INSTITUTO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DA PARAÍBA
DIRETORIA DE DESENVOLVIMENTO DE ENSINO
COORDENAÇÃO DO CURSO SUPERIOR DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

JOÃO PEDRO ALVES DE LIMA
MYRLLA LUCAS PEREIRA

CODE2KNOW — UM PROTÓTIPO DE SISTEMA DE ENSINO DE PROGRAMAÇÃO BASEADO EM JUIZ ONLINE

CAMPINA GRANDE - PB
2023

JOÃO PEDRO ALVES DE LIMA
MYRLLA LUCAS PEREIRA

CODE2KNOW — UM PROTÓTIPO DE SISTEMA DE ENSINO DE PROGRAMAÇÃO BASEADO EM JUIZ ONLINE

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de Computação, do Instituto Federal da Paraíba – *Campus* Campina Grande, em cumprimento às exigências parciais para a obtenção do título bacharel em Engenharia de Computação.

ORIENTADOR(A):
HENRIQUE DO NASCIMENTO CUNHA

CAMPINA GRANDE - PB
2023

L732c Lima, João Pedro Alves de

Code2know: um protótipo de sistema de ensino de programação baseado em Juiz online / João Pedro Alves de Lima, Myrlla Lucas Pereira. - Campina Grande, 2023.

20 f.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia de computação) - Instituto Federal da Paraíba, 2023.

Orientador: Prof. Me. Henrique do Nascimento Cunha.

1. Programação - ensino 2. Programação - juiz online 3. Casos de teste público I. Pereira, Myrlla Lucas. II.Cunha, Henrique do Nascimento. II Título.

CDU 004.438

JOÃO PEDRO ALVES DE LIMA
MYRLLA LUCAS PEREIRA

CODE2KNOW — UM PROTÓTIPO DE SISTEMA DE ENSINO DE PROGRAMAÇÃO BASEADO EM JUIZ ONLINE

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de Computação, do Instituto Federal da Paraíba – *Campus* Campina Grande, em cumprimento às exigências parciais para a obtenção do título bacharel em Engenharia de Computação.

Aprovada em ____ / ____ / _____

Banca Examinadora

_____
Prof(a). Henrique do Nascimento Cunha, DSc. - IFPB
Orientador (IFPB)

_____
Prof(a). Anderson Fabiano Batista Ferreira da Costa, DSc - IFPB
Examinador

_____
Prof(a). Ianna Maria Sodré Ferreira de Sousa, DSc. - IFPB
Examinador

Code2Know – Um protótipo de sistema de ensino de programação baseado em juiz online

João Pedro Alves de Lima[1], **Myrlla Lucas Pereira**[1]

[1]Departamento de Ensino Superior (DES)
Instituto Federal da Paraíba (IFPB) – Campina Grande, PB – Brazil

jpalves1101@gmail.com, myrllajucas20@gmail.com

Abstract. *The present work proposes the development of a prototype for an online judge system focused on programming education. With the aim of enhancing the aspects of these systems, such as usability, scalability, and clear feedback to students. The work seeks to fill gaps in current systems, which are often hybrids between competition and teaching. The proposed system aims to improve the learning environment, allowing students to solve exercises with public test cases and receive personalized feedback based on the exception obtained in the submission process. As a result, a proof of concept of the system was obtained, with the basic functionalities to be implemented in the classroom.*
Keywords: *public test cases; programming teaching; online judge; feedback.*

Resumo. *O presente trabalho propõe o desenvolvimento de protótipo um sistema baseado em juiz online com foco no ensino de programação. Com o objetivo de aperfeiçoar os aspectos desses sistemas, como usabilidade, escalabilidade e um feedback claro aos alunos. O trabalho busca preencher lacunas existentes nos sistemas atuais, que muitas vezes são híbridos entre competição e ensino. O sistema proposto visa melhorar o ambiente de aprendizado, permitindo que os alunos resolvam exercícios com casos de teste públicos e recebam o feedback personalizado de acordo com a exceção obtida no processo de submissão. Como resultado foi obtido uma prova de conceito do sistema, com as funcionalidades básicas para ser implantado em sala de aula.*
Palavras chave: *casos de teste públicos; ensino de programação; feedback; juiz online.*

1. Introdução

Os sistemas com base em juiz *online* inicialmente foram criados tendo como finalidade a sua utilização em competições [Santos and Ribeiro 2012], tais como maratonas de programação, para automatizar a correção de questões com base em um conjunto de casos de teste. Entretanto, com o passar do tempo foi percebido que esses sistemas poderiam ser aplicados no ensino de programação, visto que auxiliaria o professor de diversas formas, principalmente diminuindo a carga horária necessária para dar um *feedback* para que o aluno pudesse resolver algum problema. Nesta mesma linha, o professor seria auxiliado na criação de roteiros de exercícios, visto que questões elaboradas por outros professores e cadastradas na plataforma poderiam ser usadas.

Os sistemas atuais possuem algumas limitações, principalmente pelo fato de terem sido pensados parcial ou exclusivamente para finalidades de competições. Um dos problemas encontrados é a falta de maturidade dos sistemas quando se trata de

funcionalidades que podem ajudar o estudante iniciante no mundo da programação. [Francisco et al. 2018]

Existem sistemas que buscam estar aptos para o ensino de programação, como o beecrowd [beecrowd 2023] e o TheHuxley [The Huxley 2023], mas apesar disso, são poucos os que na sua concepção já buscavam o foco inicial na educação e essa falta de prioridade faz com que existam lacunas que possamos preencher. Além disso, podemos incorporar os pontos fortes desses sistemas consolidados, como o gerenciamento de diferentes turmas, acompanhamento do desempenho dos estudantes, importação de atividades, entre outras características, no nosso sistema tentando sempre melhorá-los. [Francisco et al. 2018]

Apesar de nos últimos anos os sistemas baseados em juiz *online* aplicados ao ensino de programação tenham se popularizado, muitos dos sistemas existentes são híbridos. Sendo assim, presume-se que um sistema focado no ensino pode conquistar espaço no mercado e, além de ajudar os estudantes e professores, posteriormente conectar empresas e alunos.

Além das contribuições citadas anteriormente, como feedback claro e implementação de casos de testes púlicos, é importante haver preocupação com a segurança. Ao avaliar códigos enviados, deve-se garantir que não exista código malicioso. O ideal é que ele seja executado em um ambiente controlado.

Existem também diversos tipos de funcionalidades que podem ser um diferencial, como a possibilidade do sistema funcionar localmente para sincronização posterior. Essas funcionalidades serão comentadas nas seções a frente e poderão ser implementadas futuramente.

Após a criação do sistema a sua validação pode ser feita nas novas turmas de Algoritmos e Lógica de programação, pois é a porta de entrada na programação para muitos dos alunos que estão ingressando em cursos da área. Dentre as melhorias idelaizadas, propõe-se a disponibilização dos casos de teste para que aluno tenha mais autonomia no momento da resolução de problemas, melhorando o *feedback* e diminuindo a interação aluno professor. Desse modo o professor pode se dedicar mais a parte do ensino de programação em si.

Em conformidade com o dito anteriormente, o presente trabalho se propõe a desenvolver um protótipo de sistema baseado em juiz *online* com o foco no ensino de programação, visando aperfeiçoar os aspectos desses sistemas que são focados em competições e por isso alguns fatores podem ser melhorados, para que assim seja possível integrar com outros sistemas de Ambiente Virtual de Aprendizagem (AVA) [Francisco et al. 2018]. Pretende-se melhorar o *feedback* dado ao aluno, pois apesar dele ser instantâneo as mensagens recebidas pelo sistema muitas vezes não são claras e acabam sendo genéricas e limitadas.

1.1. Objetivos

1.1.1. Objetivo geral

- Desenvolver um protótipo de sistema de apoio ao ensino de programação que permita a avaliação programas por meio de casos de teste públicos, com o foco

prioritário na aprendizagem, visando preencher algumas lacunas encontradas nos sistemas usados como base.

1.1.2. Objetivos específicos

- Manter casos de testes públicos, visto que o sistema será focado na aprendizagem.
- Com base na exceções mais comuns, prover ao estudante, mensagens relacionadas a cada exceção.
- Criação de uma documentação rica para uma melhor manutenção do sistema e também poder proporcionar a escalabilidade.

1.2. Justificativa

O processo de aprendizagem com a linguagem de programação pode ser desafiador. Por se tratar de um processo difícil e complexo, principalmente quando se está iniciando na área, isso acaba exigindo dos alunos maior dedicação e esforço para adquirir o raciocínio lógico necessário para a resolução dos problemas.

Independente da modalidade de ensino, sendo à distância ou presencial, disciplinas de programação exigem forte participação do aluno nas aulas, principalmente na realização de exercícios e desafios elaborados pelo professor que permite a melhor fixação do conteúdo.

É fato que o professor precisa muitas vezes elaborar execícios e, após a resolução realizada pelos alunos, ainda precisa corrigí-los: um processo moroso. Não raras às vezes, no decorrer da realização dos exercícios acabam surgindo dúvidas por parte dos alunos, e é habitual que eles recorram ao professor para saná-las. Assim, devido a alta carga de trabalho, o professor muitas vezes acaba não conseguindo sanar todas dúvidas, o que pode resultar em desmotivação e frustração por parte dos estudantes.

Considerando esses pontos, ter um sistema de apoio ao ensino de programação pode ajudar a melhorar o ambiente de aprendizado, onde o professor poderá adicionar exercícios com casos de testes públicos, e os alunos poderão resolvê-los direto da plataforma. Isso ajuda a diminuir a carga de trabalho por parte do professor e os alunos podem contar com o *feedback* fornecido pelo sistema, com mensagens que indiquem se ocorreu algum erro e o seu tipo (de natureza sintática ou de lógica).

2. Referencial Teórico

Hodiernamente percebe-se o empenho em automatizar os mais diversos setores da nossa sociedade, o setor no qual foi visto com mais clareza essa automação sendo o ambiente industrial, com o foco em aumentar a produção e minimizar as falhas humanas durante o processo.

Nos últimos anos com a popularização de competições de programação nas quais necessitavam de uma correção veloz dos exercícios propostos, e um *feedback* rápido, para que os alunos usassem o seu tempo com foco integral na resolução dos exercícios. Com a ideia central de aumentar a produção e minimizar as falhas, tanto do aluno quanto do professor, esses sistemas de correção automática passaram a ser incorporados nas salas de aula, mas com seu foco primário sendo em competições, faz com que apesar de cumprir

o seu papel, algumas vezes o *feedback* não é amigável para os iniciantes no contexto da programação. Apesar da adoção desses sistemas em sala de aula, ainda existem limitações que ao serem sanadas trarão benefício tanto para o professor quanto para o aluno. Ter um sistema focado primeiramente no ensino seria um diferencial para poder mitigar os problemas presentes nos produtos já estabelecidos no mercado.

Existem diversos sistemas no mercado que utilizam o conceito de Juiz *Online*, porém é perceptível dois nichos específicos. O primeiro diz respeito ao uso de sistemas com foco em competição, já o segundo diz respeito aos sistemas com foco em ensino. Também existe uma área cinza entre esses dois nichos, uma espécie de sistema híbrido, que busca se inserir, inserção muitas vezes ocorrida pela migração dos sistemas focados em competição, criando funcionalidades para suprir as necessidades encontradas por esse outro público.

Existem diversos sistemas no mercado que utilizam o conceito de Juiz *Online*, porém é perceptível dois nichos específicos. O primeiro diz respeito ao uso de sistemas com foco em competição, já o segundo diz respeito aos sistemas com foco em ensino. Também existe uma área cinza entre esses dois nichos, uma espécie de sistema híbrido, que busca criar funcionalidades para suprir as necessidades encontradas por um público distinto, se inserindo também no outro nicho. Essa inserção ocorre muitas vezes pela migração dos sistemas focados em competição.

Os sistemas que possuem ferramentas com suporte ao ensino, acabam não disponibilizando os casos de testes para os usuários ou mensagens de *feedback* mais claras. Na maior parte das vezes, é mostrado para o usuário, a mensagem obtida no terminal em que o caso obteve a exceção. Isso acaba sendo um problema, pois não é comum um usuário iniciante ter familiariadade com as mensagens de exceções.

A seguir serão listados os sistemas com base no seu foco e as suas principais caracteristicas.

2.1. Sistemas com foco em competição

O Hacker Rank é um sistema que se dispõe a ensinar programação através de desafios. Possui ferramentas de estímulo baseada em ranking, incentivando a competição entre os usuários. Não possui ferramentas de suporte para atividades de um professor, logo não é possível fazer lista de exercícios, verificar similaridades ou acompanhar o desempenho dos alunos [HackerRank 2023].

CodeChef é um sistema onde os programadores podem aprender e competir entre si, estimulando a disputa entre os usuários através de rankings. Apesar de possuir um sistema de suporte para universidades, o qual possui um *"Basic Plan"* e *"Plus Plan"*, nele a base de aprendizado é através de competições, práticas e discussões em fóruns [CodeChef 2023].

2.2. Sistemas com foco em ensino

Desenvolvido pela Universidade Regional Integrada de Erechim e lançado em 2012 o Beecrowd (antigo *URI online Judge*), é uma ferramenta baseada em *online judge* feita para auxiliar e melhorar o aprendizado na disciplina de programação. Permite interação entre os usuários, dispõe de várias linguagens, banco de questões divididas em

categorias e permite correção em tempo real. A ferramenta inicialmente não possuia um suporte para atividades de um professor, porém em 2013 foi criado um novo módulo denominado *Academic*, tendo um acesso diferenciado para professores, possibilitando a criação de disciplinas e de listas de exercícios. [beecrowd 2023]

O The Huxley é uma ferramenta com suporte para atividades acadêmicas, no qual o professor pode criar turmas, roteiros e acompanhar o desempenho dos estudantes. O The Huxley [The Huxley 2023] permite a submissão de código em diversas linguagens de programação, onde os alunos recebem o *feedback* de correção automática pelo sistema através de análise sintática do código e testes de aceitação.

2.3. Frontend

JavaScript

JavaScript é uma linguagem de programação lançada em 1995 no NETSCAPE 2.0 com o nome de LiveScript. Sendo a principal linguagem usada para criação de scripts em páginas web, tornando-se a linguagem de programação mais onipresente da história [Flanagan 2011]. Considerada uma linguagem multiparadigma é uma linguagem interpretada, leve, orientada a objetos com tipagem dinâmica fraca.

TypeScript

Typescript, desenvolvido pela Microsoft, é superconjunto de *javascript* que contém uma sintaxe adicional semelhante à orientação a objetos, oferecendo melhores ferramentas em qualquer escala [Maharry 2013]. Ele adiciona uma sintaxe ao *javascript* fornecendo uma melhor documentação permitindo que o *Typescript* possa validar se o código está funcionando corretamente, o que ajuda a detectar erros do início. Seu código é convertido em *JavaScript* e é executado em qualquer lugar que o *javascript* pode ser executado. O *TypeScript* entende *JavaScript* e usa inferência de tipo para fornecer ótimas ferramentas sem código adicional.

React

React [React 2022], criado pelo Facebook, é uma biblioteca *open-source* do *Javascript* usada para construir interface de usuário. Tem como objetivo facilitar a conexão entre diferentes partes de uma página. Baseado em componentes, auxilia o reaproveitamento de código, padronização de interface e facilita a manutenção. Isso faz com que o React se torne uma tecnologia flexível para solução de problemas e interfaces reutilizáveis, já que esses componentes podem ser manipulados de maneira distinta. O React pode ser renderizado no servidor usando o Node.js e também em aplicações móveis usando o React Native.

Ant Design

O Ant Design [Ant Design 2022] é uma biblioteca React que contém um conjunto de componentes para construir interfaces de usuários ricas e interativas. Escrito em *Typescript*, tem um pacote completo de design e ferramentas de desenvolvimento e personalização poderosa do tema em cada detalhe, por conta dessas vantagens, foi a biblioteca escolhida para uso de componentes.

2.4. Backend

Java

O Java [AWS 2022] é uma linguagem de programação criada pela *Sun MicroSystems* em 1995, que desde então evoluiu bastante e se tornou bastante popular ficando na 5ª posição no ranking de linguagens mais populares do *Stack Overflow* [STACK OVERFLOW 2021] e alguns dos fatores que fazem o Java ser popular nos dias de hoje é o fato de ser gratuito, ser uma plataforma de desenvolvimento confiável, ter uma documentação detalhada e principalmente ser uma plataforma independente podendo ser executado em diversas plataformas sem a necessidade de ser reescrito. É uma linguagem orientada a objetos, sendo conhecida por ser bastantes versátil por ser multiplataforma podendo ser usada tanto em desenvolvimento *mobile* como em desenvolvimento de aplicações *desktop*, podendo também ser usada no desenvolvimento para a web tanto na criação de API (*Aplicaton Public Interface*) quanto na criação de sites com estratégia de SSR (*Server Side Rendering*).

Spring Boot

O Spring Boot [SPRING 2022] facilita a criação de aplicativos *stand-alone* que são baseados no *Spring Framework*, fazendo com que você possa simplesmente executar o arquivo gerado pela *build*. Durante a utilização do *Spring* não muito tempo atrás, eram necessárias diversas configurações e ao adicionar o *Spring Boot* diversas configurações padrões são feitas, mas caso o arquivo de configuração seja encontrado ele levará em consideração as configurações locais, o fato do *Spring* ser executado sem problemas após o *build* se dá porque existem um servidor embarcado nele, também poupando dessas configurações e entregando bastante agilidade no processo de desenvolvimento.

MariaDB

MariaDB Server [MARIADB 2022] é um dos servidores de bancos de dados mais populares do mundo, dentro os usuários estão, Wikipedia, Nokia, Samsung e Google, além disso foi é *open-source*, gratuito e foi criado pelos desenvolvedores originais do *MySQL* e a escolha por esse banco de dados se deu por:

- **Compatibilidade total com o *MySQL***
  Pelo MySQL já ser bem estabelecido no mercado, também é possível usar a base de conhecimento presente na ferramenta durante a implementação do projeto, e caso seja preciso alterar o banco de dados no futuro para uma melhor escalabilidade, o *MySQL* possui compatibilidade e facilita a migração dos dados.
- **Atualizações constantes e estabilidade**
  Por ser um sistema de código aberto ele recebe atualizações constantes sempre buscando manter a estabilidade, como também adicionando novas features que são julgadas como prioritárias.
- **Gratuito e *Open-Source***
  Característica importante pois não há necessidade de licença e bugs podem ser reportados caso aconteçam e acompanhados de perto o andamento para a correção, ou até mesmo tentar modificar para usar temporariamente caso seja necessário.

Bcrypt

O Bcrypt [Sriramya and Karthika 2015] é um algoritmo criptográfico desenvolvido por Niels Provos e David Mazières em 1999 e tem como objetivo fazer o *hashing*

de uma mensagem, porém esse algoritmo se diferencia por ser um meio para atingir uma forma segura de armazenar as senhas, devido ao custo computacional variável necessário para criptografar a senha. O algoritmo tem como base o cifrador de bloco de chave simétrica *Blowfish* criado por Bruce Schneier em 1993.

O uso de Bcrypt adiciona segurança a ataques ataques de força bruta e em conjunto com o Bcrypt podemos adicionar um salt no momento da geração do *hash*, fazendo também com que o sistema seja resistente a ataques de *rainbow tables*.

Swagger

O Swagger [SWAGGER 2022] consiste em uma mistura de ferramentas de código aberto, gratuitas e comercialmente disponíveis que permitem que qualquer pessoa crie APIs. E ele se iniciou como uma especificação de código aberto simples para projetar APIs RESTful em 2010 e a partir daí ferramentas de código aberto como Swagger UI, Swagger Editor e Swagger Codegen foram desenvolvidas para melhor implementar e visualizar as APIs definidas na especificação. Portanto como pretende-se que o sistema seja escalável e de simples entendimento, para isso é necessário documentar a API para seja menor a barreira enfrentada ao tentar integrar o sistema em uma solução já existente.

DBeaver

O *DBeaver* [DBeaver 2023] é uma ferrmanta universal de banco de dados de código aberto que tem como objetivo principal a usabilidade, suporta qualquer banco de dados que tenha o *driver* JDBC [Oracle 2023], além disso por ser baseado em tecnologias opensource, permite a a adição e criação de diversos plugins não se limitando apenas a conexão com o *driver* citado anteriormente.

2.5. Ferramentas

Whimsical

O Whimsical [WHIMSICAL 2022] é uma plataforma com suporte a uso colaborativo em tempo real, que possui a licença gratuíta e também possui diversas possibilidades na criação de diagramas como: diagramas de fluxo, wireframes, mapas mentais, entre outros tipos de diagramas.

brModelo

O brModelo é uma ferramenta gratuita e de código aberto com foco no ensino de modelagem de banco de dados relacionais, e tem como base a metodologia defendida por Carlos A. Heuser [Candido and dos Santos Mello 2017].

Notion

O Notion é uma ferramenta que suporta o uso colaborativo, e que tem como principio ser um *all-in-one-workspace* com a possibilidade de que num mesmo arquivo seja possível adicionar componentes que normalmente estão presentes em arquivos de tipos diferentes, como por exemplo um mesmo arquivo com uma lista de tarefas, imagens, gráficos, bases de dados para diferentes formas de visualizações entre diversas outras possibilidades. Vale lembrar também que é possivel aninhar um arquivo dentro de outro, fazendo com que seja simples caso seja preciso organizar os itens para uma melhor visualização do time [NOTION 2022].

Git

Git [Git 2022] é um sistema *open-source* de controle de versão, criado por Linus Torvalds, projetado para lidar desde projetos pequenos a muito grandes. O Git registra alterações feitas no projeto e através dele diversas pessoas podem contribuir simultaneamente, isso porque ele permite a existência de várias ramificações locais que podem ser totalmente independentes umas das outras, possibilitando a edição e criação de novos arquivos sem risco de que suas alterações sejam sobrescritas.

Github

Github [Github 2023] é um serviço baseado em nuvem, uma plataforma que possibilita criar um ambiente de colaboração entre desenvolvedores, usada também para gerenciar código utilizando o Git como sistema de controle. Ele permite que desenvolvedores façam mudanças em projetos compartilhados mantendo um registro detalhado do seu progresso.

Figma

Lançado em setembro de 2016, o Figma é uma ferramenta vetorial focada no desenvolvimento de design de telas. Um quesito importante do figma é que se trata de uma ferramenta colaborativa na nuvem, permitindo do desenvolvimento colaborativo em tempo real com outros usuários do seu time remotamente [Figma 2023].

Docker

Lançado em 2013, desenvolvido com a necessidade de melhoria na grande demanda de máquinas virtuais, o Docker é uma plataforma *open-source* que facilita a criação e administração de ambientes isolados [Docker 2023].

3. Arquitetura

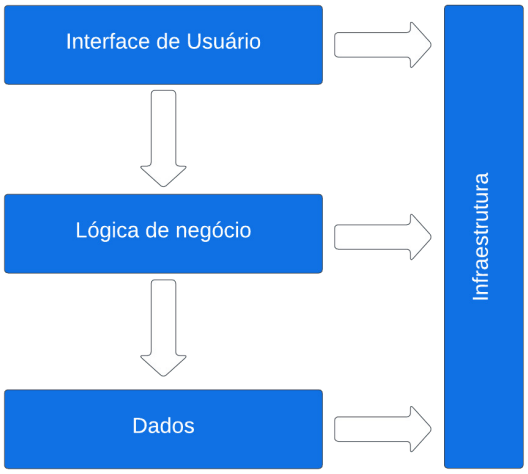

Figura 1. Arquitetura do sistema
Fonte: autoria própria

A arquitetura de código escolhida foi o modelo de Arquitetura em camadas [Valente 2020], é possível visualizar a estrutura dessa arquitetura através da figura 1.

Foi optado esse uso pois existia uma preferencia a um código um pouco mais desacoplado, porém não era desejável a rigorosidade de usar uma arquitetura como a *Clean Architecture* proposta por Robert Cecil Martin, apesar disso foi utilizado alguns conceitos presentes em "*Clean Architecture: A Craftsman's Guide to Software Structure and Design*"[Martin 2017], que seriam os seguintes:

- Independência da interace do usuário: A interface pode ser mudada de forma facil, sem que sejam necessárias alterações no restante do sistema. Sendo assim, a interface web, pode ser substituida por uma interface *mobile* caso seja necessário.
- Independência de banco de dados: Através desse princípio é definido que deve ser simples trocar de banco de dados, pois a sua a camada responsável por lidar com o banco deve ser completamente independente, ou seja, é possível alterar facilmente alterar o banco de dados sem afetar em momento algum as regras de negócio do nosso sistema.

4. Métodos

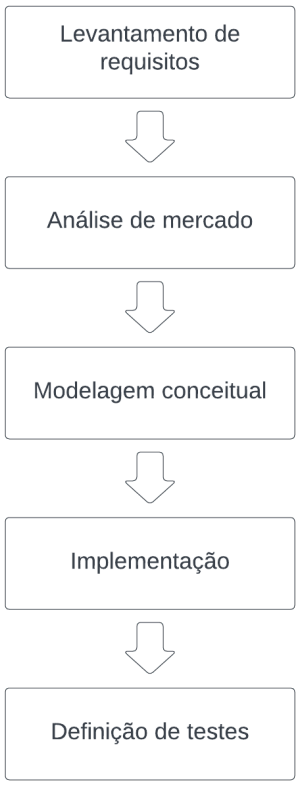

Figura 2. Fluxograma de desenvolvimento
Fonte: autoria própria

Para projetar o sistema, primeiro foi realizado o levantamento de requisitos junto com uma análise de mercado. Isso para que se possa entender os pontos fracos dos sistemas concorrentes e, a partir disso, criar as nossas potencialidades. Em seguida realizou-se a modelagem em nível conceitual e lógico, e então a criação dos diagramas do banco de dados. Posteriormente seguimos com a implementação do *backend* do sistema e a

definição de quais testes seriam usados durante o desenvolvimento da aplicação. Todo esse processo se deu de forma iterativa e incremental.

Para a etapa de desenvolvimento do sistema, inicialmente foi feita a criação das partes cruciais, como o sistema de submissão. Após isso, foi gerado o esquema do modelo físico do banco de dados e o *mock* para o frontend. O desenvolvimento foi seguido inicialmente com o *backend*, no qual primeiramente foi implementado as operações fundamentais de acordo com a lógica de negócio. O processo seguinte foi o desenvolvimento do *frontend*, focando nas telas de integração que não possuiam algum pré-requisito para para existir no sistema, e após cada semana, eram selecionadas novas telas que estavam desempedidas devido ao desenvolvimento realizado na semana anteior para serem incorporada ao projeto. Em paralelo a isso, foram feitas alterações no *backend* de acordo com as necessidades do *frontend*. O modelo com as atividades realizadas é mostrado na figura 2.

No processo de desenvolvimento, os testes foram feitos apenas pelo professor orientado deste projeto que também é professor da disciplina de Algoritmos e Programação. Sendo assim, após a aprovação dos testes feitos por ele, os professores que lecionam as cadeiras de programação no Instituto Federal da Paraíba, campus Campina Grande, poderão realizar testes de usabilidade.

Na fase de implantação inicialmente será realizado um experimento com uma turma de Algoritmos e Programação, no qual será disponibilizado um questionário para buscar um *feedback*. Posteriormente, com as mudanças realizadas no sistema levando em conta esse retorno, ele poderá ser implementado com as demais turmas de Algoritmos.

Frontend

Para construir o *frontend*, foi usado como editor o Visual Studio Code, versão 1.78.2, devido ao controle de versionamento do *Git* incorporado, juntamente com suporte para depuração, *autocomplete* inteligente e refatoração de código.

Por sua vez, a linguagem usada para o desenvolvimento web é majoritariamente o *JavaScript*, por isso juntamente com o *TypeScript*, foi escolhida para o projeto. O *TypesScript*, além de ter uma ótima documentação, oferece uma sintaxe adicional ao *JavaScript*, sendo capaz, através de sua tipagem e inferência de tipos, detectar erros no inicio do desenvolvimento do sistema.

Como biblioteca principal na criação da interface, foi escolhida a utilização do React, versão 18.2, que tem como objetivo facilitar a conexão entre diferentes partes de uma página, baseado em componentes, auxilia o reaproveitamento de código e padronização da interface, além da facilidade de manutenção durante o projeto. *Ant Design*, versão 5.3.2, foi escolhido devido a sua biblioteca de componentes de fácil uso e bastante rica, com diversos componentes que podem ser reaproveitados durante a construção do projeto.

Por fim, mas não menos importante, usamos o firebase, que apresenta vários serviços disponíveis para web. Neste foi utilizado apenas o serviço de armazenamento em nuvem, onde optamos por utilizar essa tecnologia para armazenar as imagens de perfil dos usuários e não salvar arquivos brutos no banco de dados.

Backend

A IDE (*Integrated development environment*) escolhida para a implementação do sistema foi o IntelliJ, versão 2022.3.2. Seguindo com o desenvolvimento *backend*, optou-se por utilizar o Java, versão 17, para criar uma API REST em conjunto com com o Spring Boot como framework para agilizar a criação da API, pois ele é de fácil uso e configuração para o desenvolvimento da aplicação, além de possuir uma vasta documentação e comunidade ativa.

Para realizar a persistência de dados no sistema, foi escolhido o MariaDB, pela compatibilidade com o *MySQL* e por ser um SGBD (Sistema Gerenciador de Banco de Dados) bem difundido no mercado e código aberto. O *DBeaver*, versão 23.0.4, foi usado para facilitar a visualização dos valores persistidos no banco de dados.

Pensando numa maior segurança contra ataques de *rainbow tables*, fizemos a escolha de usar o Bcrypt. Utilizamos também a biblioteca Lombok que provê anotações para evitar a escrita manual dos métodos *getter* e *setter*, provendo também anotações para método *equals*, *toString*, *hashCode*, além de construtores.

Para a documentação da API REST utilizamos o Swagger, devido a nossa vontade de construir um projeto com uma documentação rica, principalmente referente ao *backend*, parte na qual pretendemos que possa ser integrada com ambientes de aprendizagem, facilitando com que os desenvolvedores desses ambientes sintam-se familiarizados com nosso projeto.

Para a criação do subsistema utilizamos Python 3.11 em conjunto com o Flask framework na sua versão 2.2, pois precisávamos de um endpoint para submeter o código resposta e avaliar o caso de teste. A escolha da linguagem Python foi feita pois a sua utilização é presente na disciplina de Algoritmos e Programação, do curso de Engenharia de Computação do IFPB, que é a disciplina onde os alunos tem o primeiro contato com a programação. Decidimos inicialmente focar na submissão dessa linguagem para podermos aplicar as especificidades do nosso projeto nesse subsistema.

5. Resultados

Uso do sistema

Antes de iniciar a codificação do sistema, foi definido todo o escopo que deve abrangido, as funcionalidades definidas como necessárias para que o Prova de conceito (*POC*) possa funcionar. Partindo dessas funcionalidades foram definidas as entidades que deveriam existir para que o sistema pudesse funcionar corretamente. São elas: usuário, turma, tarefa, problema, instituição, casos de teste, resposta caso de teste e submissão. A entidade usuário possui quatro tipos, cada tipo possui determinados privilégios no sistema, tendo um fluxo de navegação diferente dos demais podendo acessar telas e funcionalidades distintas, como mostrado na Tabela 1.

O fluxo de acesso ao sistema pode ser dividido em duas partes: a navegação pelo sistema quando o usuário está logado e quando não está. Na primeira parte existe o acesso a todas as telas, de acordo com o perfil do usuário, já na segunda o acesso é composto por 6 telas. Dentre essas, está contida a listagem de problemas, onde o usuário poderá visualizá-los e ter uma ideia das atividades presentes no sistema, mas não sendo possível enviar a submissão.

	Administrador	Professor	Monitor	Aluno
Gerenciamento de permissões	X			
Adição/Edição/Remoção de Turma		X		
Adição/Remoção de Tarefa		X		
Edição de Tarefa		X	X	
Adição/Edição/Remoção de Problema		X		
Adição/Edição/Remoção de Casos de Teste		X		
Problema	X	X	X	X
Tarefa	X	X	X	X
Turma	X	X	X	X
Perfil/Gerenciamento de Conta	X	X	X	X

Tabela 1. Tabela de permissões de usuário.
Fonte: autoria própria

É tida como primeira página do sistema a *Landing Page* se o usuário não estiver logado, caso contrário temos a *Home*. Como dito anteriormente, dentre as 6 telas disponíveis para o acesso, temos a tela de Contato. Nessa tela, o usuário poderá entrar em contato com o suporte, podendo sugerir melhorias ou até solicitar acesso como administrador de uma determinada instituição. Nesse segundo caso, haverá uma troca de emails para que a permissão seja concedida.

Como mencionado, as telas que o usuário tem disponível varia de acordo com o seu perfil. Em geral, quando o usuário está logado, ele tem acesso a tela de listagem de turmas em que está alocado. Dentro da turma encontra-se disponível a lista de membros cadastrados e roteiros/provas referentes a mesma. Outro acesso presente, é a listagem de tarefas em geral, sem separação por turma. Nesta podem ser encontrados todos os roteiros e provas que estão alocados para o usuário.

Por fim, existe o perfil, onde o usuário pode ver as informações das submissões feitas e também gerenciar sua conta.

O perfil de professor traz funcionalidades como a criação/edição/remoção de tarefas, que podem ser um roteiro ou uma prova. Também criação/edição/remoção de turmas, podendo adicionar professores e disponibilizar uma chave de acesso para os alunos. Essa chave disponibilizada serve para que os alunos possam inseri-la em um campo especifíco e assim serem adicionados na turma. Além disso o professor também pode realizar a criação/edição/remoção de problemas. Vale ressaltar que a ação de remover turmas, tarefas e problemas só pode ser feita pelo professor que os criou.

As contas com privilégio de administrador possuem acesso a uma tela de gerenciamento de permissões, onde o usuário que possui esse cargo, pode selecionar os demais usuários para atribuir permissões de administrador ou de professor para os mesmos.

Como declarado nos objetivos do presente trabalho, optou-se por manter os casos de testes públicos e mensagens de *feedback* claras, como mostrado na Tabela 2, visto que o sistema é focado em aprendizagem. Assim, tendo em mãos o caso em que ocorreu o problema e a sua entrada, será possível simular o erro (sendo uma exceção ou um erro de apresentação). Através da maneira que aluno preferir, seja com uso de depurador ou com uso de comando de print, será possível alcançar a resposta correta sem o auxilio direto do professor. Com isso, é possível que aconteça uma compreensão maior relacionada ao que está acontecendo no código e ao que é pedido na questão.

Exceções tratadas	Mensagem
ZeroDivisionError	Divisão por zero
IndexError	Índice não encontrado
SyntaxError	Erro de Sintáxe
NameError	Variável não definida
TypeError	Erro de tipo
ValueError	Erro de valor
OSError	Erro de sistema operacional
Exception	Exceção não encontrada

Tabela 2. Tabela de exceções tratadas inicialmente.
Fonte: autoria própria

O fluxo de integração do sistema descrito na figura 3 segue da seguinte forma: o *backend* é responsável por se comunicar com o banco de dados e com o subsistema de submissão de casos de teste.

O subsistema diz respeito a uma *API* feita em python para tratar o código fonte enviado pelo usuário e com isso reproduzir um caso de teste, obtendo uma resposta com os seguintes atributos no formato JSON (*Javascript Object Notation*): erro, saida, status e tempo de execução. Após obter essa resposta, ela é enviada para o *backend* java, para que ele possa receber e salvar no banco de dados. Com isso, pode ocorrer a validação para cada caso de teste. Esse processo se dá através da comparação entre a saída obtida pelo código feito pelo usuário, com a saída esperada salva no banco de dados. Se houve sucesso em todos os casos de teste, a submissão foi um sucesso [Crockford 2006].

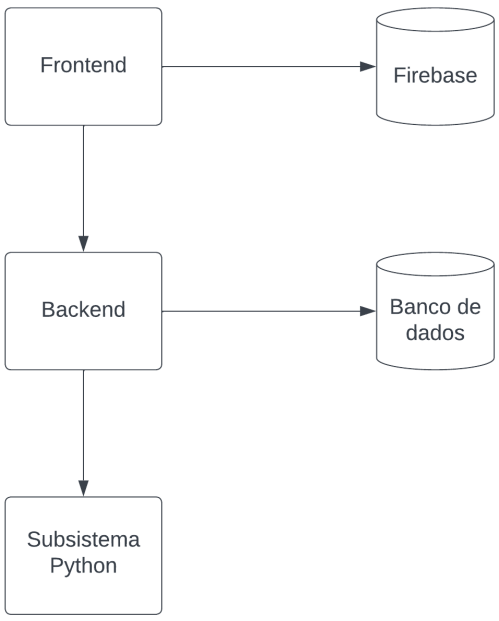

Figura 3. Fluxo de integração
Fonte: autoria própria

Foi possível a implementação do protótipo funcional do sistema, de acordo com o escopo escolhido, com exceção da utilização do docker com finalidade de isolar a execução do caso de teste. Apesar de não conseguir implementar da maneira inicialmente formulada, conseguimos contornar nossa falta de maturidade no uso de docker, criando um subsistema e tratando para que não seja possível importar bibliotecas ao fazer uma submissão. Devido a essa alteração de implementação tivemos que mudar a ordem que desenvolvemos o sistema, iniciando com o subsistema, e posteriormente implementando todo o *backend*, para assim poder testar a submissão e verificar se estava funcionanado da maneira esperada. Após isso, iniciamos o desenvolvimento da interface web, e a medida que a interface progredia, alterações necessárias no banco e no *backend* foram realizadas.

O projeto possui outras lacunas, além da falta de conteinerização, passíveis de serem melhoradas devido a nossa escolha de escopo e elas podem se tornar ideias, que serão um ponto de partida para outros projetos futuros que agreguem no nosso sistema.

Todo o processo de desenvolvimento se deu de forma local, então caso o leitor queira executar o sistema, como também visualizar o código fonte, isso é possível através do repositório https://github.com/jpalvesl/tcc que contém um passo a passo para subir o sistema e também sumariza os repositórios de código no seu *README*.

6. Conclusão

Este trabalho teve como principal objetivo desenvolver um sistema de apoio ao ensino de programação permitindo a avaliação de programas por meio de casos de teste, com o foco prioritário na aprendizagem. Focou-se inicialmente nas funcionalidades mais críticas após uma análise de outros sistemas já presentes no mercado. O objetivo do trabalho foi atingido e sua aplicabilidade será feita com as turmas de algoritmos e programação do IFPB campus Campina Grande, obtendo assim um *feedback* dos alunos e professores, visando os pontos de melhoria e otimização, que poderão ser implementados por novos alunos responsáveis por seguir com o desenvolvimento da aplicação, podendo evoluí-la seguindo as práticas de desenvolvimento de software.

6.1. Considerações finais

As tecnologias utilizadas neste trabalho apresentaram um resultado satisfatório, assim oferecendo uma aplicação moderna com uma boa usabilidade e interface amigável para o usuário. Apesar de ser a primeira vez que os autores deste trabalho desenvolveram um sistema desse tipo, a familiaridade já existente com algumas dessas tecnologias trouxeram um certo conforto ao trabalho, e aquelas sobre as quais não existia certo conhecimento ou uso anterior se revelaram como uma escolha acertada, em vista de possuirem uma interface amigável, comunidade ativa, serem ferramentas de propósito específico e possuirem uma ampla variedade de informações.

A principal limitação deste trabalho foi a falta de conhecimento com gerenciamento de escopo, e apesar da escolha de um escopo reduzido essa falta de experiencia na concepção de um sistema desde sua base, fez com que houvesse bastante reatrabalho ao longo do projeto.

6.2. Sugestões para Trabalhos Futuros

- Conteinerização da execução de cada caso de teste durante a execução da submissão.

- Criação de um sistema de avaliação automático de dificuldade de questões.
- Análise de similaridade de código fonte.
- Adição de novas linguagens de programação para submissão.
- Monetização através da coleta de informações do perfil do usuário e a conexão desse perfil com empresas interessadas.
- Uma abordagem que permita o uso do sistema localmente, para sincronização posterior.

Referências

Ant Design (2022). Ant Design of React. Disponível em: https://ant.design/docs/react/introduce. Acesso em: 01 Dez 2022.

AWS (2022). O que é Java? Disponível em: https://aws.amazon.com/pt/what-is/java/. Acesso em: 30 Nov 2022.

beecrowd (2023). beecrowd. Disponível em: https://www.beecrowd.com.br/. Acesso em: 09 Mai 2023.

Candido, C. H. and dos Santos Mello, R. (2017). Ferramenta de modelagem de banco de dados relacionais brmodelo v3. *XIII Escola Regional de Banco de Dados (ERBD)*.

CodeChef (2023). CodeChef: Pratical coding for everyone. Disponível em: https://www.codechef.com/. Acesso em: 09 Mai 2023.

Crockford, D. (2006). The application/json media type for javascript object notation (json). Technical report.

DBeaver (2023). About — DBeaver Community. Disponível em: https://dbeaver.io/about/. Acesso em: 15 Mai 2023.

Docker (2023). Docker: Accelerated, Containerized Application Development. Disponível em: https://www.docker.com/. Acesso em: 20 Jun 2023.

Figma (2023). Figma: the collaborative interface design tool. Disponível em: https://www.figma.com/. Acesso em: 20 Jun 2023.

Flanagan, D. (2011). *JavaScript : the definitive guide*. O'Reilly Media.

Francisco, R. E., Ambrósio, A. P. L., Junior, C. X. P., and Fernandes, M. A. (2018). Juiz online no ensino de cs1-lições aprendidas e proposta de uma ferramenta. *Revista Brasileira de Informática na Educação*, 26(03):163.

Git (2022). About. Disponível em: https://git-scm.com/about/branching-and-merging. Acesso em: 01 Dez 2022.

Github (2023). Github Docs. Disponível em: https://docs.github.com/pt. Acesso em: 09 Mai 2023.

HackerRank (2023). HackerRank - Online Coding Tests and Technical Interviews. Disponível em: https://www.hackerrank.com/. Acesso em: 09 Mai 2023.

Maharry, D. (2013). *TypeScript revealed*. Apress.

MARIADB (2022). About MariaDB Server. Disponível em: https://mariadb.org/about/. Acesso em: 30 Nov 2022.

Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson.

NOTION (2022). A story of tools and the future of work. Disponível em: https://www.notion.so/about. Acesso em: 30 Nov 2022.

Oracle (2023). Drivers JDBC — Oracle Brasil. Disponível em: https://www.oracle.com/br/database/technologies/appdev/jdbc.html. Acesso em: 15 Mai 2023.

React (2022). React A JavaScript library for building user interfaces. Disponível em: https://reactjs.org/. Acesso em: 01 Dez 2022.

Santos, J. C. and Ribeiro, A. R. (2012). Jonline: proposta preliminar de um juiz online didático para o ensino de programação. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 1.

SPRING (2022). Spring Boot. Disponível em: https://mariadb.org/about/. Acesso em: 30 Nov 2022.

Sriramya, P. and Karthika, R. (2015). Providing password security by salted password hashing using bcrypt algorithm. *ARPN journal of engineering and applied sciences*, 10(13):5551–5556.

STACK OVERFLOW (2021). 2021 Developer Survey. Disponível em: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language. Acesso em: 30 Nov 2022.

SWAGGER (2022). About Swagger. Disponível em: https://swagger.io/about/. Acesso em: 1 Dez 2022.

The Huxley (2023). The Huxley. Disponível em: https://www.thehuxley.com/. Acesso em: 09 Mai 2023.

Valente, M. T. (2020). Engenharia de software moderna. *Princípios e Práticas para Desenvolvimento de Software com Produtividade*, 1:24.

WHIMSICAL (2022). Pricing. Disponível em: https://whimsical.com/pricing. Acesso em: 30 Nov 2022.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Ostensivo (Público)

Versão final do TCC

Assunto: Versão final do TCC
Assinado por: Myrlla Pereira
Tipo do Documento: Anexo
Situação: Finalizado
Nível de Acesso: Ostensivo (Público)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Myrlla Lucas Pereira**, **ALUNO (201821250041) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 10/07/2023 20:22:19.

Este documento foi armazenado no SUAP em 10/07/2023. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 874580
Código de Autenticação: c903e4b92d