getStudentReport', getStudentReportController);
    app.get('/getLessonReport', getLessonReportController);
}

module.exports = routesInit;
```

Controllers

```
const unenrollStudentsModel = require('../models/unenrollStudents');
async function unenrollStudents (req,res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).send({error: validationMessage});
    }else{
        const ans = await unenrollStudentsModel(req.body);
        if(!ans.error){
            res.status(200).send(ans);
        }else{
            res.status(400).send({error: ans.error});
        }
    }
};

function validateParameters(body){
    const { idLesson, students } = body;
       if(typeof  idLesson  !==  "string"  ||  idLesson  ===  "")  return
"INVALID ID LESSON";
    if(!Array.isArray(students)) return "INVALID ARRAY OF STUDENTS";

    return null;
};

module.exports = unenrollStudents;
const loginModel = require('../models/login');
async function login(req, res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).json({error:validationMessage});
    }else{
        req.session.destroy();
        res.status(200).json({error: null});
    }
};

function validateParameters(body){
    return null;
```

```
};
module.exports = login;
const loginModel = require('../models/login');
async function login(req, res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).json({error:validationMessage});
    }else{
        const ans = await loginModel(req.body);
        if(!ans.error){
            if(!req.session.APRENDA){
                req.session.APRENDA = {};
            }
            req.session.APRENDA.idUser = ans.idUser;
            req.session.APRENDA.role = ans.role;
            res.status(200).send(ans.idUser);

        }else{
            res.status(200).json(ans);
        }
    }
};

function validateParameters(body){
    const {email, password} = body;

      if(typeof email !== "string" || email === "") return "INVALID
EMAIL";
       if(typeof password !== "string" || password === "") return
"INVALID PASSWORD";

    return null;
};
module.exports = login;
function isLogged(req, res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).send(validationMessage);
    }else{
                          if(req.session   &&   req.session.APRENDA   &&
req.session.APRENDA.idUser){
            res.status(200).send({
                isLogged: true,
                idUser: req.session.APRENDA.idUser
            });
        }else{
            res.status(200).send({
                isLogged: false,
            })
        }
    }
};
function validateParameters(body){
    return null;
};

module.exports = isLogged;
const getUsersModel = require('../models/getUsers');
```

```
async function getUsers(req, res){
    if(!validateParameters(req.body)){
        res.status(400).send('MISSING PARAMATERS');
    }else{
        const ans = await getUsersModel();
        res.status(200).send(ans);
    }
}

function validateParameters(body){
    return true
};
module.exports = getUsers;
const getUserByLoginModel = require('../models/getUserByLogin');
async function getUserByLogin(req, res){
    const validationMessage = validateParameters(req.query);
    if(validationMessage){
        res.status(400).send(validationMessage);
    }else{
        const ans = await getUserByLoginModel(req.query);
        res.status(200).send(ans);
    }
};
function validateParameters(body){
    const {email} = body;
      if(typeof email !== "string" || email === "") return "INVALID
EMAIL";
    return null;
};

module.exports = getUserByLogin;
const getStudentLessonsModel = require('../models/getStudentLessons');

async function getStudentLessons(req,res){
    const validationMessage = validateParameters(req.query);
    if(validationMessage){
        res.status(400).json({error: validationMessage});
    }else{
        const ans = await getStudentLessonsModel(req.query);
        res.status(200).json(ans);
    }
};

function validateParameters(body){
    const { idStudent } = body;

      if(typeof idStudent !== 'string' || idStudent === "") return
"INVALID ID STUDENT";

    return null;
};

module.exports = getStudentLessons;
const getStudentAnswerModel = require('../models/getStudentAnswer');
async function getLesson(req,res){
    const validationMessage = validateParameters(req.query);
    if(validationMessage){
        res.status(400).send({error:validationMessage});
```

```
    }else{
        const ans = await getStudentAnswerModel(req.query);
        res.status(200).json(ans);
    }
};

function validateParameters(body){
    const { idLesson, idStudent } = body;
       if(typeof idLesson !== "string" || idLesson === "") return
"INVALID ID LESSON";
      if(typeof idStudent !== "string" || idStudent === "") return
"INVALID ID STUDENT";

    return null;
};

module.exports = getLesson;
const getLessonModel = require('../models/getLesson');
async function getLesson(req,res){
    const validationMessage = validateParameters(req.query);
    if(validationMessage){
        res.status(400).send({error:validationMessage});
    }else{
        const ans = await getLessonModel(req.query);
        res.status(200).send(ans);
    }
};

function validateParameters(body){
    const { idLesson } = body;
       if(typeof idLesson !== "string" || idLesson === "") return
"INVALID ID LESSON";

    return null;
};

module.exports = getLesson;
const getAllLessonsModel = require('../models/getAllLessons');
async function getAllLessons(req,res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).send(validationMessage);
    }else{
        const ans = await getAllLessonsModel(req.body);
        res.status(200).send(ans);
    }
};

function validateParameters(body){
    return null;
};

module.exports = getAllLessons;
const enrollStudentsModel = require('../models/enrollStudents');
async function enrollStudents(req, res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).send({error: validationMessage});
```

```
    }else{
        const ans = await enrollStudentsModel(req.body);
        if(!ans.error){
            res.status(200).send(ans);
        }else{
            res.status(400).send({error: ans.error});
        }
    }
};
function validateParameters(body){
    const { idLesson, students } = body;
       if(typeof idLesson !== "string" || idLesson === "") return
"INVALID ID LESSON";
    if(!Array.isArray(students)) return "INVALID ARRAY OF STUDENTS";

    return null;
};

module.exports = enrollStudents;
const editLessonModel = require('../models/editLesson');
async function editLesson(req,res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).send(validationMessage);
    }else{
        const ans = await editLessonModel(req.body);
        res.status(200).send(ans);
    }
};

function validateParameters(body){
    const {idLesson} = body;

     if(!idLesson || typeof idLesson !== 'string') return "INVALID ID
LESSON";

    return null;
};

module.exports = editLesson;
const deleteUserModel = require('../models/deleteUser');
async function deleteUser(req, res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).send(validationMessage);
    }else{
        const ans = await deleteUserModel(req.body);
        res.status(200).send(ans)
    }
};
function validateParameters(body){
    const {idUser} = body

     if(typeof idUser !== "string" || idUser === "") return "INVALID
USER ID";

    return null;
};
```

```
module.exports = deleteUser;
const                 checkLessonQuestionAnswerModel                =
require('../models/checkLessonQuestionAnswer');

async function checkLessonQuestionAnswer(req,res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).json({error: validationMessage});
    }else{
        const ans = await checkLessonQuestionAnswerModel(req.body);
        res.status(200).send(ans);
    }
};

function validateParameters(body){
    const { idLesson, idStudent,questionIndex, answer } = body;

       if(typeof  idLesson  !==  'string'  ||  idLesson  ===  "")  return
"INVALID ID LESSON";
      if(typeof  idStudent  !==  'string'  ||  idStudent  ===  "")  return
"INVALID ID STUDENT";
    if(isNaN(questionIndex)) return "INVALID QUESTION INDEX";
      if(answer  &&  !Array.isArray(answer))  return  "INVALID  ARRAY  OF
ANSWER"

    return null;
};

module.exports = checkLessonQuestionAnswer;
const addUserModel = require('../models/addUser');
async function addUser(req, res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).send(validationMessage);
    }else{
        const ans = await addUserModel(req.body);
        if(!ans.error){
            res.status(200).send({idUser: ans.id});
        }else{
            res.status(400).send({error: ans.error});
        }
    }
};
function validateParameters(body){
    const {email, name, role, password} = body
    const roles = ['professor', 'student'];

      if(typeof  email  !==  "string"  ||  email  ===  "")  return  "INVALID
EMAIL";
    if(typeof name !== "string" || name === "") return "INVALID NAME";
    if(roles.indexOf(role) < 0 ) return "INVALID ROLE";
       if(typeof  password  !==  "string"  ||  password  ===  "")  return
"INVALID PASSWORD";

    return null;
};
```

```
module.exports = addUser;
const addLessonModel = require('../models/addLesson');
async function addLesson(req,res){
    const validationMessage = validateParameters(req.body);
    if(validationMessage){
        res.status(400).send(validationMessage);
    }else{
        const ans = await addLessonModel(req.body);
        res.status(200).send(ans);
    }
};

function validateParameters(body){
    return null;
};

module.exports = addLesson;
```

```
const                      getStudentReportModel                      =
require('../../models/reports/getStudentReport');

async function getStudentReport(req,res){
    const validationMessage = validateParameters(req.query);
    if(validationMessage){
        res.status(400).json({error: validationMessage});
    }else{
        const ans = await getStudentReportModel(req.query);
        res.status(200).json(ans);
    }
};

function validateParameters(body){
    const { idStudent } = body;
      if(typeof idStudent !== 'string' || idStudent === "") return
"INVALID ID STUDENT";
    return null;
};

module.exports = getStudentReport;
const                       getLessonReportModel                       =
require('../../models/reports/getLessonReport');
async function getLessonReport(req,res){
    const validationMessage = validateParameters(req.query);
    if(validationMessage){
        res.status(400).json({error: validationMessage});
    }else{
        const ans = await getLessonReportModel(req.query);
        res.status(200).json(ans);
    }
}

function validateParameters(body){
    const { idLesson } = body;
       if(typeof idLesson !== 'string' || idLesson === "") return
"INVALID ID LESSON";
    return null;
};
```

```
module.exports = getLessonReport;

const                        getAnswersTotalModel                        =
require('../../models/reports/getAnswersTotal');

async function getAnswersTotal(req,res){
    const validationMessage = validateParameters(req.query);
    if(validationMessage){
        res.status(400).json({error: validationMessage});
    }else{
        const ans = await getAnswersTotalModel();
        res.status(200).json(ans);
    }
};

function validateParameters(body){
    return null;
};

module.exports = getAnswersTotal;
```

Models

```
const Answer = require('../schemas/Answer');

async function updateAnswer(context){
    const { idLesson, idStudent, question } = context;
                          let      lessonAnswer      =            await
findStudentAnsToLesson(idLesson,idStudent);
    if(!lessonAnswer){
        const newLessonAns = new Answer({
            ...context
        });
        lessonAnswer = await newLessonAns.save();
    }
                                    const        questionIndex        =
lessonAnswer.questions.findIndex(el=>question.index === el.index);
    const newQuestions = [...lessonAnswer.questions];
    if(questionIndex === -1 ){
        newQuestions.push(question);
    }else{
        newQuestions[questionIndex] = question;
    }
       const  newAnswer  =  await  updateLessonAswers(lessonAnswer._id,
newQuestions);
    return new Promise((resolve,reject)=>{
        resolve(newAnswer.questions);
    });
};

function findStudentAnsToLesson(idLesson, idStudent){
    return new Promise((resolve,reject)=>{
        Answer.findOne({idLesson,idStudent}, (err,answer)=>{
            if(err){
                reject(err);
            }else{
```

```
                resolve(answer);
            }
        })
    });
};

function updateLessonAswers(idAnswer,questions){
    const queryOptions ={
        new: true,
    }
    return new Promise((resolve,reject)=>{
            Answer.findByIdAndUpdate(idAnswer,{questions},queryOptions,
(err, answer)=>{
            if(err){
                reject(err);
            }else{
                resolve(answer);
            }
        });
    });
};


module.exports = updateAnswer;
const Lesson = require('../schemas/Lesson');
async function unenrollStudents (context){
    const { idLesson, students } = context;
    const lesson = await findLesson(idLesson);
    if(!lesson) return {error: 'LESSON NOT FOUND'}
    const previousStudents = lesson.students;

    const updateStudents = previousStudents.reduce((acc,curr)=>{
        if(students.indexOf(curr.toString()) === -1){
            acc.push(curr);
        }
        return acc;
    },[]);

                    const     lessonWithUpdatedStudents     =     await
enrollStudentsToLesson(idLesson, updateStudents);

    return lessonWithUpdatedStudents.students;
};

function findLesson(id){
    return new Promise((resolve, reject)=>{
        Lesson.findById(id,(err, lesson)=>{
            if(err){
                reject(err);
            }else{
                resolve(lesson);
            }
        })
    });
};

function enrollStudentsToLesson(idLesson, students){
    const queryOptions = {
```

```
        new:true
    };
    return new Promise((resolve, reject)=>{
                Lesson.findByIdAndUpdate(idLesson,  {students:students},
queryOptions,(err, lesson)=>{
            if(err){
                reject(err);
            }else{
                resolve(lesson)
            }
        })
    })
}

module.exports = unenrollStudents;
const User = require('../schemas/User');
async function login(context){
    const user = await findUser(context.email);
    if(!user){
        return {
            error: 'USER_NOT_FOUND',
        }
    }else{
                                          const     isValidPassword     =
user.validatePassword(context.password);
        if(!isValidPassword){
            return {
                error: 'PASSWORD_INCORRECT'
            }
        }else{
            return {idUser: user._id, role: user.role}
        }
    }
}
function findUser(email){
    return new Promise((resolve, reject) =>{
        User.findOne({email: email}, (err, user) =>{
            if(err){
                reject(err);
            }else{
                resolve(user);
            }
        });
    });
}
module.exports = login;
const User = require('../schemas/User');
async function getUsers(context){
    const users = await findUsers();
    const usersInformations = users.map(user=>({
        _id: user._id,
        name: user.name,
        email: user.email,
        role: user.role
    }))
    return usersInformations;
};
```

```
function findUsers(){
    return new Promise((resolve, reject) =>{
        User.find({},(err, users) =>{
            if(err){
                reject(err);
            }else{
                resolve(users);
            }
        });
    });
}
module.exports = getUsers;
const User = require('../schemas/User');
async function getUserByLogin(context){
    const user = await findUser(context.email);
    if(user){
        return {
            user: {
                idUser: user.id,
                role: user.role
            }
        }
    }else{
        return {
            error: 'USER NOT FOUND IN DB'
        }
    }
};

function findUser(email){
    return new Promise((resolve, reject) =>{
        User.findOne({email: email}, (err, user) =>{
            if(user){
                resolve(
                    user
                );
                return;
            }
            resolve(null);
        });
    });
}
module.exports = getUserByLogin;
const Lesson = require('../schemas/Lesson');
async function getStudentLessons(context){
    const { idStudent } = context;
    const lessons = await findStudentLessons(idStudent);
    return lessons;
};

function findStudentLessons(idStudent){
    return new Promise((resolve, reject)=>{
        Lesson.find({ students: idStudent},(err, lessons)=>{
            if(err){
                reject(err);
            }else{
                resolve(lessons);
            }
```

```
        });
    });
}

module.exports = getStudentLessons;
const Answer = require('../schemas/Answer');

async function getStudentAnswer(context){
    const { idLesson, idStudent } = context;
    const answer = await findAnswer(idLesson, idStudent);

    return answer ? answer.questions : [];
};

function findAnswer(idLesson, idStudent,){
    return new Promise((resolve, reject)=>{
        Answer.findOne({idLesson,idStudent}, (err,answer)=>{
            if(err){
                reject(err);
            }else{
                resolve(answer);
            }
        });
    });
};

module.exports = getStudentAnswer;
const Lesson = require('../schemas/Lesson');
async function getLesson(context){
    const lesson = await findLesson(context.idLesson);
    if(!lesson){
        return {
            error: 'LESSON NOT FOUND',
        }
    }else{
        return lesson;
    }
};

function findLesson(id){
    return new Promise((resolve, reject)=>{
        Lesson.findById(id,(err, lesson)=>{
            if(err){
                reject(err);
            }else{
                resolve(lesson);
            }
        })
    });
}

module.exports = getLesson;
const Lesson = require('../schemas/Lesson');
async function getAllLessons(context){
    const lessons = await findLessons();
    return lessons;
};
```

```
function findLessons(){
    return new Promise((resolve, reject) =>{
        Lesson.find({},(err, lessons) =>{
            if(err){
                reject(err);
            }else{
                resolve(lessons);
            }
        });
    });
}

module.exports = getAllLessons;
const Lesson = require('../schemas/Lesson');
async function enrollStudents(context){
    const { idLesson, students } = context;
    const lesson = await findLesson(idLesson);
    if(!lesson) return {error: 'LESSON NOT FOUND'}
    const previousStudents = lesson.students;

    const newStudents = students.reduce((acc, curr)=>{
        if(previousStudents.indexOf(curr) === -1){
            acc.push(curr);
        }
        return acc;
    },[]);

                        const      lessonWithNewStudents      =      await
enrollStudentsToLesson(idLesson,
[...previousStudents, ...newStudents]);

    return lessonWithNewStudents.students;

};

function findLesson(id){
    return new Promise((resolve, reject)=>{
        Lesson.findById(id,(err, lesson)=>{
            if(err){
                reject(err);
            }else{
                resolve(lesson);
            }
        })
    });
};

function enrollStudentsToLesson(idLesson, students){
    const queryOptions = {
        new:true
    };
    return new Promise((resolve, reject)=>{
                Lesson.findByIdAndUpdate(idLesson,  {students:students},
queryOptions,(err, lesson)=>{
            if(err){
                reject(err);
            }else{
                resolve(lesson)
```

```
            }
        })
    })
}

module.exports = enrollStudents;
const Lesson = require('../schemas/Lesson');

async function editLesson(context){
    const { idLesson } = context;
    const lesson = await findLesson(idLesson);

    if(!lesson) return{error: "LESSON NOT FOUND"};

    const newLesson = {
        ...lesson._doc,
        ...context
    };

    const editedLesson = editLessonOnDB(idLesson, newLesson);

    return editedLesson;
};

function findLesson(id){
    return new Promise((resolve, reject)=>{
        Lesson.findById(id,(err, lesson)=>{
            if(err){
                reject(err);
            }else{
                resolve(lesson);
            }
        })
    });
};

function editLessonOnDB(id, newLesson){
    const queryOptions = {
        new:true
    };
    return new Promise((resolve, reject)=>{
            Lesson.findByIdAndUpdate(id,  newLesson,  queryOptions,(err,
lesson)=>{
            if(err){
                reject(err);
            }else{
                resolve(lesson)
            }
        })
    });
}

module.exports = editLesson;
const User = require('../schemas/User');
async function addUser(context){
    const deletedUser = await deleteUser(context.idUser);
    console.log('deleted user', deletedUser);
    if(deletedUser){
```

```
        return {
            error: null,
            idUser: deletedUser.id
        }
    }else{
        return {
            error: 'USER NOT FOUND IN DB'
        }
    }
};

function deleteUser(id){
    return new Promise((resolve, reject) =>{
        User.findByIdAndRemove(id, (err, user) =>{
            if(err){
                reject(err);
            }else{
                resolve(user);
            }
        });
    });
}
module.exports = addUser;
const Lesson = require('../schemas/Lesson');
const updateAnswer = require('./updateAnswer');

async function checkLessonQuestionAnswer(context){
      const { idLesson, questionIndex, answer = [], idStudent } =
context;
    const lesson = await findLesson(idLesson);
    if(!lesson) return {error: 'LESSON NOT FOUND'};
    const question = lesson.questions[questionIndex]
       const questionAnswers = question.options.reduce((acc, option,
index)=>{
        if(option.selected){
            acc.push(index);
        }
        return acc;
    },[]);

    const answersComparison = await updateAnswer({
        idLesson,
        idStudent,
        question: {
            index: questionIndex,
            questionAnswers,
            studentAnswers: answer
        }
    });
    return answersComparison;
};

function findLesson(id){
    return new Promise((resolve, reject)=>{
        Lesson.findById(id,(err, lesson)=>{
            if(err){
                reject(err);
            }else{
```

```
                resolve(lesson);
            }
        })
    });
}

module.exports = checkLessonQuestionAnswer;
const User = require('../schemas/User');
async function addUser(context){
    const user = await findUser(context.email);
    if(user) return {error: 'USER EMAIL ALREADY EXISTS'}

    const newUser = new User({
        ...context
    });
    newUser.hashed_password = newUser.hashPassword(context.password);
    const savedUser = await newUser.save();
    return {
        error: null,
        id: savedUser.id
    }
};

function findUser(email){
    return new Promise((resolve, reject) =>{
        User.findOne({email: email}, (err, user) =>{
            if(user){
                resolve({
                    user
                });
                return;
            }
            resolve(null);
        });
    });
}
module.exports = addUser;
const Lesson = require('../schemas/Lesson');
async function addLesson(context){
    const newLesson = new Lesson({
        ...context
    });
    const savedLesson = await newLesson.save();
    return savedLesson;
};

module.exports = addLesson;
```

```
const User = require('../../schemas/User');
const Answer = require('../../schemas/Answer');
const Lesson = require('../../schemas/Lesson');

async function getStudentReport(context) {
    const { idStudent } = context;
    const student = await findStudent(idStudent);
    if(!student) return {error: "STUDENT NOT FOUND"}
    const studentLessons = await findStudentLessons(idStudent);
```

```
    const studentAnswers = await findStudentAnswers(idStudent);

    const lessons = studentLessons.map((lesson) =>{
        const les = {...lesson.toObject()};
                    const   answer   =   studentAnswers.find((ans)   =>
ans.idLesson.toString() === lesson._id.toString());
        if(answer){
            answer.questions.forEach((ques) =>{
                les.questions[ques.index].answer = ques;
            });
        }
        return les;
    });
    return {
        _id: student._id,
        role: student.role,
        name: student.name,
        email: student.email,
        lessons
    };
};

function findStudent(idStudent){
    return new Promise((resolve, reject) =>{
        User.findById(idStudent,(err, User) =>{
            if(err){
                reject(err);
            }else{
                resolve(User);
            }
        });
    });
}

function findStudentLessons(idStudent) {
    return new Promise((resolve, reject) => {
        Lesson.find({ students: idStudent }, (err, lessons) => {
            if (err) {
                reject(err);
            } else {
                resolve(lessons);
            }
        });
    });
}

function findStudentAnswers(idStudent) {
    return new Promise((resolve, reject) =>{
        Answer.find({idStudent}, (err, answers) =>{
            if(err){
                reject(error);
            }else{
                resolve(answers);
            }
        });
    });
}
```

```
module.exports = getStudentReport;
const Lesson = require('../../schemas/Lesson');
const Answer = require('../../schemas/Answer');
async function getLessonReport(context){
    const { idLesson } = context;
    const lesson = await findLesson(idLesson);
    if(!lesson) return {error:'NO LESSON FOUND'}
    const lessonAnswers = await findLessonAnswers(idLesson);
    return joinLessonWithAnswers(lesson.toObject(), lessonAnswers);
}

function findLesson(id){
    return new Promise((resolve, reject)=>{
        Lesson.findById(id,(err, lesson)=>{
            if(err){
                reject(err);
            }else{
                resolve(lesson);
            }
        })
    });
}

function findLessonAnswers(idLesson){
    return new Promise((resolve, reject)=>{
        Answer.find({idLesson}, (err, answers)=>{
            if(err){
                reject(err);
            }else{
                resolve(answers);
            }
        });
    });
}

function joinLessonWithAnswers(lesson, answers){
    const lessonWithAns = {...lesson, respondents: []};
    answers.forEach((ans) =>{
        const ansObj = ans.toObject();
        ansObj.questions.forEach((quest)=>{
             if(lessonWithAns.respondents.indexOf(ansObj.idStudent) ===
-1){
                lessonWithAns.respondents.push(ansObj.idStudent);
            }
                                                const    isCorrect    =
checkAnswerToQuestion(quest.studentAnswers, quest.questionAnswers);
            if(isCorrect){
                                                                    if(!
lessonWithAns.questions[quest.index].correctStudents){
                    lessonWithAns.questions[quest.index].correctStuden
ts = [ansObj.idStudent]
                }else{
                    lessonWithAns.questions[quest.index].correctStuden
ts.push(ansObj.idStudent);
                }
            }else{
                                                                    if(!
lessonWithAns.questions[quest.index].incorrectStudents){
```

```
                    lessonWithAns.questions[quest.index].incorrectStud
ents = [ansObj.idStudent]
                }else{
                    lessonWithAns.questions[quest.index].incorrectStud
ents.push(ansObj.idStudent);
                }
            }
        });
    });
    return lessonWithAns;
}

function checkAnswerToQuestion(studentAns, questionAns){
                      return      studentAns.sort().join(',')      ===
questionAns.sort().join(',');
}

module.exports = getLessonReport;
const Answer = require('../../schemas/Answer');

async function getAnswersTotal (context){
    const numberOfAnswersTotal = await getAllAnswersCount();

    return numberOfAnswersTotal;
};

function getAllAnswersCount(){
    return new Promise((resolve, reject) =>{
        Answer.countDocuments({}, (err, count)=>{
            if(err){
                reject(err);
            }else{
                resolve(count);
            }
        });
    });
}
module.exports = getAnswersTotal;
```

Esquemas do Banco de dados

```
const mongoose = require('mongoose');
const crypto = require('crypto');

const User = new mongoose.Schema({
    email: String,
    name: String,
    hashed_password: String,
    salt: String,
    role: String,
    active: {
        type: Boolean,
        default: true,
    },
});

User.methods.hashPassword = function(password){
```

```
    this.salt = crypto.randomBytes(16).toString('hex');
          return  crypto.pbkdf2Sync(password,  this.salt,  1000,  64,
'sha512').toString('hex');
};

User.methods.validatePassword = function(password){
      const  hash  =  crypto.pbkdf2Sync(password,  this.salt,  1000,  64,
'sha512').toString('hex');
    return this.hashed_password === hash;
};
module.exports = mongoose.model('User', User);;
const mongoose = require('mongoose');

const Lesson = new mongoose.Schema({
    title: {
        type: String,
    },
    students: [
        mongoose.SchemaTypes.ObjectId
    ],
    questions: [
        {
            _id: false,
            type: {
                type: String,
                default: 'single',
            },
            title: {
                type: String,
            },
            options: [
                {
                    _id: false,
                    order: Number,
                    title: String,
                    selected: Boolean
                }
            ]

        }
    ],
});

module.exports = mongoose.model('Lesson', Lesson);
// const lesson = {
//     title: 'avaliacao 1',
//     id: 'aa7171a8',
//     students: ['idaw3','aiawi12', 'wiaw'],
//     questions: [
//         {
//             type: 'single',
//             title: 'O que é uma api? ',
//             alternatives :[
//                 {
//                     order:0,
//                     title: 'é uma interface de troca de mensagens',
//                     selected: true,
//                 },
```

```
//                 {
//                     order:1,
//                     title: 'não sei',
//                     selected: false,
//                 },
//                 {
//                     order:2,
//                     title: 'todas as anteriores',
//                     selected: false,
//                 },
//             ]
//         },
//         {
//             type: 'multiple',
//             title: 'quais são os paradigmas de programação',
//             alternatives:[
//                 {
//                     order:0,
//                     title: 'Imperativa',
//                     selected: true,
//                 },
//                 {
//                     order:1,
//                     title: 'funcional',
//                     selected: true,
//                 },
//                 {
//                     order:2,
//                     title: 'tipada',
//                     selected: false,
//                 },
//             ]
//         }
//     ]
// }
const mongoose = require('mongoose');

const Answer = new mongoose.Schema({
    idLesson: mongoose.SchemaTypes.ObjectId,
    idStudent: mongoose.SchemaTypes.ObjectId,
    questions:[
        {
            _id: false,
            index: Number,
            questionAnswers: {
                _id: false,
                type: [Number]
            },
            studentAnswers: {
                _id: false,
                type: [Number]
            },
            answered: Boolean
        }
    ]
});

module.exports = mongoose.model('Answer', Answer);
```

Front-end

Main

```
const express = require('express');
const session = require('express-session');
const MongoStore = require('connect-mongo')(session);
const app = express();
const mongoose = require('mongoose');
const settings = require('../settings');

app.set('views', settings.TEMPLATES_PATH); // specify the views
directory
app.set('view engine', 'pug');
app.use(express.static(settings.STATIC_FILES_PATH)); // sets the
folder that servers the static files
mongoose.connect(settings.DB_URL,{useNewUrlParser:true,
useUnifiedTopology:true});
app.set('trust proxy', 1);
app.use(session({
    secret: settings.SESSION_SECRET,
    resave: true,
    saveUninitialized: true,
    cookie: {secure:false}, // MAYBE WILL HAVE TO CHANGE ON PRODUCTION
    store: new MongoStore({
        mongooseConnection: mongoose.connection
    })
}));
//////////////////////                                          REACT
ROUTING ////////////////////////////////////////////
app.get('/professor/*',
    professorMiddleware,
            (req, res) => res.render('default', { scripts:
['/javascripts/professor-build.js']}));
app.get('*',
            (req, res) => res.render('default', { scripts:
['/javascripts/student-build.js']}));

// routes debbug
app.use((req,res,next)=>{
    console.log(req.method, req.url);
    next();
});
app.listen(settings.PORT, () => console.log(`Server Listening on port
${settings.PORT}`));


function professorMiddleware(req, res, next){
    // test if has role to acess professor page, other wise, redirects
do student page
     if(req.session && req.session.APRENDA && req.session.APRENDA.role
=== 'professor'){
        next();
    }else{
            res.render('default', { scripts: ['/javascripts/student-
build.js']});
    }
}
```

```
html
  head
    title= "Aprenda"
                link(rel="stylesheet",    href="/stylesheets/reset.css",
type="text/css")
         link(rel="stylesheet",  href="/stylesheets/bootstrap.min.css",
type="text/css")
                  link(rel="stylesheet",    href="/stylesheets/all.css",
type="text/css")
     meta(name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1, user-scalable=0")
  body
    div(id='root')
      if !scripts || scripts.length == 0
        h1 There are no scripts to load
      else
        each js in scripts
          script(src=js)
```

Componentes com a lógica

```
import React, { useState, useEffect } from 'react';
import { SERVER_URL } from '../../../../settings';

import ReportUser from '../presentational/ReportUser';

const ReportUserContainer = ({
    idUser
}) => {
    const [user, setUser] = useState(undefined);
    async function getUserReport() {
        const url = new URL(`${SERVER_URL}/getStudentReport`);
        url.search = new URLSearchParams({
            idStudent: idUser
        });
        const response = await fetch(url, {
            method: 'GET',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials: 'include',
        });
        const ans = await response.json();
        if (!ans.error) {
            setUser(ans);
        }
    }

    useEffect(()=>{
        getUserReport();
    }, [])
    if (!user) return <p>Loading...</p>
    return (
        <ReportUser user={user} />
    )
};
```

```
export default ReportUserContainer;
import React, { useEffect, useState } from 'react';
import { SERVER_URL } from '../../../../settings';

import ReportLesson from '../presentational/ReportLesson';

const ReportLessonContainer = ({
    idLesson,
}) => {
    const [lesson, setLesson] = useState(undefined);
    useEffect(() => {
        getLessonReport();
    }, []);

    async function getLessonReport() {
        const url = new URL(`${SERVER_URL}/getLessonReport`);
        url.search = new URLSearchParams({
            idLesson
        });
        const response = await fetch(url, {
            method: 'GET',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials: 'include',
        });
        const ans = await response.json();
        if(!ans.error){
            setLesson(ans);
        }
    };
    return (
        <ReportLesson lesson={lesson} />
    )
}

export default ReportLessonContainer;
import React, { useEffect, useState } from 'react';
import { useDispatch } from 'react-redux';
import  {  PieChart,  Pie,  Cell,Tooltip,  ResponsiveContainer  }  from
'recharts';
import { SERVER_URL } from '../../../../settings';

import { setUsers } from '../actions/users';

const COLORS = ['#0088FE', '#00C49F'];

const ReportAllStudents = ({ }) => {
    const dispatch = useDispatch();
    const [totalUsers, setTotalUsers] = useState([
        {
            name: 'Professores',
            value: 0,
        },
        {
            name: 'Alunos',
```

```
            value: 0,
        }
    ]);
    useEffect(() => {
        getAllStudentsReport();
    }, []);

    async function getAllStudentsReport() {
        const response = await fetch(`${SERVER_URL}/getUsers`, {
            method: 'GET',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials: 'include'
        });
        const ans = await response.json();
        if (!ans.error) {
                  const students = ans.filter((usr) => usr.role ===
'student');
                 const professors = ans.filter((usr) => usr.role ===
'professor');
            setTotalUsers([
                {
                    name: 'Professores',
                    value: professors.length
                },
                {
                    name: 'Alunos',
                    value: students.length
                }
            ]);
            dispatch(setUsers(ans));
        }
    }
    return (
        <ResponsiveContainer height='100%' width='100%'>
            <PieChart>
                <Pie
                    data={totalUsers}
                    dataKey="value"
                    nameKey="name"
                    outerRadius={50}
                    label
                    startAngle={45}
                    endAngle={405}
                >
                    {
                        totalUsers.map((user, index) => {
                        return (
                            <Cell
                                key={`cell-${index}`}
                                fill={COLORS[index % COLORS.length]}
                                />
                            )})
                    }
                </Pie>
                <Tooltip />
```

```
            </PieChart>
        </ResponsiveContainer>
    )
};

export default ReportAllStudents;
import  React, { useEffect, useState } from 'react';
import { useDispatch } from 'react-redux';
import { SERVER_URL } from '../../../../settings';
import   {   PieChart,   Pie,   Tooltip   ,   ResponsiveContainer}   from
'recharts';

import { setLessons } from '../../commons/actions/lessons';

const ReportAllLessons = ({}) =>{
    const [totalLessons, setTotalLessons] = useState(0);
    const dispatch = useDispatch();
    useEffect(()=>{
        getAllLessonsReport();
    },[]);

    async function getAllLessonsReport(){
        const response = await fetch(`${SERVER_URL}/getAllLessons`, {
            method: 'GET',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include'
        });
        const ans = await response.json();
        if(!ans.error){
            setTotalLessons(ans.length);
            dispatch(setLessons(ans));
        }
    }
    return (
        <ResponsiveContainer height='100%' width='100%'>
            <PieChart>
                <Pie
                          data={[{name: 'Total de atividades', value:
totalLessons}]}
                    dataKey="value"
                    nameKey="name"
                    cx="50%"
                    cy="50%"
                    outerRadius={50}
                    fill="#8884d8"
                    label
                />
                <Tooltip />
            </PieChart>
        </ResponsiveContainer>
    );
}

export default ReportAllLessons;
import React, { useState, useEffect } from 'react';
```

```
import { SERVER_URL } from '../../../../settings';
import { PieChart, Pie, Tooltip, ResponsiveContainer } from
'recharts';

const ReportAllAnswers = () => {
    const [totalAnswers, setTotalAnswers] = useState(0);
    useEffect(()=>{
        getAllAnswers();
    },[]);

    async function getAllAnswers(){
         const response = await fetch(`${SERVER_URL}/getAnswersTotal`,
{
            method: 'GET',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include'
        });
        const ans = await response.json();
        if(!ans.error){
            setTotalAnswers(ans);
        }
    }
    return (
        <ResponsiveContainer height='100%' width='100%'>
            <PieChart >
                <Pie
                            data={[{name: 'Total de respostas', value:
totalAnswers}]}
                    dataKey="value"
                    nameKey="name"
                    cx="50%"
                    cy="50%"
                    outerRadius={50}
                    fill="#FFBB28"
                    label
                />
                <Tooltip />
            </PieChart>
        </ResponsiveContainer>
    )
};

export default ReportAllAnswers;
import React from 'react';
import { connect } from 'react-redux';

import { addQuestion } from '../actions/lesson';

import QuestionContainer from './QuestionContainer';
import Button from '../../commons/presentational/Button';

const mapStateToProps = state =>({
    questions: state.lessonReducer.questions,
});
```

```
const mapDispatchToProps = dispatch =>({
    _addQuestion: () => dispatch(addQuestion()),
});

const QuestionsContainer = ({
    questions,
    _addQuestion
}) =>{
    return (
        <>
            {
                questions.map((question,index)=>{
                    return <QuestionContainer
                        key={index}
                        index={index}
                    />
                })
            }
            <Button
                label='Nova questão'
                onClick={_addQuestion}
            />
        </>
    )
};

export       default       connect(mapStateToProps,mapDispatchToProps)
(QuestionsContainer);
import React from 'react';
import { connect } from 'react-redux';

import { addOption } from '../actions/lesson';

import OptionContainer from './OptionContainer';

import Button from '../../commons/presentational/Button';

const mapStateToProps = state =>({
    questions: state.lessonReducer.questions
});

const mapDispatchToProps = dispatch =>({
    _addOption: questionIndex => dispatch(addOption(questionIndex))
});

const OptionsContainer = ({
    questionIndex,
    _addOption,
    questions
}) =>{
    const options = questions[questionIndex].options;
    return (
        <>
            {
                options.map((option, index)=>{
                    return <OptionContainer
                            key={index}
                            questionIndex={questionIndex}
```

```
                            index={index}
                        />
                })
            }
            <Button
                onClick={()=>_addOption(questionIndex)}
                customClass="addOptionBtn fas fa-plus-circle"
            />
        </>
    )
};
export       default       connect(mapStateToProps,mapDispatchToProps)
(OptionsContainer);
import React from 'react';
import { connect } from 'react-redux';

import     {     changeOptionTitle,     setOptionSelected     }     from
'../actions/lesson';

import Option from '../presentational/Option';

const mapStateToProps = state =>({
    questions: state.lessonReducer.questions,
});

const mapDispatchToProps = dispatch =>({
    _changeOptionTitle: data => dispatch(changeOptionTitle(data)),
    _setOptionSelected: data => dispatch(setOptionSelected(data)),
});

const OptionContainer = ({
    questionIndex,
    index,
    _changeOptionTitle,
    _setOptionSelected,
    questions,
}) =>{
    const question = questions[questionIndex];
    const option = question.options[index];
    function setTitle(title){
        _changeOptionTitle({
            questionIndex,
            index,
            title
        });
    }
    function onOptionSelected(){
        if(question.type === 'single'){
            question.options.map((option, optionIndex) => {
                if(index === optionIndex){
                    _setOptionSelected({
                        questionIndex,
                        index: optionIndex,
                        selected: true,
                    });
                }else{
                    _setOptionSelected({
                        questionIndex,
```

```
                        index: optionIndex,
                        selected: false,
                    });
                }
            })
        }else{
            _setOptionSelected({
                questionIndex,
                index,
                selected: !option.selected,
            })
        }
    }

    return(
        <Option
            setTitle={setTitle}
            questionType={question.type}
            setOptionSelected={onOptionSelected}
            option={option}
        />
    )
}
export       default       connect(mapStateToProps,mapDispatchToProps)
(OptionContainer);
import React, { useEffect, useState } from 'react';
import { connect } from 'react-redux';
import { SERVER_URL } from '../../../../settings';

import { setUsers } from '../actions/users';

import UsersList from '../presentational/UsersListInput';

const mapStateToProps = state => ({
    users: state.usersReducer.users
});

const mapDispatchToProps = dispatch =>({
    _setUsers: users => dispatch(setUsers(users))
});

const LessonStudentsContainer = ({
    idLesson,
    students,
    setStudents,
    users,
    _setUsers
}) =>{
       const  [selectedUsignedStudents,  setSelectedUnsignedStudents]  =
useState([]);
      const  [selectedEnrolledStudents,  setSelectedEnrolledStudents]  =
useState([]);
    useEffect(()=>{
        if(users.length === 0){
            fetchUsers();
        }
    },[]);
```

```
    async function fetchUsers(){
        const response = await fetch(`${SERVER_URL}/getUsers`,{
            method: 'GET',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include',
        });
        const ans = await response.json();
        if(!ans.error){
            _setUsers(ans);
        }
    }
    function onSelectUnsignedStudent(student, checked){
        if(checked){
             setSelectedUnsignedStudents([...selectedUsignedStudents,st
udent._id]);
        }else{
                                                const    studentIndex    =
selectedUsignedStudents.indexOf(student._id);
            setSelectedUnsignedStudents(
                [...selectedUsignedStudents.slice(0, studentIndex),
                      ...selectedUsignedStudents.slice(studentIndex +1,
selectedUsignedStudents.length)]
            );
        }
    }

    function onSelectEnrolledStudent(student,checked){
        if(checked){
             setSelectedEnrolledStudents([...selectedEnrolledStudents,s
tudent._id]);
        }else{
                                                const    studentIndex    =
selectedEnrolledStudents.indexOf(student._id);
            setSelectedEnrolledStudents(
                [...selectedEnrolledStudents.slice(0, studentIndex),
                     ...selectedEnrolledStudents.slice(studentIndex +1,
selectedEnrolledStudents.length)]
            );
        }
    };

    async function enrollStudents(event){
        const response = await fetch(`${SERVER_URL}/enrollStudents`,{
            method: 'PUT',
            body:JSON.stringify({
                idLesson,
                students: selectedUsignedStudents
            }),
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include',
        });
        const ans = await response.json();
```

```
        if(!ans.error){
            setStudents(ans);
            setSelectedUnsignedStudents([]);
        }
    };

    async function unenrollStudents(event){
         const response = await fetch(`${SERVER_URL}/unenrollStudents`,
{
            method: 'PUT',
            body:JSON.stringify({
                idLesson,
                students: selectedEnrolledStudents
            }),
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include',
        });
        const ans = await response.json();
        if(!ans.error){
            setStudents(ans);
            setSelectedEnrolledStudents([]);
        }
    };

    function idsToUsersInformations(ids){
        const usersInformations = ids.map(id=>{
            const userIndex = users.findIndex(user=> user._id === id);
            if(userIndex !== -1){
                return users[userIndex];
            }else{
                return id;
            }
        });
        return usersInformations;
    }
    return (
        <>
            <h3 style={{marginLeft:'10px'}}>Alunos Matriculados</h3>
            <UsersList
                users={idsToUsersInformations(students)}
                onSelect={onSelectEnrolledStudent}
                  unlockActionButton={selectedEnrolledStudents.length >
0}
                actionButtonLabel='Desmatricular'
                onAction={unenrollStudents}
            />

                            <h3  style={{marginLeft:'10px'}}>Alunos  não
matriculados</h3>
            <UsersList
                   users={users.filter(user=>students.indexOf(user._id)
=== -1)}
                onSelect={onSelectUnsignedStudent}
                   unlockActionButton={selectedUsignedStudents.length >
0}
```

```
                actionButtonLabel='Matricular'
                onAction={enrollStudents}
            />
        </>
    )
};

export     default     connect(mapStateToProps,     mapDispatchToProps)
(LessonStudentsContainer);
import React, { useEffect } from 'react';
import { connect } from 'react-redux';
import { SERVER_URL } from '../../../../settings';

//actions
import { setLessons } from '../../commons/actions/lessons';

//presentationals
import LessonsList from '../presentational/LessonsList';
import Button from '../../commons/presentational/Button';

const mapStateToProps = state => ({
    lessons: state.lessonsReducer.lessons,
});

const mapDispatchToProps = dispatch => ({
    _setLessons: lessons => dispatch(setLessons(lessons)),
});

async function getLessons(_setLessons) {
    const response = await fetch(`${SERVER_URL}/getAllLessons`, {
        method: 'GET',
        headers: {
            'Accept': 'application/json',
            'Content-Type': 'application/json'
        },
        credentials: 'include'
    });
    const ans = await response.json();
    if (!ans.error) {
        _setLessons(ans);
    }
};
function goToAddLesson(history) {
    history.push('/professor/lessons/add');
}

const LessonsContainer = ({
    lessons,
    _setLessons,
    history
}) => {
    useEffect(() => {
        getLessons(_setLessons);
    }, []);
    function onSelectLesson(idLesson) {
        history.push(`/professor/lessons/${idLesson}`);
    }
    return (
```

```
        <div style={{margin: '10px'}}>
                                    <Button   label='Nova'   onClick={()   =>
goToAddLesson(history)} />
            <LessonsList
                lessons={lessons}
                onSelectLesson={onSelectLesson}
            />
        </div>
    )
};

export     default     connect(mapStateToProps,     mapDispatchToProps)
(LessonsContainer);
import React, { useEffect } from 'react';
import { useHistory } from 'react-router-dom';
import { useDispatch, useSelector } from 'react-redux';
import { SERVER_URL } from '../../../../settings';

import { changeLessonTitle } from '../actions/lesson';
import { addNotification } from '../../commons/actions/notifications';

import LessonForm from '../presentational/LessonForm';

const LessonContainer = ({
    idLesson,
}) =>{
             const   {title,   questions}   =   useSelector(state   =>
state.lessonReducer);
    const dispatch = useDispatch();
    const history = useHistory();
    function onSave(){
        if(idLesson){
            editLesson();
        }else{
            addLesson();
        }
    }
    function _changeLessonTitle(title){
        dispatch(changeLessonTitle(title));
    }
    async function addLesson(){
        const response = await fetch(`${SERVER_URL}/addLesson`, {
            method: 'POST',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include',
            body: JSON.stringify({
                title,
                questions
            })
        });

        const ans = await response.json();
        if(!ans.error){
                    dispatch(addNotification('Atividade  adicionada  com
sucesso!'));
```

```
            history.push('/professor/lessons');
        }
    }
    async function editLesson(){
        const response = await fetch(`${SERVER_URL}/editLesson`, {
            method: 'PUT',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include',
            body: JSON.stringify({
                idLesson,
                title,
                questions
            })
        });
        const ans = await response.json();

        if(!ans.error){
            dispatch(addNotification('Atividade atualizada!'));
        }
    }
    return (
        <LessonForm
            title={title}
            setTitle={_changeLessonTitle}
            addLesson={onSave}
        />
    )
}

export default LessonContainer;
import React from 'react';
import { SERVER_URL } from '../../../../settings';

import Header from '../presentational/Header';

const HeaderContainer = ({
    history
}) =>{
    async function logout(){
        const response = await fetch(`${SERVER_URL}/logout`,{
            method: 'POST',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include',
        });
        const ans = await response.json();
        if(!ans.error){
            window.location.replace('/');
        }
    }
    return (
        <Header
            logout={logout}
```

```
        />
    )
};

export default HeaderContainer;
import React from 'react';
import { useSelector, useDispatch } from 'react-redux'

import { hideNotification } from '../actions/notifications';

import NotificationsList from '../presentational/Notifications';

const Notifications = () => {
    const dispatch = useDispatch();
                                  const        {notifications}        =
useSelector(state=>state.notificationsReducer);
    function hideBox(id){
        dispatch(hideNotification(id));
    }
    return (
        <NotificationsList
            notifications={notifications}
            hideBox={hideBox}
        />
    )
};

export default Notifications;
import React, { useEffect, useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { SERVER_URL } from '../../../../settings';

import { selectOption, setAnswers } from '../actions/lesson';
import { showModal, hideModal } from '../actions/modal';

import Question from '../presentional/Question';
import FeedbackQuestion from '../presentional/FeedbackQuestion';

const QuestionContainer = ({
    question,
    index,
    idLesson,
}) => {
         const   {   questions,   answers   }   =   useSelector(state   =>
state.lessonReducer);
          const   {   idUser:   idStudent   }   =   useSelector(state   =>
state.loginReducer);
    const dispatch = useDispatch();
    const [unlockActionBtn, setUnlockActionBtn] = useState(false);
     const selectedOptions = (questions.find(quest => quest.index ===
index) || {}).answer || [];
    const answerIndex = answers.findIndex(ans => ans.index === index);
    const disabled = answerIndex !== -1;

    useEffect(() => {
             const  ansOfThisQuestionIndex  =  answers.findIndex(ans  =>
ans.index === index);
        if (ansOfThisQuestionIndex !== -1) {
```

```
            selectAnswers(ansOfThisQuestionIndex);
        }
    }, [answers]);
    function onSelectOption(optionIndex) {
        if (question.type === 'single') {
            dispatch(selectOption(index, [optionIndex]));
        } else {
                                    const   indexOfOptionOnAnsArray   =
selectedOptions.indexOf(optionIndex);
            const existsOnAnsArray = indexOfOptionOnAnsArray !== -1;
            if (!existsOnAnsArray) {
                    dispatch(selectOption(index, [...selectedOptions,
optionIndex]));
            } else {
                dispatch(selectOption(index, [
                                            ...selectedOptions.slice(0,
indexOfOptionOnAnsArray),
                    ...selectedOptions.slice(indexOfOptionOnAnsArray +
1, selectedOptions.length),
                ]));
            }
        }
        if (!unlockActionBtn) {
            setUnlockActionBtn(true);
        }
    }
    async function checkAnswer() {
         const questionAnswer = questions.find(quest => quest.index ===
index);
                                        const    response    =    await
fetch(`${SERVER_URL}/checkLessonQuestionAnswer`, {
            method: 'POST',
            body: JSON.stringify({
                idLesson,
                idStudent,
                questionIndex: index,
                answer: (questionAnswer || {}).answer || []
            }),
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials: 'include',
        });
        const ans = await response.json();
        if (ans && !ans.error) {
                    const  ansOfThisQuestion  =  ans.find((question  =>
question.index === index));
                       const  {  studentAnswers,  questionAnswers  }  =
ansOfThisQuestion;
            dispatch(setAnswers(ans));
            setUnlockActionBtn(false);
            dispatch(
                showModal(
                    <FeedbackQuestion
                              isCorrect={compareAnswer(studentAnswers,
questionAnswers)}
                        closeModal={() => dispatch(hideModal())}
```

```
                    />
                )
            );
        }
    }
    function compareAnswer(userAns, realAns) {
        return userAns.sort().join(',') === realAns.sort().join(',');
    }
    function selectAnswers(idx) {
        dispatch(selectOption(index, answers[idx].studentAnswers));
    }
    return (
        <Question
            question={question}
            onSelectOption={onSelectOption}
            selectedOptions={selectedOptions}
            checkAnswer={checkAnswer}
            unlockActionBtn={unlockActionBtn}
            disabled={disabled}
                                questionAnswers={answers[answerIndex]  ?
answers[answerIndex].questionAnswers : []}
        />
    )
};

export default QuestionContainer;
import React from 'react';
import { useSelector } from 'react-redux';

import Modal from '../presentional/Modal';

const ModalContainer = ({

}) => {
          const  {  showing,  content  }  =  useSelector(state  =>
state.modalReducer);
    return (
        showing ?
        <Modal>
            {content}
        </Modal>
        :
        null
    )
};

export default ModalContainer;
import React, { useEffect } from 'react';
import { connect } from 'react-redux';
import { SERVER_URL } from '../../../../settings';

import { setLessons } from '../../commons/actions/lessons';

import LessonsList from '../presentional/LessonsList';

const mapStateToProps = state =>({
    idStudent: state.loginReducer.idUser,
    lessons: state.lessonsReducer.lessons,
```

```
});

const mapDispatchToProps = dispatch =>({
    _setLessons: lessons => dispatch(setLessons(lessons)),
});

const LessonsContainer = ({
    lessons,
    idStudent,
    _setLessons,
    history
}) =>{
    useEffect(()=>{
        if(lessons.length === 0){
            fetchUserLessons(idStudent);
        }
    },[]);
    async function fetchUserLessons(idStudent){
        const url = new URL(`${SERVER_URL}/getStudentLessons`);
        url.search = new URLSearchParams({
            idStudent
        });
        const response = await fetch(url,{
            method: 'GET',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include',
        });
        const ans = await response.json();
        if(!ans.error){
            _setLessons(ans);
        }
    }
    function goToLesson(idLesson){
        history.push(`/lessons/${idLesson}`);
    }
    return (
        <LessonsList
            lessons={lessons}
            goToLesson={goToLesson}
        />
    )
};

export     default     connect(mapStateToProps,     mapDispatchToProps)
(LessonsContainer);
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import { SERVER_URL } from '../../../../settings';

import { cleanLesson, setAnswers } from '../actions/lesson';

import Lesson from '../presentional/Lesson';

const mapStateToProps = state =>({
    lessons: state.lessonsReducer.lessons,
```

```
    idStudent: state.loginReducer.idUser,
});

const mapDispatchToProps = dispatch =>({
    _cleanLesson: () => dispatch(cleanLesson()),
    _setAnswers: answers => dispatch(setAnswers(answers)),
});

const LessonContainer = ({
    idLesson,
    lessons,
    _cleanLesson,
    _setAnswers,
    idStudent
})=>{
    const [lesson, setLesson] = useState({});

    async function getLesson(idLesson){
        let lesson = lessons.find(lesson=>lesson._id === idLesson);
        if(!lesson){
            lesson = await fetchLesson(idLesson);
        }
        setLesson(lesson);
    }

    async function fetchLesson(idLesson){
        const url = new URL(`${SERVER_URL}/getLesson`);
        url.search = new URLSearchParams({
            idLesson
        });
        const response = await fetch(url, {
            method: 'GET',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include',
        });
        const ans = await response.json();
        return ans;
    }

    async function fetchAnswer(){
        const url = new URL(`${SERVER_URL}/getStudentAnswer`);
        url.search = new URLSearchParams({
            idLesson,
            idStudent,
        });
        const response = await fetch(url, {
            method: 'GET',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include',
        });
        const ans = await response.json();
        if(ans){
```

```
            _setAnswers(ans);
        }
        return ans;
    }

    useEffect(()=>{
        getLesson(idLesson);
        fetchAnswer();
        return () =>{
            _cleanLesson();
        }
    },[]);
    return (
        <Lesson
            lesson={lesson}
        />
    )
}

export       default       connect(mapStateToProps,mapDispatchToProps)
(LessonContainer);
import React from 'react';
import { SERVER_URL } from '../../../../settings';

import Header from '../presentional/Header';

const HeaderContainer = () =>{
    async function logout(){
        const response = await fetch(`${SERVER_URL}/logout`,{
            method: 'POST',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            credentials:'include',
        });
        const ans = await response.json();
        if(!ans.error){
            window.location.reload();
        }
    }
    return (
        <Header
            logout={logout}
        />
    )
};

export default HeaderContainer;
```

Componentes visuais

```
import React, {useState} from 'react';
import styled from 'styled-components';


const Container = styled.div`
```

```
    position: absolute;
    right: 15px;
    top: 0;
    padding: 15px 20px;
`;
const MenuIcon = styled.div`
    cursor: pointer;
    div{
        width: 35px;
        height: 5px;
        background-color: #333;
        margin: 6px 0;
        transition: 0.4s;
    }
`;
const UlStyled = styled.ul`
    background-color: #FFFFFF;
    box-shadow: 0px 0px 20px rgba(0, 0, 0, 0.2);
    padding: 20px 15px;
    > * {
        cursor: pointer;
    }
`;


const SandwichMenu = ({children}) =>{
    const [showing, toggleShowing] = useState(false);
    return (
        <Container>
            <MenuIcon onClick={() => toggleShowing(!showing)}>
                <div></div>
                <div></div>
                <div></div>
            </MenuIcon>
            {
                showing &&
                <UlStyled>
                    {children}
                </UlStyled>
            }
        </Container>
    )
};

export default SandwichMenu;
import React, {useState} from 'react';

import Button from './Button';
import Input from './Input';


const LoginUsername = ({email, setEmail, onSubmit}) =>{
    return (
        <>
            <Input
                type='text'
                placeholder='login'
                onChange={(event)=>setEmail(event.target.value)}
```

```
                onKeyDown={(event)=>event.keyCode===13 && onSubmit()}
            />
                    <Button  onClick={onSubmit}  enable={email  !==  ''}
label='Próximo'/>
        </>
    )
};

export default LoginUsername
import React from 'react';

import Button from './Button';
import Input from './Input';

const LoginPasswordAndRole = ({password,onSubmit, setPassword}) =>{
    return (
        <>
            <Input
                type='password'
                placeholder='senha'
                onChange={(event)=>setPassword(event.target.value)}
                onKeyDown={(event)=>event.keyCode===13 && onSubmit()}
            />
                  <Button  onClick={onSubmit}  enable={password  !==  ''}
label='Entrar'/>
        </>
    )
};

export default LoginPasswordAndRole;

import React from 'react';
import styled from 'styled-components';

const StyledLayout = styled.div`
    display: flex;
    flex-direction: column;
`;

const Layout = ({children, header}) =>{
    return(
        <>
            {header}
            <StyledLayout>
                {children}
            </StyledLayout>
        </>
    )
}

export default Layout;
import React from 'react';
import styled from 'styled-components';

const CheckboxLabel = styled.label`
    display: block;
    position: relative;
    padding-left: 35px;
```

```
    margin-bottom: 12px;
    cursor: pointer;
    font-size: 22px;
    > input{
        position: absolute;
        opacity: 0;
        cursor: pointer;
        height: 0;
        width: 0;
        &:checked ~ .checkmark{
            background-color: #2196F3;
        }
    }
    > input:checked ~ .checkmark:after {
        display: block;
    }
    &:hover{
        > input ~ .checkmark{
            background-color: #ccc;
        }
    }
`;
const CheckMark = styled.span`
    position: absolute;
    top: 0;
    left: 0;
    width: 1rem;
    height: 1rem;
    background-color: #fff;
    border: #adb5bd solid 1px;
    border-radius: ${props=>props.type === 'radio' ? '50%' : 0};
    &:after{
        content: "";
        position: absolute;
        display: none;
        left: 4px;
        top: ${props=>props.type === 'radio' ? '4px' : '1px'};
        width: 5px;
        height: ${props=>props.type === 'radio' ? '5px' : '10px'};
        border: solid white;
         border-width: ${props=>props.type === 'radio' ? 'initial' :'0
3px 3px 0'};
        border-radius: ${props=>props.type === 'radio' ? '50%' : 0};
        transform: rotate(45deg);
    }
`;

const Input = ({
    type,
    label,
    placeholder,
    onChange,
    onKeyDown,
    checked,
    disabled = false,
    style,
    value,
    ...props
```

```
}) =>{
    let inputElement;
    switch (type) {
        case "text":
            inputElement =
            <div className="form-group" style={{...style}}>
                {
                    label &&
                                                                    <label
htmlFor="exampleInputEmail1">{label}</label>
                }
                <input
                    type="text"
                    className="form-control"
                    id="exampleInputEmail1"
                    placeholder={placeholder}
                    onChange={onChange}
                    onKeyDown={onKeyDown}
                    value={value}
                />
            </div>
            break;
        case 'password':
            inputElement =
            <div className="form-group">
                {
                    label &&
                    <label for="exampleInputPassword1">{label}</label>
                }
                <input
                    type="password"
                    className="form-control"
                    id="exampleInputPassword1"
                    placeholder={placeholder}
                    onChange={onChange}
                    onKeyDown={onKeyDown}
                />
            </div>
            break;
        case 'radio':
            inputElement =
            <div className="form-group" style={{...style}}>
                <div className="custom-control custom-radio">
                    <CheckboxLabel>
                        <input
                            type="radio"
                            className="custom-control-input"
                            checked={checked}
                            onChange={onChange}
                            disabled={disabled}
                        />
                        <CheckMark
                            className='checkmark'
                            type={'radio'}
                        />
                    </CheckboxLabel>
                </div>
            </div>
```

```
            break;
        case 'checkbox':
            inputElement =
            <div className="form-group" style={{...style}}>
                <div className="custom-control custom-checkbox">
                    <CheckboxLabel>
                        <input
                            type="checkbox"
                            className="custom-control-input"
                            checked={checked}
                            onChange={onChange}
                            disabled={disabled}
                        />
                        <CheckMark className='checkmark'/>
                    </CheckboxLabel>
                </div>
            </div>
            break;
        default: null
    }
    return (
        inputElement
    )
};

export default Input;
import React from 'react';

const Button = ({onClick, label, enable=true, customClass}) =>{
    const extraClass = enable ? '' : 'disabled'
    return (
        <button
            onClick={(event) => enable ? onClick(event) : {}}
             className={`btn btn-primary ${extraClass} ${customClass ?
customClass : ''}`.trim()}>
            {label}
        </button>
    )
};

export default Button;
import React from 'react';

import AlertBox from '../AlertBox';

import Container from './styles';

const Notifications = ({
    notifications,
    hideBox
}) =>{
    return (
        <Container>
            {
                notifications.map((notification)=>{
                    return (
                        <AlertBox
                            key={notification.id}
```

```
                            message={notification.message}
                            hideBox={()=>hideBox(notification.id)}
                            id={notification.id}
                        />
                    )
                })
            }
        </Container>
    )
};

export default Notifications;

import React, {useState} from 'react';
import LoginUsername from '../LoginUsername';
import LoginPasswordAndRole from '../LoginPasswordAndRole';

import { Container, Box, Title } from './styles';

const LoginBox = ({loginAttempt, loginStep, checkLogin})=>{
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    return (
        <Container>
            <ul className="circles">
                    <li></li>
                    <li></li>
                    <li></li>
                    <li></li>
                    <li></li>
                    <li></li>
                    <li></li>
                    <li></li>
                    <li></li>
                    <li></li>
            </ul>
            <Box>
                <Title>Bem vindo ao Apprenda</Title>
                {
                    loginStep === 'login' ?
                    <LoginUsername
                        setEmail={setEmail}
                        onSubmit={()=>checkLogin(email)}
                        email={email}
                    />
                    :
                    <LoginPasswordAndRole
                        setPassword={setPassword}
                        onSubmit={()=>loginAttempt({email, password})}
                        password={password}
                    />
                }
            </Box>
        </Container>
    )
};

export default LoginBox;
```

```
import React from 'react';

//styles
import { CardStyled } from './styles';

const cardImage = '/assets/card-image.jpg';
const Card = ({
    fontColor = 'white',
    title,
    onClick
})=>{
    return (
        <CardStyled fontColor={fontColor} onClick={onClick}>
                 <div  className="card_image"> <img  src={cardImage}  />
</div>
            <div className="card_title">
                <p>{title}</p>
            </div>
        </CardStyled>
    )
};

export default Card;
import React, { useEffect } from 'react';

import Box from './styles';

const AlertBox = ({
    message,
    hideBox,
    id
}) =>{
    useEffect(()=>{
        setTimeout(() => {
            hideBox(id);
        }, 4000);
    }, []);
    return (
        <Box className='alert alert-dismissible alert-primary'>
            <button
                type="button"
                className="close"
                data-dismiss="alert"
                onClick={hideBox}>&times;</button>
            <span>{message}</span>
        </Box>
    )
}
export default AlertBox;
import React from 'react';
import styled from 'styled-components';

import Input from '../../commons/presentational/Input';
import Button from '../../commons/presentational/Button';

const Ul = styled.ul`
```

```
    margin: 40px 10px;
`;

const UsersList = ({
    users,
    onSelect,
    unlockActionButton,
    actionButtonLabel,
    onAction
}) =>{
    return (
        <>
            <Ul className="list-group">
                {
                    users.length === 0 ?
                        <li className="list-group-item d-flex justify-
content-between align-items-center">
                        Nenhum usuário
                    </li>
                    :
                    users.map((user,index)=>{
                        return (
                             <li key={index} className="list-group-item
d-flex justify-content-between align-items-center">
                                {user.name}
                                <Input
                                    type='checkbox'
                                    index={index}
                                      onChange={(event)=>onSelect(user,
event.target.checked)}
                                />
                            </li>
                        )
                    })
                }
            </Ul>
            {
                unlockActionButton &&
                <Button
                    label={actionButtonLabel}
                    onClick={onAction}
                />
            }
        </>
    )
};

export default UsersList;
import React from 'react';
import {
    PieChart,
    Pie,
    ResponsiveContainer,
    Cell,
    Tooltip,
    Legend
} from 'recharts';
```

```
import { NoDataHolder } from './styles';

const CustomPieChart = ({
    data,
    colors = ['#0088FE', '#00C49F'],
    noDataText = 'Nenhum dado encontrado',
    icon = 'far fa-folder-open'
}) => {
     const valueTotal = data.reduce((acc,curr) => acc += curr.value,
0);
    console.log('value total', valueTotal);
    return (
        <>
            {
                valueTotal > 0 ?
                <div className="chartHolder">
                    <ResponsiveContainer height='100%' width='100%'>
                        <PieChart>
                            <Pie
                                data={data}
                                dataKey="value"
                                nameKey="name"
                                outerRadius={50}
                                label
                                startAngle={45}
                                endAngle={405}
                            >
                                {
                                    data.map((user, index) => {
                                    return (
                                        <Cell
                                            key={`cell-${index}`}
                                                   fill={colors[index %
colors.length]}
                                            />
                                        )})
                                }
                            </Pie>
                            <Tooltip />
                        </PieChart>
                    </ResponsiveContainer>
                </div>
                :
                <NoDataHolder>
                    <p>{noDataText}</p>
                    <i className={icon}></i>
                </NoDataHolder>
            }
        </>
    )
};

export default CustomPieChart;
import React from 'react';
import { Link } from 'react-router-dom';

import HeaderStyled from './styles';
```

```
import                          SandwichMenu                          from
'../../../commons/presentational/SandwichMenu';

const Header = ({
    logout
}) =>{
    return(
        <HeaderStyled>
                                        <Link   to={'/professor/lessons'}
style={{color:'#FFFFFF'}}>Atividades</Link>
                                        <Link   to={'/professor/reports'}
style={{color:'#FFFFFF'}}>Relatórios</Link>
            <SandwichMenu>
                <li onClick={()=>logout()}>SAIR</li>
            </SandwichMenu>
        </HeaderStyled>
    )
}

export default Header;
import React from 'react';

import Input from '../../../commons/presentational/Input';
import Button from '../../../commons/presentational/Button';
import QuestionsContainer from '../../containers/QuestionsContainer';

import Form from './styles';

const LessonForm = ({
    title,
    setTitle,
    addLesson,
}) =>{
    return (
        <Form>
            <Input
                type='text'
                placeholder='Título'
                onChange={(event) => setTitle(event.target.value)}
                value={title}
            />
            <QuestionsContainer/>
            <Button
                onClick={addLesson}
                label='Salvar lição'
                customClass='saveLessonBtn btn-success'
            />
        </Form>
    )
};

export default LessonForm;
import React from 'react';

import Container from './styles';

const LessonsList = ({
    lessons,
```

```
    onSelectLesson,
}) =>{
    return (
        <>
            <h5>Atividades</h5>
            <Container className="list-group">
                {
                    lessons.map((lesson, index) =>{
                        return(
                            <li
                                    className="list-group-item d-flex
justify-content-between align-items-center"
                                                                onClick={() =>
onSelectLesson(lesson._id)}
                                key={lesson._id}>
                                {lesson.title}
                            </li>
                        )
                    })
                }
            </Container>
        </>
    )
};

export default LessonsList;
export { default } from './MainReports';
import React from 'react';
import styled from 'styled-components';

import Form from './styles';

import Input from '../../../commons/presentational/Input';

const Option = ({
    setTitle,
    questionType,
    setOptionSelected,
    option,
    readOnly,
    customClass
}) =>{
    return (
        <Form className={customClass ? customClass : ''}>
            <Input
                        type={questionType === 'single' ? 'radio' :
'checkbox'}
                                onChange={ readOnly ? () => {} :
()=>setOptionSelected()}
                checked={option.selected}
                style={{
                    backgroundColor: 'rgb(33,150,243)',

                }}
            />
            {
                !readOnly ?
                <Input
```

```
                    type='text'
                    placeholder='opção'
                    onChange={(event)=>setTitle(event.target.value)}
                    value={option.title}
                    style={{
                        width: '100%',
                    }}
                />
                :
                     <p  className={option.selected  ?  'selected'  :
''}>{option.title}</p>
            }
        </Form>
    )
};

export default Option;
import React from 'react';


import Input from '../../../commons/presentational/Input';
import OptionsContainer from '../../containers/OptionsContainer';
import Option from '../Option';

import Form from './styles';


const Question = ({
    setTitle,
    setType,
    title,
    index,
    type,
    readOnly,
    options=[],
    customClass
}) =>{
    return(
        <Form className={customClass ? customClass: ''}>
            <div className='questionTypes'>
                {
                    !readOnly &&
                    <>
                        <div className='form-group'>
                            <Input
                                type='radio'
                                onChange={()=>setType('single')}
                                checked={type==='single'}
                            />
                            <p>Única escolha</p>
                        </div>
                        <div className='form-group'>
                            <Input
```

```
                        })
                    }
                </>
                :
                <>
                    <Input
                        type='text'
                        placeholder='Pergunta'
                         onChange={(event)=>setTitle(event.target.value
)}
                        value={title}
                    />
                    <OptionsContainer
                        questionIndex={index}
                    />
                </>
            }
        </Form>
    )
};

export default Question;
import React from 'react';

import Question from '../Question';
import CustomPieChart from '../CustomPieChart';

import Container from './styles';

const ReportLesson =  ({
    lesson
}) =>{
    if(!lesson) return <p>Loading....</p>
    const chartData = [
        {
            name: 'Alunos matriculados',
            value: lesson.students.length
        },
        {
            name: 'Alunos que responderam',
            value: lesson.respondents.length
        }
    ]
    return(
        <Container>
            <h1>{lesson.title}</h1>
            <CustomPieChart
                data={chartData}
                noDataText="Nenhum aluno matriculado"
                icon="fas fa-user-slash"
            />
            <>
                <h3 id="questionsHeader">Questões</h3>
                {lesson.questions.map((question, index)=>{
                    const colors = ['#90CC9C', '#D15E53', '#C5C3B4'];
                      const correctStudents = question.correctStudents
|| [];
```

```
import Input from '../../../commons/presentational/Input';

import {
    OptionContainer,
    P,
} from './styles';

const Option = ({
    option,
    type,
    onSelectOption,
    checked,
    disabled,
    isCorrect,
})=>{
    return(
        <OptionContainer>
            <Input
                type={type === 'single' ? 'radio' : 'checkbox'}
                onChange={onSelectOption}
                checked={checked}
                disabled={disabled}
                style={{
                    backgroundColor: '#2196F3',
                    margin: 0,
                    display: 'flex',
                    alignItems: 'center',
                    padding: '5px 0',
                }}
            />
            <P isCorrect={isCorrect}>{option.title}</P>
        </OptionContainer>

    )
};

export default Option;
import React from 'react';

import Option from '../Option';
import Button from '../../../commons/presentational/Button';

//styles
import {
    QuestionContainer,
    QuestionTitle,
} from './styles';

const Question = ({
    question,
    onSelectOption,
    selectedOptions,
    checkAnswer,
    unlockActionBtn,
    disabled,
    questionAnswers
})=>{
    return(
```

```
        <QuestionContainer>
            <QuestionTitle>{question.title}</QuestionTitle>
            {
                question.options.map((option,index)=>{
                    return (
                        <Option
                            key={index}
                              checked={selectedOptions.indexOf(index) !
== -1}
                            option={option}
                            type={question.type}
                            onSelectOption={()=>onSelectOption(index)}
                            disabled={disabled}
                              isCorrect={questionAnswers.length === 0 ?
undefined : questionAnswers.indexOf(index) !== -1}
                        />
                    )
                })
            }
            {
                unlockActionBtn &&
                <Button onClick={checkAnswer} label='enviar'/>
            }
        </QuestionContainer>
    )
};

export default Question;
```