UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

LUCAS GABRIEL DE LIMA SILVA

DESENVOLVIMENTO DE UM SOFTWARE WEB PARA AUXILIAR O PROCESSO DE ENSINO-APRENDIZAGEM DA DISCIPLINA DE INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO (ICC) DA UNESP - INSTITUTO DE CIÊNCIA E TECNOLOGIA DE SOROCABA

Sorocaba
2024

LUCAS GABRIEL DE LIMA SILVA

DESENVOLVIMENTO DE UM SOFTWARE WEB PARA AUXILIAR O PROCESSO DE ENSINO-APRENDIZAGEM DA DISCIPLINA DE INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO (ICC) DA UNESP - INSTITUTO DE CIÊNCIA E TECNOLOGIA DE SOROCABA

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Márcio Alexandre Marques

Sorocaba

2024

S586d

Silva, Lucas Gabriel de Lima

Desenvolvimento de um software Web para auxiliar o processo de ensino-aprendizagem da disciplina de Introdução à Ciência da Computação (ICC) da UNESP - Instituto de Ciência e Tecnologia de Sorocaba / Lucas Gabriel de Lima Silva. -- Sorocaba, 2024

76 p. : il., fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (Unesp), Instituto de Ciência e Tecnologia, Sorocaba

Orientador: Márcio Alexandre Marques

1. Ciência da computação. 2. C (Linguagem de programação de computador). 3. Software – Desenvolvimento. 4. Software educacional. 5. Material didático. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba.
Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

DESENVOLVIMENTO DE UM SOFTWARE WEB PARA AUXILIAR O PROCESSO DE ENSINO-APRENDIZAGEM DA DISCIPLINA DE INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO (ICC) DA UNESP - INSTITUTO DE CIÊNCIA E TECNOLOGIA DE SOROCABA

LUCAS GABRIEL DE LIMA SILVA

ESTE PROJETO FINAL DE CURSO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO GRAU DE **BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

Prof. Dr. Everson Martins
Coordenador

BANCA EXAMINADORA:

Prof. Dr. MÁRCIO ALEXANDRE MARQUES
Orientador/UNESP – Câmpus de Sorocaba

Prof.ª Dra. MARIA LÚCIA PEREIRA ANTUNES
UNESP – Câmpus de Sorocaba

Prof. Dr. EDUARDO VERRI LIBERADO
UNESP – Câmpus de Sorocaba

Maio de 2024

AGRADECIMENTOS

Agradeço aos meus pais, Nelson e Maria, por todo apoio, confiança e incentivo proporcionados durante a graduação e aos meus irmãos, Letícia e Leonardo, por serem exemplos de dedicação e grandes inspirações para mim ao longo da trajetória acadêmica.

Agradeço à minha namorada, Érika Dantas, por sempre estar ao meu lado, me apoiando nas minhas decisões e não deixando que eu desista dos meus objetivos.

Agradeço ao meu orientador, Prof. Dr. Márcio Alexandre Marques, por sua orientação, confiança e auxílio. Seu conhecimento e apoio foram fundamentais para o desenvolvimento deste trabalho.

Agradeço aos meus amigos, em especial ao Guilherme Doubek, Giovanni Dotta, Matheus Nunes e Willian Saito pela parceria em todos os momentos bons e ruins vivenciados no decorrer da graduação, inclusive durante o desenvolvimento deste trabalho. Vocês fazem parte da minha história e das minhas conquistas e tudo o que vivemos juntos ficará guardado para sempre em minha memória.

Agradeço a todo o corpo docente, pelos conhecimentos, ensinamentos e conselhos que moldaram minha formação profissional e pessoal, bem como aos funcionários e prestadores de serviço da UNESP Câmpus de Sorocaba, por me ajudarem com as questões administrativas da graduação.

Minha eterna gratidão a todos vocês que fizeram parte da minha trajetória acadêmica, sem a ajuda fornecida não teria chegado até aqui.

RESUMO

Nos cursos de graduação da área de tecnologia, grande parte dos discentes vêm apresentando dificuldades no aprendizado de disciplinas introdutórias de programação, tanto por conta da complexidade inerente ao seu conteúdo, quanto pela ausência de formas alternativas de ensino que estimulem o interesse pelo assunto e mantenham os alunos em um ritmo constante de estudo. Esta situação tem levado ao uso de Softwares Educacionais (SE) para auxiliar no processo de construção da aprendizagem. Embora existam Softwares Educacionais disponíveis, como o Codecademy e HackerRank, eles carecem de uma solução unificada que integre os recursos de material didático, exercícios práticos com *feedback* automático e monitoramento das atividades pelo docente. Diante disso, o presente trabalho propõe desenvolver um protótipo de Software Educacional que integre os recursos supracitados de forma personalizada para a disciplina de Introdução à Ciência da Computação (ICC) do curso de graduação em Engenharia de Controle e Automação da UNESP Câmpus de Sorocaba. Para o desenvolvimento do protótipo, utilizou-se a ferramenta Figma na elaboração do *layout*, o Trello na definição das regras de negócio em formato de tarefas, o DB Designer na estruturação do banco de dados e o Visual Studio Code na construção dos códigos do *front-end* e do *back-end*. Foi possível desenvolver a aplicação e disponibilizá-la na internet para ser acessada por meio de um *link*. A ferramenta foi testada por ex-alunos da disciplina de ICC e ao final do período de validação, eles responderam um formulário, fornecendo apreciações e avaliações pertinentes. As respostas obtidas demonstraram um alto nível de satisfação dos participantes com o protótipo, destacando a sua fácil usabilidade, a relevância do material didático, o valor dos exercícios com *feedback* automático e a importância do acompanhamento do docente. A análise dos resultados evidenciou não só a eficácia de cada um dos recursos individualmente, mas também do protótipo como um todo em relação à usabilidade e à relevância destes para a disciplina. Com base nos resultados obtidos, foi possível concluir que o protótipo de Software Educacional, desenvolvido de forma personalizada para a disciplina de ICC, integra com sucesso os recursos de material didático, exercícios com *feedback* automático e monitoramento das atividades pelo docente.

Palavras-chave: ensino de programação; software educacional; Introdução à Ciência da Computação; desenvolvimento *web*; *feedback* automático.

ABSTRACT

In undergraduate courses in the technology field, a large portion of students have been facing difficulties in learning introductory programming disciplines, both due to the inherent complexity of their content and the absence of alternative teaching methods that stimulate interest in the subject and keep students in a constant study rhythm. This situation has led to the use of Educational Software (ES) to assist in the learning process. Although there are available Educational Software such as Codecademy and HackerRank, they lack a unified solution that integrates resources such as educational material, practical exercises with automatic feedback, and monitoring of activities by the teacher. Therefore, this work proposes to develop an Educational Software prototype that integrates the aforementioned resources in a personalized way for the Introduction to Computer Science (ICC) discipline of the Control and Automation Engineering course at UNESP Sorocaba. For the development of the prototype, the Figma tool was used for layout design, Trello for defining business rules in task format, DB Designer for structuring the database, and Visual Studio Code for building front-end and back-end codes. It was possible to develop the application and make it available on the internet to be accessed through a link. The tool was tested by former ICC students, and at the end of the validation period, they responded to a form, providing relevant feedback and evaluations. The responses obtained demonstrated a high level of satisfaction from participants with the prototype, highlighting its easy usability, the relevance of the educational material, the value of exercises with automatic feedback, and the importance of teacher monitoring. The analysis of the results not only evidenced the effectiveness of each resource individually but also the prototype as a whole regarding usability and relevance for the discipline. Based on the results obtained, it was possible to conclude that the prototype of Educational Software, developed specifically for the ICC discipline, successfully integrates the resources of educational materials, exercises with automatic feedback, and monitoring of activities by the professor.

Keywords: programming education; educational software; Introduction to Computer Science; web development; automatic feedback.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

As disciplinas de programação são comumente ministradas nos períodos iniciais de cursos de graduação da área de tecnologia, com o intuito de ensinar o discente a solucionar problemas reais, por meio do desenvolvimento de programas (COSTA, 2013). Entretanto, grande parte dos alunos possuem dificuldades em aprender os conceitos básicos da matéria, o que gera um aumento no índice de reprovação e evasão destas disciplinas (RAPOSO; DANTAS, 2016).

Segundo Júnior *et al*. (2005), dentre os problemas encontrados no processo de ensino-aprendizagem de programação, pode-se citar: a dificuldade de interpretação dos enunciados e abstração dos problemas; dificuldade na identificação de pré-requisitos necessários para o desenvolvimento dos programas, como por exemplo, habilidades matemáticas; dificuldade na aplicação de competências prévias na construção dos algoritmos (JÚNIOR *et al.*, 2005).

Ademais, Raposo e Dantas (2016) acrescentam dois fatores que prejudicam o ensino de programação introdutória: conteúdos de caráter acumulativo, levando alunos que não se dedicam a terem dificuldade de acompanhar a disciplina; a ausência de formas alternativas que estimulem o interesse pelo conteúdo e mantenham os discentes em um ritmo constante de estudo (RAPOSO; DANTAS, 2016).

Diante desse cenário, uma das estratégias bastante comum atualmente e que tem sido utilizada para auxiliar no processo de construção da aprendizagem é o uso do Software Educacional (SE). O Software Educacional é uma categoria de software projetada especificamente para auxiliar no processo de ensino-aprendizagem em ambientes educacionais (ALMEIDA *et al.*, 2018). Esses softwares são desenvolvidos com o objetivo de oferecer suporte pedagógico em diversas áreas do conhecimento, proporcionando recursos interativos, exercícios práticos, *feedback* automático e material didático complementar. O SE visa aprimorar a experiência de aprendizagem dos estudantes, promovendo a compreensão de conceitos complexos, o desenvolvimento de habilidades específicas e a autonomia no estudo.

Na área da programação, um exemplo de SE é o BeeCrowd Academic, uma plataforma *online* que permite aos professores criar e gerenciar disciplinas, convidar alunos e designar exercícios de programação para serem resolvidos. Os alunos, por sua vez, têm a oportunidade de submeter suas soluções e receber *feedback* automático sobre suas respostas. Além disso, os professores têm acesso a um painel que fornece uma visão abrangente das submissões e do desempenho dos alunos (BEECROWD, 2021). No entanto, é importante ressaltar que o

BeeCrowd Academic se concentra principalmente na prática da programação e na resolução de problemas, carecendo de recursos que ofereçam um material didático de apoio para os exercícios propostos.

Outro exemplo de SE é o Codecademy, uma plataforma *online* gratuita que disponibiliza cursos de linguagens de programação. Ela oferece uma combinação de conteúdo teórico e exercícios práticos, junto a correções e *feedbacks* sobre as soluções submetidas (CODECADEMY, 2024). Contudo, por ter sido projetada para o uso individual pelos usuários, a plataforma não inclui suporte para que os professores acompanhem as entregas e o desempenho dos alunos.

Um terceiro exemplo é o HackerRank, uma plataforma conhecida por sua ênfase em desafios de programação competitiva, tanto para indivíduos quanto para empresas. Nessa plataforma, os desenvolvedores competem para criar programas que atendam às especificações fornecidas, e os desafios podem ser abordados em várias linguagens de programação (HACKERRANK, 2024). No entanto, assim como o CodeAcademy, o HackerRank é projetado principalmente para uso individual e não oferece suporte para o acompanhamento dos professores.

Diante do exposto, é possível perceber que as plataformas educacionais existentes oferecem tipos específicos de recursos, como videoaulas, exercícios de programação ou monitoramento das atividades propostas, de acordo com o propósito da ferramenta. Contudo, nenhuma delas oferece uma solução unificada englobando todas essas características, podendo constituir um obstáculo na viabilização de uma experiência de aprendizagem completa e que atenda às particularidades de aprendizado de cada aluno.

Portanto, o objetivo deste trabalho foi desenvolver um protótipo de Software Educacional que integrasse os recursos de material didático de apoio, exercícios práticos com *feedback* automático e monitoramento das atividades pelo docente, de forma personalizada para a disciplina introdutória de Introdução à Ciência da Computação (ICC) do curso de Engenharia de Controle e Automação da UNESP - Câmpus Sorocaba.

2 REVISÃO BIBLIOGRÁFICA

Nesta seção são apresentados os conceitos, tecnologias e recursos utilizados no desenvolvimento do protótipo de Software Educacional para a disciplina de ICC, sendo dividida nos seguintes tópicos: desenvolvimento *web*, linguagem de programação; tecnologias para desenvolvimento de *front-end*; tecnologias para desenvolvimento de *back-end*; sistemas de controle de versão de código; ferramentas auxiliares utilizadas durante o desenvolvimento do projeto.

2.1 Desenvolvimento *Web*

Segundo Roveda (2020), desenvolvimento *web* é a área da tecnologia voltada à construção de *sites*, aplicativos, softwares, banco de dados e quaisquer outras ferramentas que, de certa forma, constroem a internet como é conhecida hoje. Dentro do desenvolvimento *web* existem diversas subáreas, como por exemplo, *front-end*, *back-end*, infraestrutura de sistemas, entre outras (ROVEDA, 2020).

O *front-end* corresponde a toda a parte visível de um *site*, com a qual o usuário é capaz de interagir. Sendo assim, seu desenvolvimento não só envolve as linguagens de programação voltadas à construção da interface gráfica apresentada pelo navegador ao acessar um determinado *site*, como o HTML, CSS e JavaScript, mas também pode envolver bibliotecas que facilitam esse processo, como React, Angular, Vue, jQuery, entre outras (ROVEDA, 2020).

Já o *back-end* envolve a programação de toda a parte interna de um *site*, isto é, tudo que diz respeito ao seu funcionamento e armazenamento de dados e informações. Ao interagir com um *site*, por meio de sua interface visível (*front-end*), o usuário pode inserir informações que serão processadas por um servidor e/ou armazenadas por um banco de dados, logo, os dois últimos representam o *back-end* da aplicação. Dentre as linguagens de programação utilizadas no *back-end*, pode-se citar: JavaScript, Python, Java, Ruby e PHP (ROVEDA, 2020).

2.2 Linguagem de programação

Segundo Gotardo (2015), uma linguagem de programação consiste em um método padronizado utilizado para expressar as instruções de um programa a um computador que possa ser programado. Por essa razão, ela deve seguir um conjunto de regras sintáticas (relativas à

forma de escrita) e semânticas (relativas ao conteúdo), para definir um programa de computador. Por meio da definição de uma linguagem de programação, é possível especificar quais dados um computador vai utilizar, como eles serão tratados, armazenados, transmitidos e quais ações deverão ser tomadas em determinadas circunstâncias (GOTARDO, 2015).

Nesse sentido, a linguagem de programação fundamentalmente utilizada por todas as tecnologias envolvidas no desenvolvimento da aplicação *web* deste projeto é chamada de JavaScript, porém, a fim de garantir uma melhor qualidade na construção do software, a linguagem final utilizada na codificação dos programas é derivada da primeira e denominada TypeScript. Os motivos e vantagens de sua utilização são explicitados no tópico seguinte.

2.3 Typescript

Desenvolvida pela Microsoft, o TypeScript (TS) é uma linguagem de programação que, conforme mencionado anteriormente, é derivada do JavaScript, adicionando tipagem e diversos outros recursos a esta. Ele foi criado para permitir a checagem de tipagem estática no código, um recurso muito útil durante o desenvolvimento de aplicações em larga escala (LEASE, 2018).

O JavaScript possui uma tipagem dinâmica, ou seja, o tipo de dado de uma variável só é definido após ser atribuído a ela um valor em tempo de execução. Assim, a variável pode receber um novo valor de tipo diferente sem a ocorrência de erros ou avisos, podendo resultar em *bugs* (termo usado para descrever erros, falhas ou comportamentos inesperados em um *site*) que passem despercebidos, especialmente em aplicações grandes (LEASE, 2018).

Já o TypeScript utiliza a tipagem estática, isto é, as variáveis recebem um tipo quando são declaradas. Dessa forma, o TypeScript verifica os tipos adotados em tempo de compilação e dispara um erro caso seja atribuído um valor de tipo diferente à variável em questão. Contudo, o disparo do erro não impede a execução do código, pois ele continua sendo “transpilado” (processo de tradução de código de uma linguagem de programação para outra similar, mantendo a mesma funcionalidade) para JavaScript e executado normalmente. Em suma, o TypeScript avisa o desenvolvedor quando há algo errado, envolvendo a tipagem, mas não modifica a forma como ele é executado (LEASE, 2018).

Pelos motivos supracitados, a utilização do TypeScript é crucial para a construção de um software com uma qualidade melhor e que possa ser escalonado de maneira adequada.

2.4 Tecnologias para desenvolvimento de *front-end*

Atualmente, o desenvolvimento de *front-end* de uma aplicação *web* tem sido facilitado e acelerado graças ao uso de tecnologias voltadas a esse objetivo. Dentre as tecnologias *front-end* existentes para a linguagem de programação JavaScript, as que se encontram em destaque são: Angular, React e Vue (SOUZA; SILVA, 2021).

Dentre elas, o React possui grandes vantagens na sua utilização, pois permite a adição de outras bibliotecas e garante a construção de *sites* complexos e performáticos, conforme é explicitado no tópico a seguir.

2.4.1 React

React é uma biblioteca JavaScript criada pela empresa Meta, cujo foco é facilitar o processo de criação de interfaces de usuário interativas (META PLATFORMS, 2024). Trata-se de uma das tecnologias JavaScript mais populares nos dias de hoje. Segundo Greif, Benitte e Rambeau (2018), 64,8% dos desenvolvedores já a utilizaram e voltariam a utilizá-la (GREIF; BENITTE; RAMBEAU, 2018).

Ela trabalha com o conceito de “componentização”, que nada mais é do que dividir a interface que se deseja construir em pequenos componentes, cada um com a sua estrutura, estilização e comportamento. Dessa forma, é possível tanto reaproveitá-los em partes do *layout* que possuem um mesmo padrão de estilo, modificando apenas alguma informação exibida em tela, como também utilizá-los para compor componentes maiores e mais complexos (META PLATFORMS, 2024).

O React também permite que sejam adicionadas outras bibliotecas (ou pacotes) durante o desenvolvimento da interface, de forma a atenderem objetivos específicos, como por exemplo, facilitar a execução de requisições HTTP, descrever a funcionalidade de certos elementos, por meio de ícones intuitivos, entre outros (META PLATFORMS, 2024).

2.5 Tecnologias para desenvolvimento de *back-end*

Da mesma forma que a construção de interfaces de aplicações *web* tem sido auxiliada por diversos tipos de tecnologias, o *back-end* também possui ferramentas que visam acelerar o processo de desenvolvimento de um servidor. No contexto da linguagem de programação

JavaScript, dentre as tecnologias utilizadas no lado do servidor, tem-se o Node.js, Express, Prisma e PostgreSQL, as quais são explicitadas a seguir.

2.5.1 Node.js

O Node.js é um software de código aberto, multiplataforma, desenvolvido com o motor V8 do Google Chrome e que permite a execução de códigos em JavaScript fora de um navegador *web*. Sua criação fez com que muitos desenvolvedores *front-end* que programam em JavaScript no navegador, se tornassem aptos a desenvolver também servidores sem a necessidade de aprender uma linguagem de programação completamente diferente (OPENJS FOUNDATION, 2024).

Uma de suas principais características é sua arquitetura assíncrona e orientada a eventos, permitindo-o seguir um modelo de I/O (entrada e saída) sem bloqueio (ou não bloqueante). Em outras palavras, quando um servidor recebe uma requisição ou precisa do retorno de um banco de dados, sua execução não é bloqueada até a tarefa assíncrona ser finalizada, pelo contrário, ele passa imediatamente para a próxima tarefa síncrona enquanto a anterior é resolvida, voltando a esta após sua finalização. Além disso, o Node.js é *single-thread*, ou seja, cada aplicação é executada em um único processo, sem a criação de uma nova *thread* para cada requisição (ARORA, 2022).

As características supracitadas fazem com que essa tecnologia seja extremamente rápida e eficiente, tornando-a na ferramenta de desenvolvimento *back-end* em JavaScript mais popular atualmente (ARORA, 2022).

2.5.2 Express

O Express é um dos mais populares *frameworks* (estrutura de software que fornece um conjunto de ferramentas, bibliotecas, padrões e diretrizes para ajudar a criar e gerenciar aplicativos da *web*) para Node.js, sendo minimalista, flexível e fornecendo um conjunto robusto de recursos para a construção de servidores HTTP (STRONGLOOP; IBM, 2017).

Desenvolver um servidor com Node.js, utilizando somente suas funções nativas, pode ser um trabalho muito verboso, confuso e limitado em recursos, além de fazer com que os

programadores escrevam uma grande quantidade de código padrão em todos os projetos (HAHN, 2016).

Nesse contexto, o Express existe para reduzir esse código padrão, simplificando a API do Node.js e adicionando novos recursos úteis (HAHN, 2016).

2.5.3 Prisma

Uma das técnicas que tem sido cada vez mais utilizada nos dias de hoje é a de ORM (Object-Relational Mapping ou Mapeamento Objeto-Relacional), que consiste em aproximar o paradigma de programação de orientação a objetos ao paradigma do banco de dados relacional. A tecnologia que utiliza essa técnica é chamada de "mapeador" objeto-relacional (recebendo também a sigla ORM), a qual geralmente consiste em uma biblioteca ou *framework* que permite relacionar objetos a nível de código com os dados que os mesmos representam no banco de dados, auxiliando no próprio uso do banco (FONSECA, 2019).

Nesse contexto, o Prisma é um ORM de código aberto, cuja arquitetura é formada por três partes: Prisma Client, ferramenta que permite a construção de *queries* (instruções utilizadas para interagir com um banco de dados) de forma automática, sem a necessidade de utilizar a linguagem SQL, além de prevenir erros relacionados à tipagem quando utilizado com Node.js e TypeScript; Prisma Migrate, um sistema de migração que permite o versionamento das modificações feitas nas tabelas do banco de dados; Prisma Studio, uma interface do usuário, na qual é possível visualizar e editar os dados armazenados no banco (PRISMA DATA, 2024).

Pelos motivos mencionados anteriormente, o Prisma é uma excelente ferramenta para facilitar a construção da estrutura do banco de dados, o relacionamento entre as tabelas, o versionamento das modificações feitas nas mesmas e as operações realizadas com a base de dados.

2.5.4 PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados do tipo objeto-relacional e de código aberto, com um foco maior em extensibilidade e em padrões de conformidade. Dentre suas principais funcionalidades, é possível destacar o armazenamento seguro dos dados, seguindo as melhores práticas, permitindo a recuperação dos dados quando

solicitado por outras aplicações e efetuando tais operações por meio da linguagem SQL. Além disso, essa tecnologia consegue lidar bem com altos volumes de solicitações e cargas de trabalho, funcionando muito bem para *sites* que são utilizados de forma simultânea por vários usuários (DEVMEDIA, 2015).

2.6 Sistemas de controle de versão

O desenvolvimento de software é um trabalho que exige extrema atenção, pois qualquer alteração feita no código de forma errônea pode resultar no não funcionamento de alguma parte do programa. Este problema pode inclusive se agravar à medida que aumenta o número de pessoas que utilizam o mesmo código. Em um caso desastroso, retornar à versão anterior do software pode facilmente resolver o problema, porém, se não há uma gestão dos arquivos do programa, este procedimento não é possível de ser realizado (SOUZA, 2015).

Nesse contexto, os sistemas de controle de versão existem para que seja possível gerenciar diferentes versões de um documento. Assim, é possível organizar o projeto de forma inteligente e eficaz, podendo acompanhar o histórico de desenvolvimento, desenvolver paralelamente com outros programadores, customizar uma certa versão sem alterar o projeto principal, resgatar a aplicação em um ponto em que ela estava estável, entre outras vantagens (SOARES, 2012).

Dentre as ferramentas utilizadas para controle de versão de código, estão o Git e o GitHub, os quais são explicitados a seguir.

2.6.1 Git

O Git é o sistema de controle de versão mais utilizado atualmente e de grande influência no mundo de desenvolvimento de software. Diferentemente das ferramentas que surgiram antes dele, que dependiam de um servidor centralizado, ele é totalmente distribuído. Isso faz com que as pessoas consigam confirmar, registrar e fazer qualquer outra coisa localmente, em tempo real, sem precisar usar uma VPN (*Virtual Private Network* ou Rede Privada Virtual, uma conexão de rede privada entre dispositivos por meio da internet, sendo geralmente utilizada para transmitir dados de forma segura e anônima) ou até mesmo estar *online* (JORGE, 2018).

2.6.2 GitHub

O GitHub é um serviço disponível na internet para armazenamento e colaboração de código, utilizando o Git no controle de versão. Sendo considerado o maior *host* (servidor que armazena e disponibiliza recursos, como arquivos, *sites* ou serviços, para acesso através da internet) único para repositórios Git, nele milhares de desenvolvedores podem colaborar entre si e mostrar seus projetos. Diversos projetos de código aberto o utilizam para hospedagem do Git, revisão de código, rastreamento de problemas, entre outras funcionalidades. Além disso, o GitHub também funciona como uma rede social para programadores, permitindo a interação e a comunicação entre eles (CHACON; STRAUB, 2014).

Pelos motivos supracitados, o GitHub é uma ótima ferramenta de armazenamento do código na nuvem, facilitando o acesso ao mesmo não apenas por meio da máquina local, mas de qualquer lugar que tenha acesso ao repositório em questão, além de possibilitar a existência de um *backup* do código em um lugar seguro.

2.7 Ferramentas auxiliares utilizadas durante o desenvolvimento do projeto

2.7.1 Figma

O Figma é uma ferramenta *online* de *design* e prototipagem utilizada para criar interfaces de usuário para *sites*, aplicativos e outros produtos digitais. Além disso, ela oferece recursos para desenvolvimento de protótipos interativos de alta fidelidade, sistemas de *design* e colaboração entre *designers* e desenvolvedores. Entre suas vantagens estão o fato de ser baseado na *web*, permitindo acesso de qualquer lugar com conexão à internet, a colaboração em tempo real, uma interface intuitiva, recursos poderosos como ferramentas de vetor e texto, uma comunidade ativa que contribui com *plugins* e tutoriais, e uma versão gratuita com muitos recursos disponíveis (VILLAIN; SILVEIRA, 2023).

2.7.2 DB Designer

O DB Designer é uma ferramenta de modelagem de banco de dados disponível gratuitamente *online*, permitindo aos usuários criar modelos visuais de bancos de dados de maneira intuitiva e poderosa. Suas funcionalidades incluem a criação de diagramas de entidade-relacionamento (ER), modelagem de diferentes tipos de bancos de dados (MySQL,

PostgreSQL, Oracle, Microsoft SQL Server, entre outros), colaboração em equipe em tempo real, exportação de modelos para SQL e importação de modelos existentes (DB DESIGNER, 2023).

Dentre suas vantagens, destacam-se sua facilidade de uso, gratuidade para utilização não comercial, suporte a funcionalidades avançadas de modelagem de dados, versatilidade, capacidade de colaboração em equipe e acessibilidade de qualquer lugar com acesso à internet. (DB DESIGNER, 2023).

2.7.3 Trello

O Trello é uma ferramenta de gerenciamento de projetos *online* que permite organizar e monitorar tarefas de maneira visual. Essa ferramenta pode ser utilizada para gerenciar projetos, colaborar com equipes, acompanhar o progresso de atividades, definir prazos e prioridades, e criar listas de tarefas de diversos tipos (ATLASSIAN, 2023).

Dentre as vantagens do Trello estão sua facilidade de uso, flexibilidade e personalização, abordagem visual e intuitiva, disponibilidade de um plano gratuito com muitos recursos, entre outras (RIBEIRO, 2022).

2.7.4 VS Code

O VS Code (Visual Studio Code) é uma ferramenta de edição de código *open-source* (código aberto) desenvolvida pela Microsoft, compatível com Windows, Mac e Linux. Reconhecido por sua leveza e potência, o VS Code oferece uma ampla variedade de recursos para desenvolvedores, como edição de código com suporte a várias linguagens (JavaScript, TypeScript, Python, Java, entre outras), depuração integrada, controle de versão através de integração com Git, e um vasto mercado de extensões que ampliam suas funcionalidades (MICROSOFT, 2024).

Utilizado para escrever e depurar projetos de software, o VS Code é valorizado por sua gratuidade, performance ágil, alta capacidade de personalização e expansão através de extensões, além de uma comunidade ativa e engajada (MIR, 2023).

2.7.5 Notion

O Notion é uma ferramenta multifuncional amplamente reconhecida por sua versatilidade e oferecimento de recursos avançados para anotações. Além dos tradicionais formatos de texto, o Notion se destaca pelas seguintes funcionalidades: personalização com diferentes estilos, tamanhos e cores; criação de subpáginas e listas numeradas para estruturar notas; inserção de imagens, vídeos e áudios; organização de informações em tabelas e inserção de trechos de código (NOTION LABS, 2024).

3 MATERIAIS E MÉTODOS

3.1 Materiais

O hardware utilizado no desenvolvimento do projeto foi um *notebook* com sistema operacional Elementary OS 6.1 Jólnir, processador Intel Core i3 de 6ª geração, placa de vídeo integrada e 4 gigabytes de memória RAM.

Com relação aos softwares, foram utilizados quatro de licença gratuita:

- Trello;
- DB Designer;
- Figma;
- Visual Studio Code.

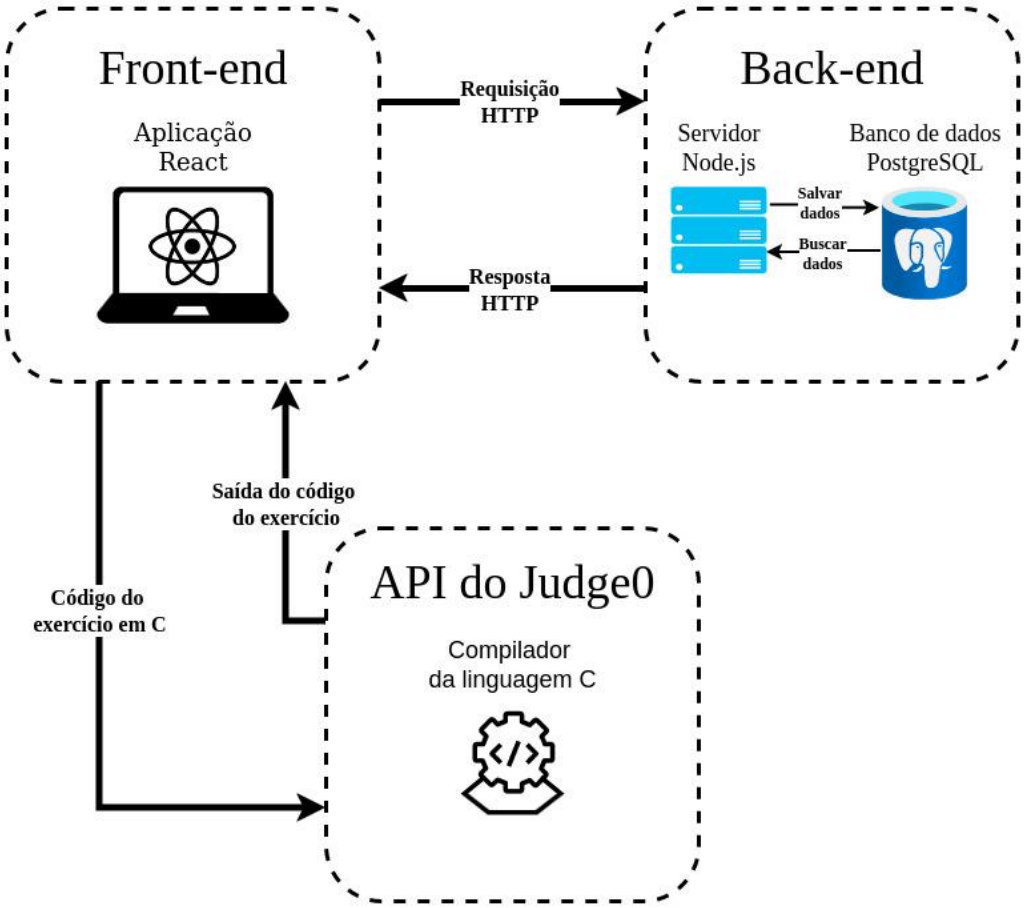
3.2 Métodos

3.2.1 Infraestrutura geral da aplicação

O protótipo foi implementado com base na infraestrutura geral mostrada no desenho da Figura 1 e as etapas de desenvolvimento do mesmo são exibidas no diagrama da Figura 2.

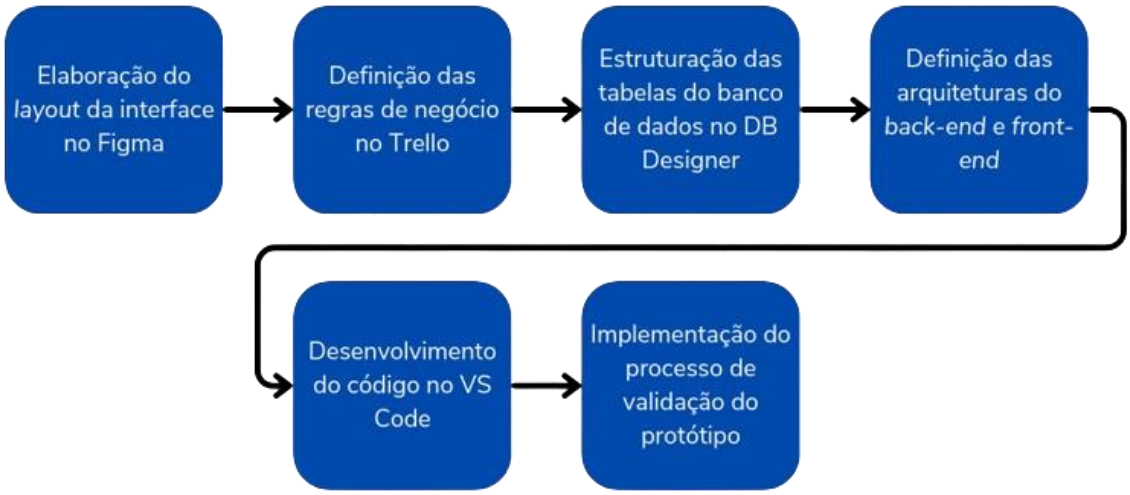
No contexto da Figura 1, o *front-end* é constituído por uma aplicação React que se comunica com o *back-end* por meio de requisições HTTP, para enviar ou receber informações. Já o *back-end* é constituído por um servidor Node.js e um banco de dados PostgreSQL que se comunicam entre si para salvar ou buscar dados. Além disso, o *front-end* também se comunica por meio de requisições HTTP com a API do Judge0, a qual consiste em um servidor contendo compiladores de diversas linguagens de programação, incluindo o da linguagem “C”. Essa API recebe o código dos exercícios em “C”, efetua a compilação e retorna a saída do mesmo para ser exibida na interface do usuário.

Figura 1 – Desenho da infraestrutura geral da aplicação



Fonte: Autoria própria.

Figura 2 – Diagrama das etapas de desenvolvimento do protótipo



Fonte: Autoria própria.

Já as etapas apresentadas na Figura 2 são abordadas com mais detalhes ao longo dos tópicos subsequentes.

3.2.2 Estruturação da interface do usuário

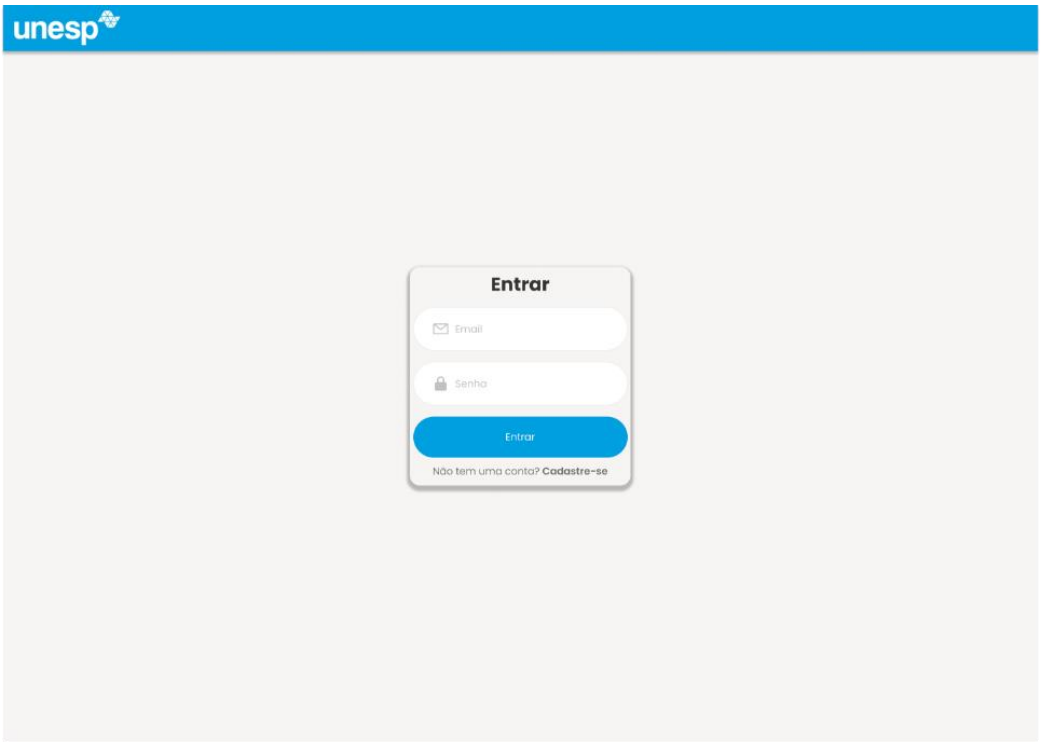
Na estruturação da interface do usuário (*User Interface* ou UI), foram adotadas diversas escolhas de *design* visando a intuitividade e a identidade visual da UNESP. Para tal, o software Figma foi empregado na elaboração do *layout*, o qual compreendeu um total de 13 páginas.

Em determinadas páginas, o *layout* foi concebido de forma a englobar variações dependentes do nível de acesso do usuário (administrador ou usuário comum). A interface de cada página é descrita no decorrer dos tópicos subsequentes.

3.2.2.1 Páginas de login e cadastro

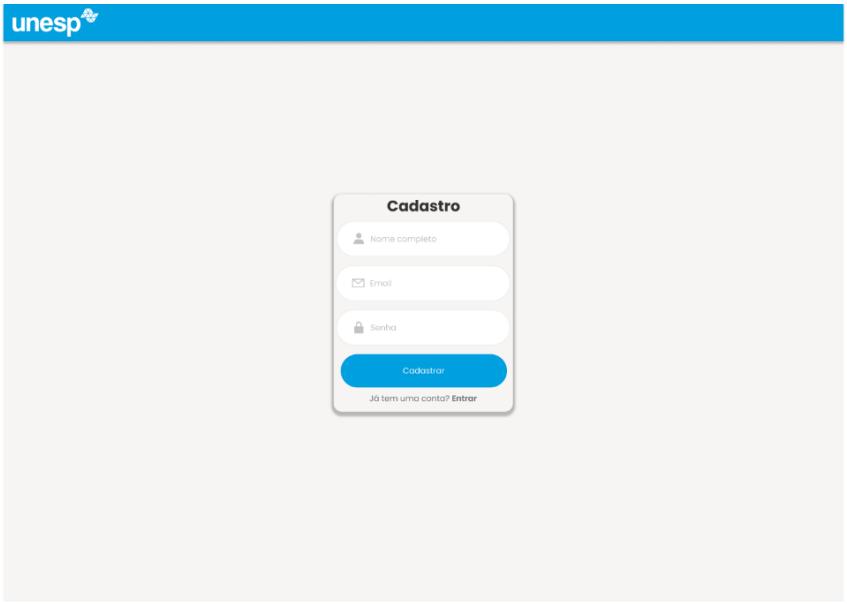
As duas primeiras telas foram projetadas de forma a permitir a autenticação do usuário por meio de formulários de *login* e cadastro do discente, como mostrado nas Figura 3 e Figura 4, respectivamente.

Figura 3 - *Layout* da página de *login*



Fonte: Autoria própria.

Figura 4 - *Layout* da página de cadastro

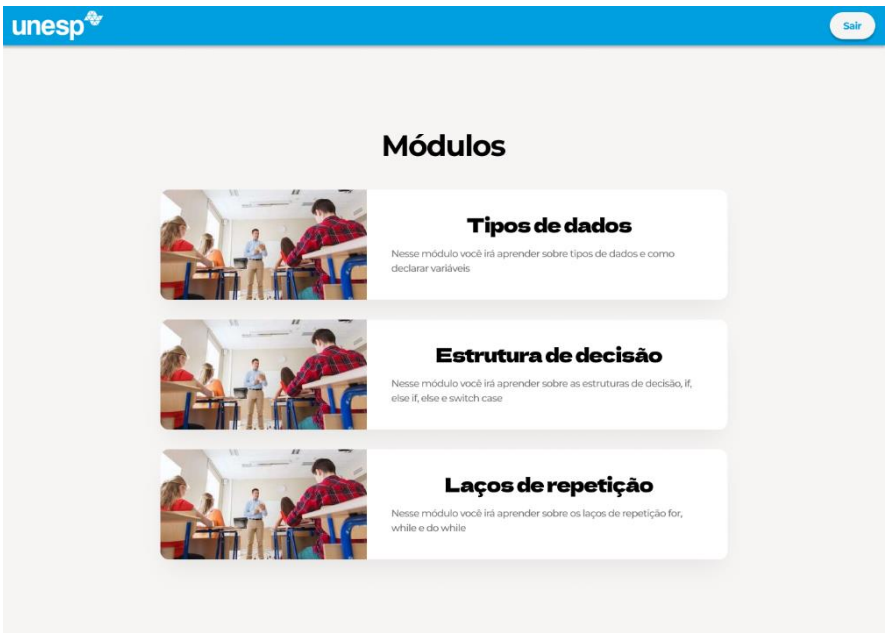


Fonte: Autoria própria.

3.2.2.2 Página de módulos

Após realizar o *login* ou o cadastro, o usuário é encaminhado para a página em que são exibidos os módulos cadastrados, organizados em *cards* (elementos visuais usados para representar uma entidade) contendo informações como nome, descrição e a imagem do módulo (Figura 5).

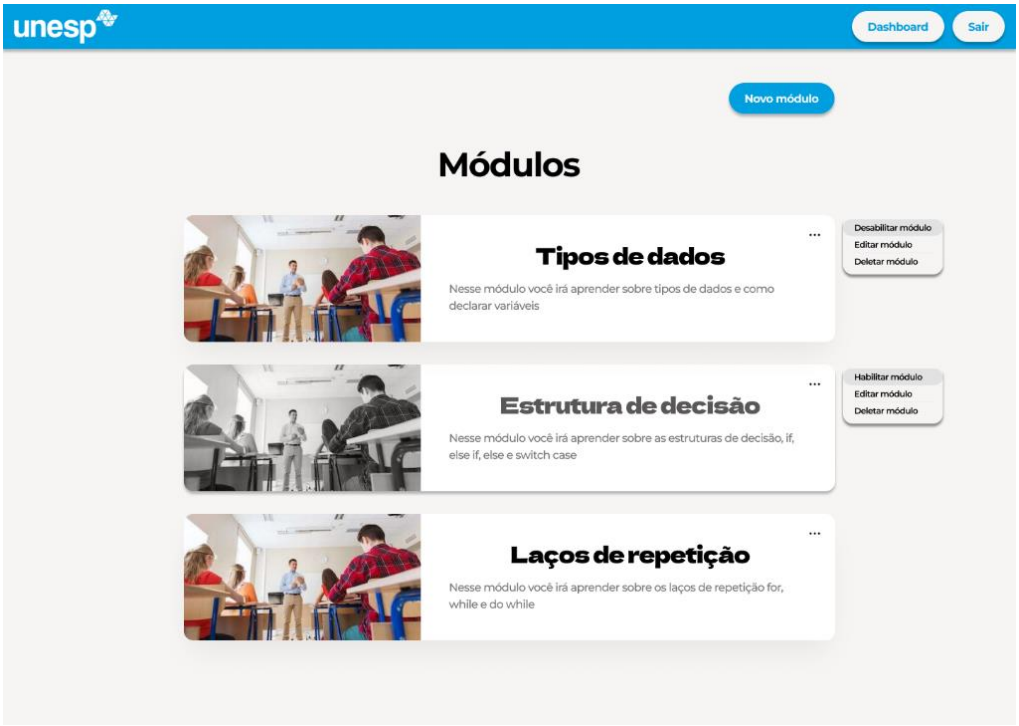
Figura 5 - *Layout* da página de módulos exibida para o usuário comum



Fonte: Autoria própria.

No caso do usuário administrador, nessa página é exibido o botão “Novo módulo” e no *card* de cada módulo também há um menu *meatballs* (elemento visual composto por três pontos alinhados horizontalmente) que, quando clicado, exibe as opções "Desabilitar/habilitar módulo", "Editar módulo" e "Deletar módulo", conforme mostrado na Figura 6.

Figura 6 - *Layout* da página de módulos exibida para o usuário administrador



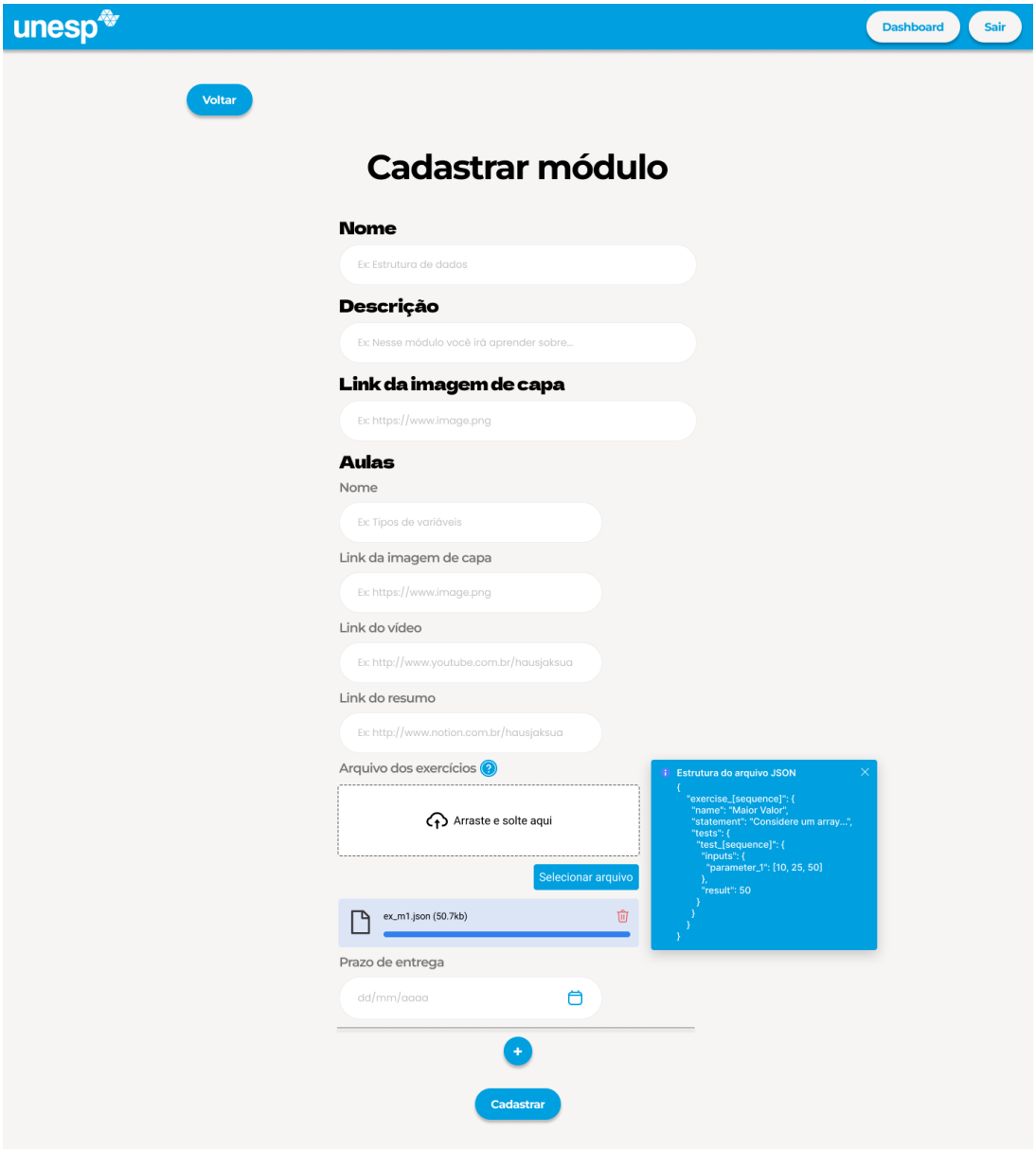
Fonte: Autoria própria.

O administrador também tem acesso aos botões “Dashboard” e “Sair” no cabeçalho, conforme mostrado na Figura 6, sendo estes exibidos em todas as páginas da aplicação.

3.2.2.3 Páginas de cadastro e edição de módulo

Ao clicar no botão “Novo módulo”, o administrador é direcionado para a página de cadastro de módulo, a qual contém um formulário com os campos “Nome”, “Descrição” e “Link da imagem de capa”, relativos ao módulo, e uma seção “Aulas”, com os campos “Nome”, “Link da imagem de capa”, “Link do vídeo”, “Link do resumo”, “Arquivo dos exercícios” e “Prazo de entrega”. Essa página foi projetada para ser acessível somente ao usuário administrador e seu *layout* é mostrado na Figura 7.

Figura 7 - *Layout* da página de cadastro de módulo, acessível somente ao administrador

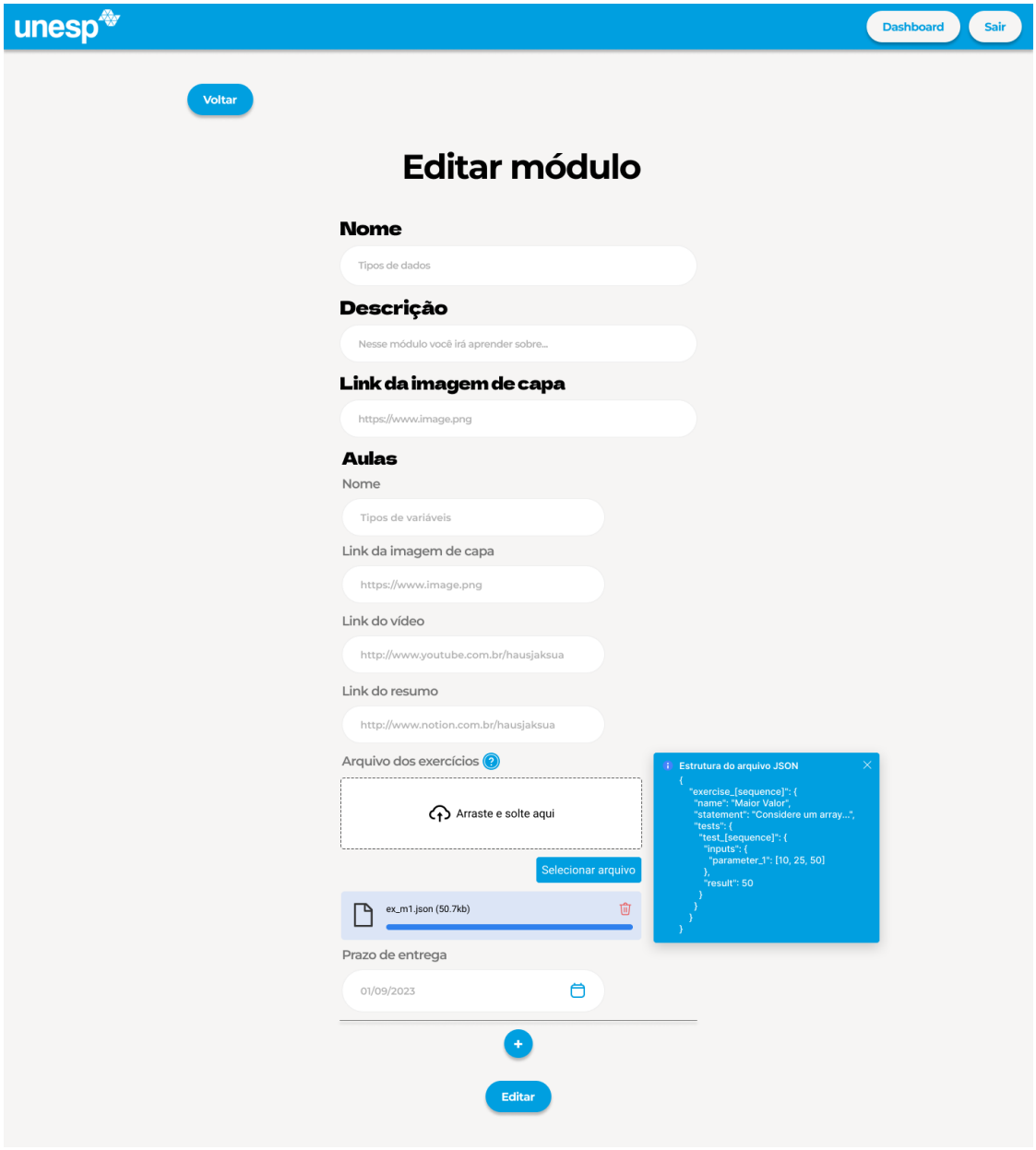


Fonte: Autoria própria.

Na Figura 7, alguns elementos visuais relacionados ao campo “Arquivo dos exercícios”, como o *card* “Estrutura do Arquivo JSON”, são abordados com mais detalhes na seção 3.2.3, referente à definição das regras de negócio.

De volta à tela de módulos (Figura 6), ao clicar na opção “Editar módulo”, o administrador é direcionado para a página de edição, a qual possui estrutura semelhante à de cadastro, diferenciando-se apenas pelo rótulo do botão no final da página. Assim como a anterior, essa página foi elaborada para ser acessível somente ao administrador e seu *layout* é mostrado na Figura 8.

Figura 8 - *Layout* da página de edição de módulo, acessível somente ao administrador

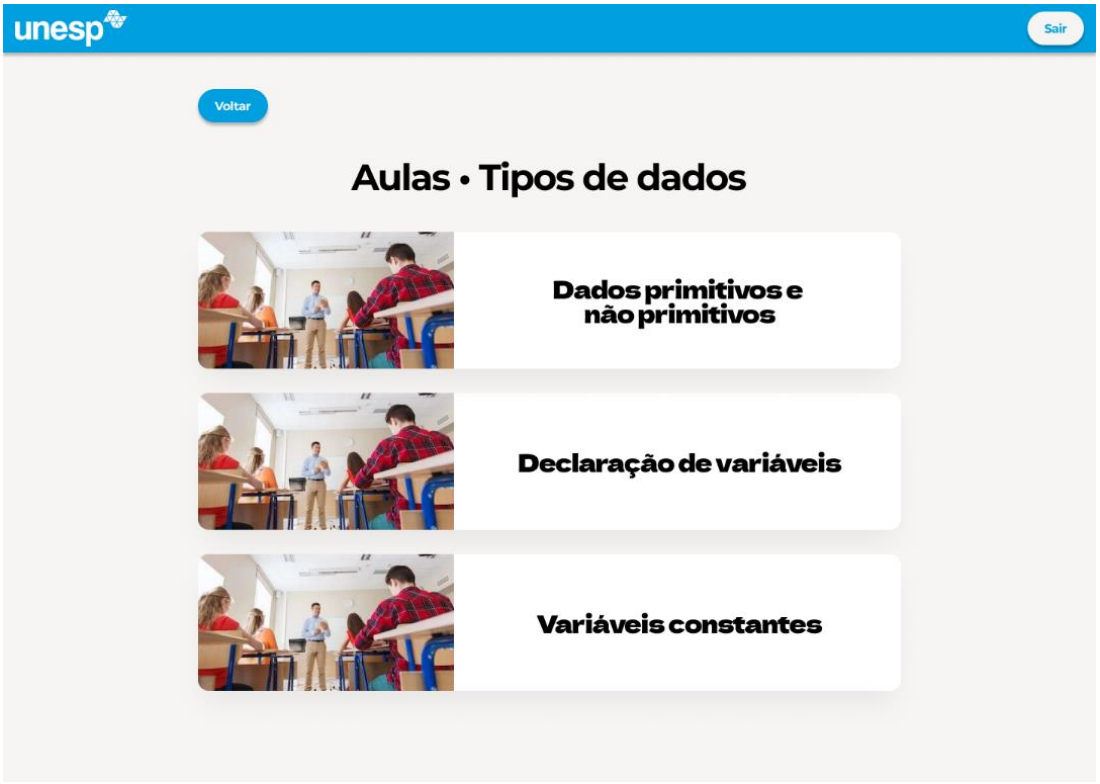


Fonte: Autoria própria.

3.2.2.4 Página de aula

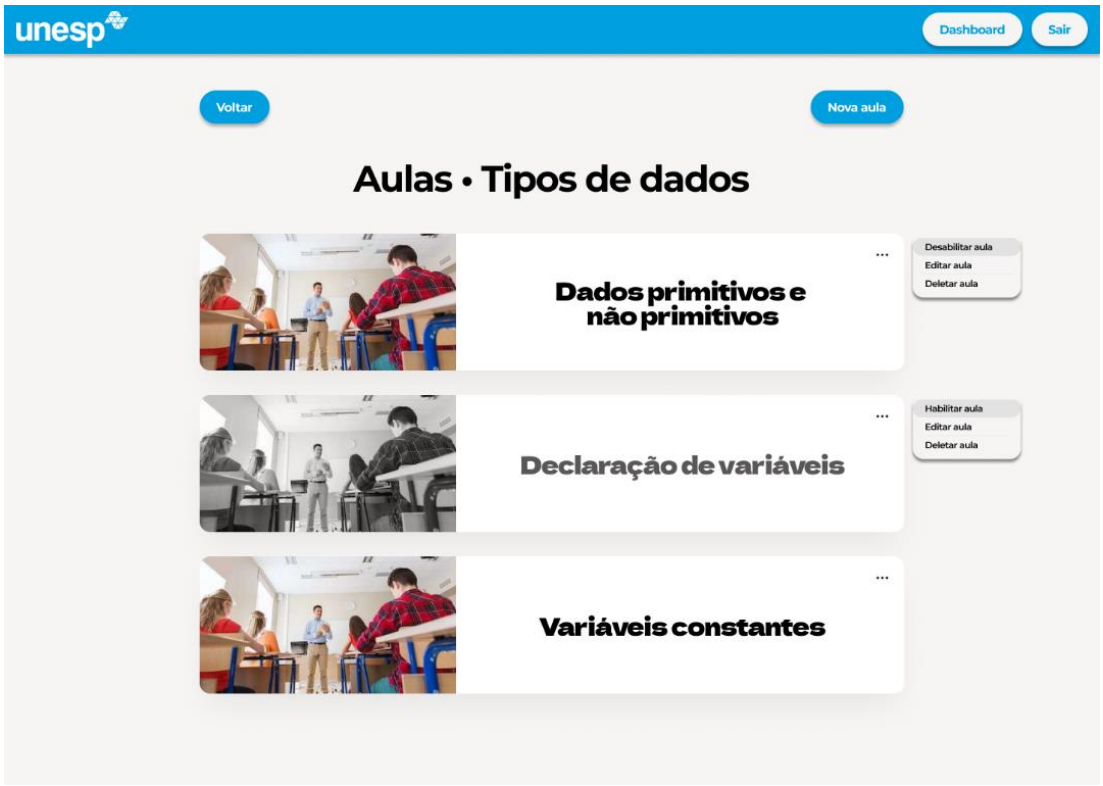
Ao clicar no *card* de um módulo, o usuário é encaminhado para a página onde são exibidas todas as aulas do módulo em questão, também organizadas em *cards* com informações pertinentes. No caso do administrador, é exibido também o botão “Nova Aula” e cada *card* possui um menu *meatballs* com as opções “Desabilitar/habilitar aula”, “Editar aula” e “Deletar aula”. Os *layouts* da página de aulas projetados para o usuário comum e o administrador são mostrados na Figura 9 e na Figura 10, respectivamente.

Figura 9 - *Layout* da página de aulas projetada para o usuário comum



Fonte: Autoria própria.

Figura 10 - *Layout* da página de aulas projetada para o usuário administrador

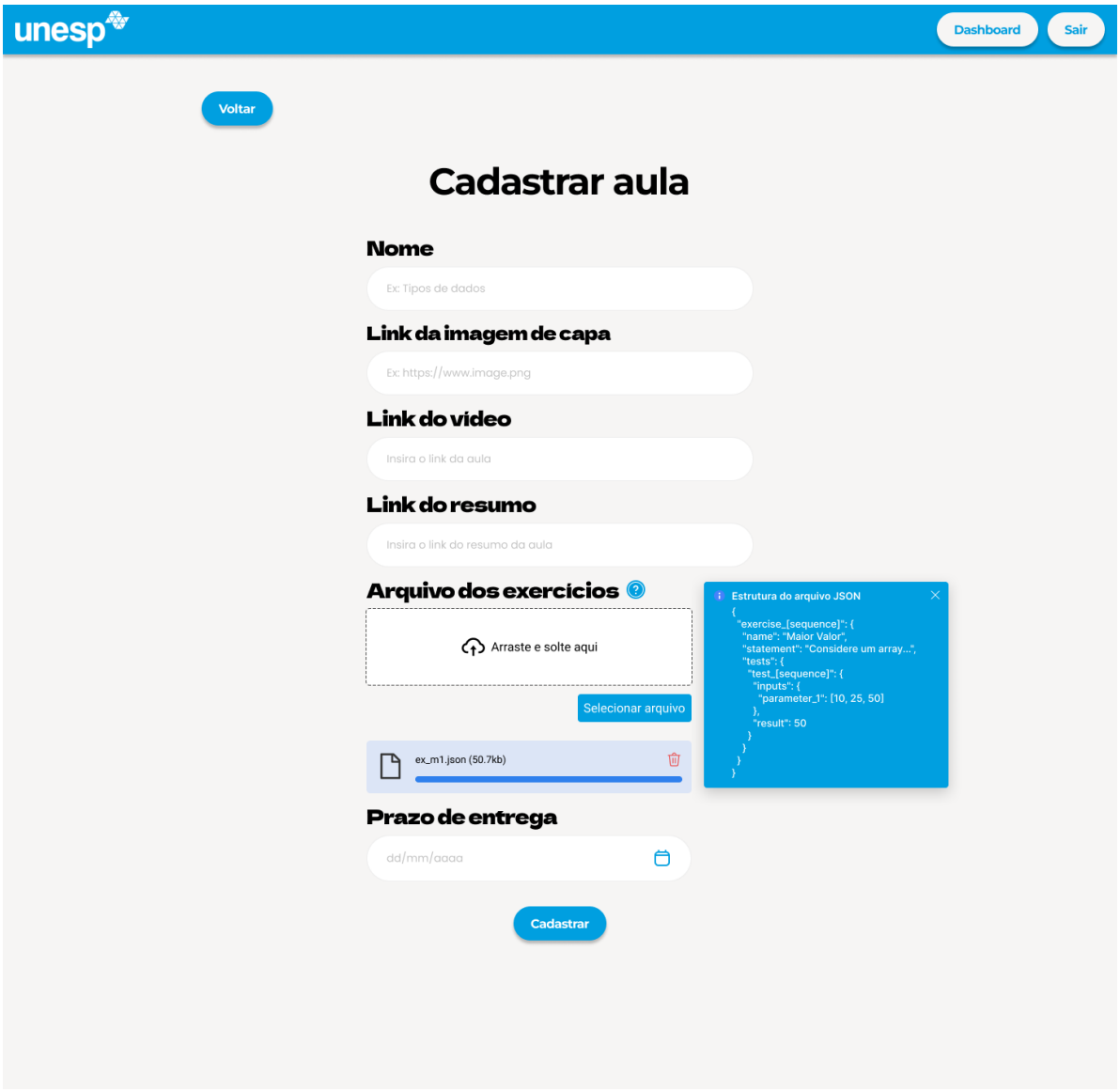


Fonte: Autoria própria.

3.2.2.5 Páginas de cadastro e edição de aula

Ao clicar no botão "Nova aula", o usuário é direcionado para a página de cadastro de aula, a qual possui uma estrutura similar à seção “Aulas” da tela de cadastro de módulo. Essa página também foi elaborada para ser acessível somente ao administrador e seu *layout* é mostrado na Figura 11.

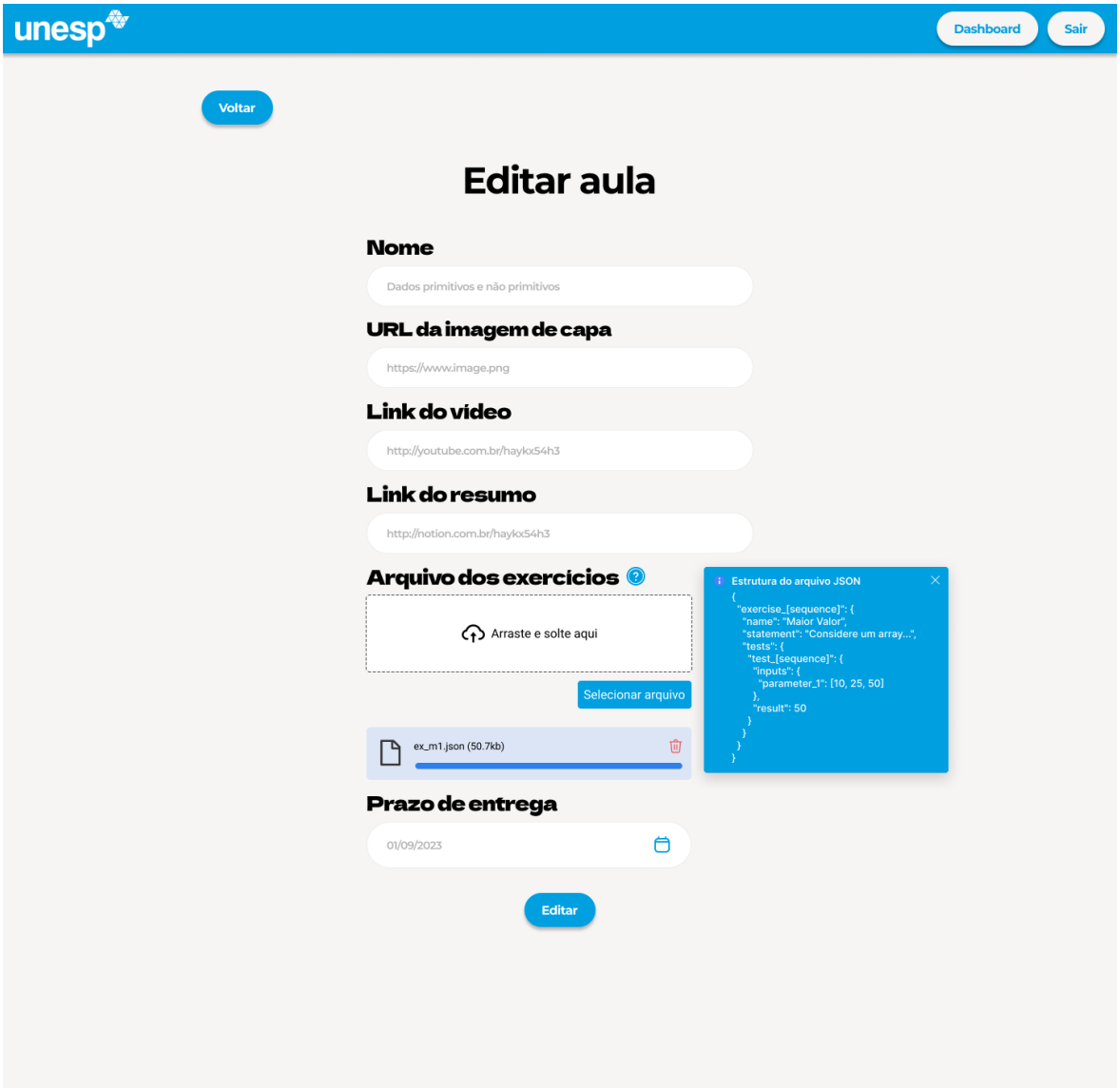
Figura 11 - *Layout* da página de cadastro de aula, acessível somente ao administrador



Fonte: Autoria própria.

Considerando o contexto da página de aulas, a opção “Editar aula” redireciona o usuário para a página de edição de aula, com estrutura semelhante à de cadastro, mudando apenas o rótulo do botão no final do formulário. Da mesma forma que a anterior, ela também foi projetada de forma a ser acessível somente ao administrador e seu *layout* é mostrado na Figura 12.

Figura 12 - *Layout* da página de edição de aula, acessível somente ao administrador



Fonte: Autoria própria.

3.2.2.6 Páginas de videoaula e resumo

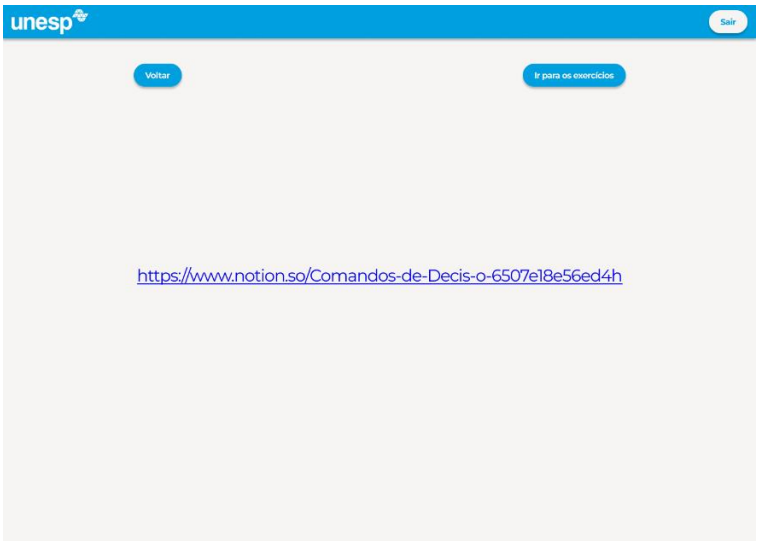
Cada *card* de aula possui um *link* que redireciona o usuário para a página da videoaula correspondente, exibindo um *player* de vídeo do YouTube e botões para acessar o resumo da aula, os exercícios ou voltar para a página anterior. O *layout* da página de videoaula é apresentado na Figura 13.

Figura 13 - *Layout* da página de videoaula

Fonte: Autoria própria.

Já a página de resumo da aula apresenta o *link* deste no centro, além de botões para retornar à videoaula ou acessar os exercícios, conforme mostra a Figura 14.

Figura 14 - *Layout* da página de resumo

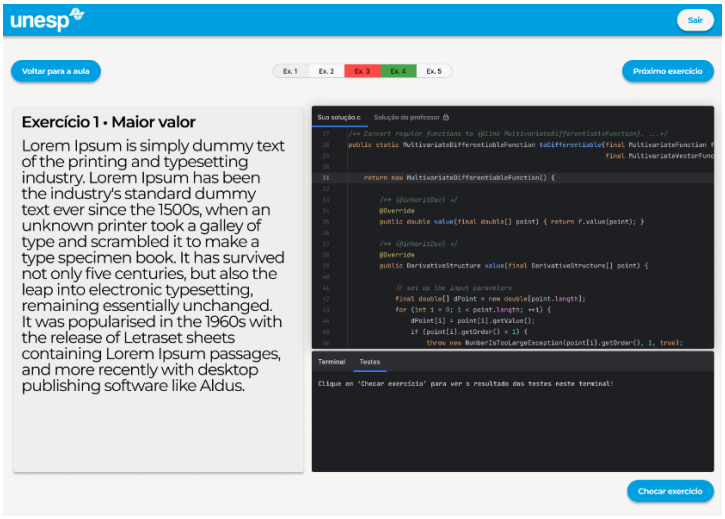


Fonte: Autoria própria.

3.2.2.7 Página de exercícios

Na página de exercícios, são exibidos o nome, o número e o enunciado do exercício do lado esquerdo. Já no lado direito, há um editor de código e um terminal de comando. Na parte superior são exibidos botões para voltar à videoaula e navegar pelos exercícios e, no canto inferior, um botão para compilar (checar) o exercício. O *layout* da página de exercícios é apresentado na Figura 15.

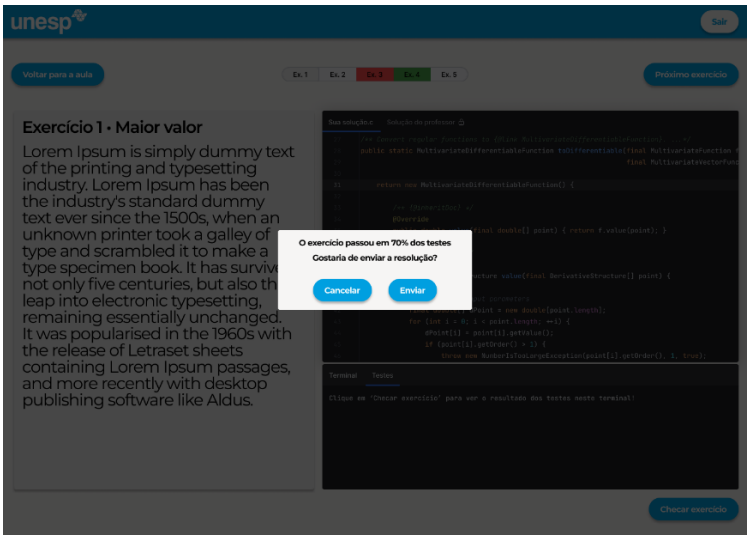
Figura 15 - *Layout* da página de exercícios



Fonte: Autoria própria.

Após clicar no botão “Checar exercício”, é exibido um *modal* (janela que fica sobreposta ao conteúdo principal de uma página da *web*) informando a porcentagem de acertos dos testes realizados no exercício (Figura 16).

Figura 16 - *Layout* do modal da página de exercícios

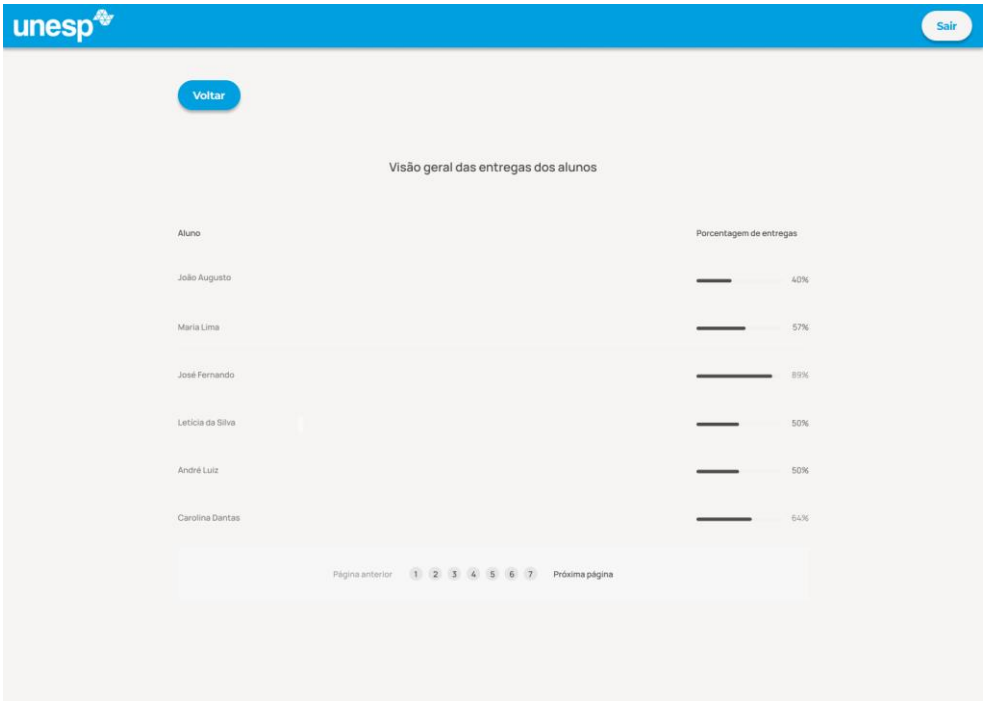


Fonte: Autoria própria.

3.2.2.8 Páginas de dashboard e dashboard do aluno

Por fim, o botão “Dashboard” do cabeçalho encaminha o administrador para a página de *dashboard.* Essa página também foi elaborada para ser acessível somente ao administrador e nela são listados os nomes dos alunos e suas respectivas porcentagens de entregas, conforme exibe a Figura 17.

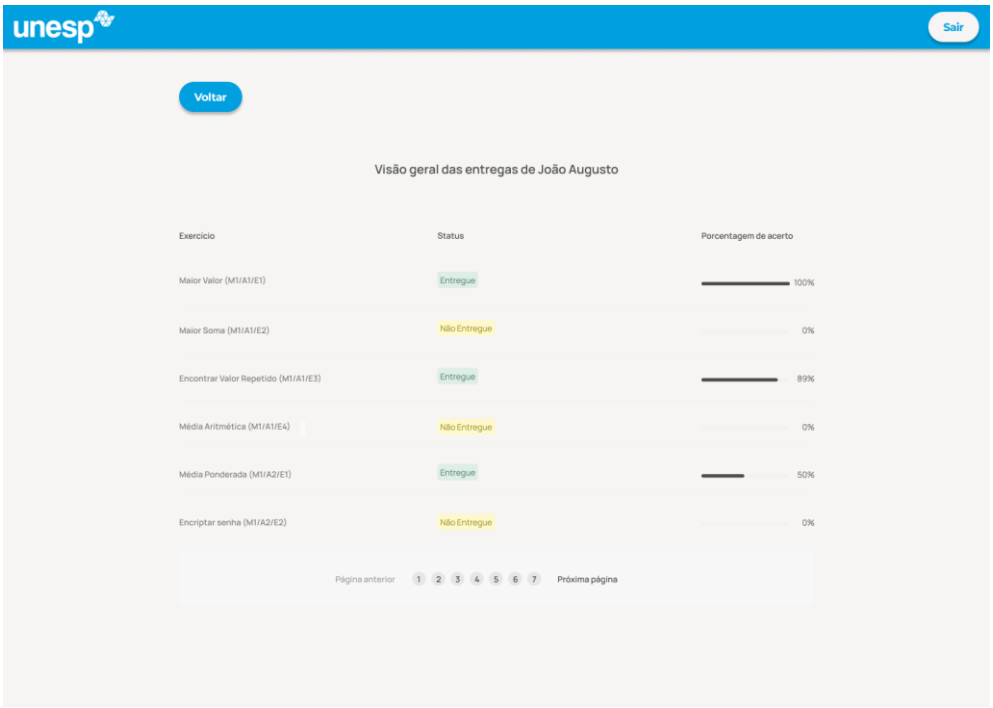
Figura 17 - *Layout* da página de *dashboard*, acessível somente ao administrador



Fonte: Autoria própria.

Ao clicar no nome de um discente, o usuário é direcionado para a página *dashboard* do aluno, também acessível somente ao administrador e contendo uma lista com o nome de cada exercício, o *status* e a porcentagem de acerto do mesmo (Figura 18).

Figura 18 - *Layout* da página de *dashboard* do aluno (acessível somente ao administrador)



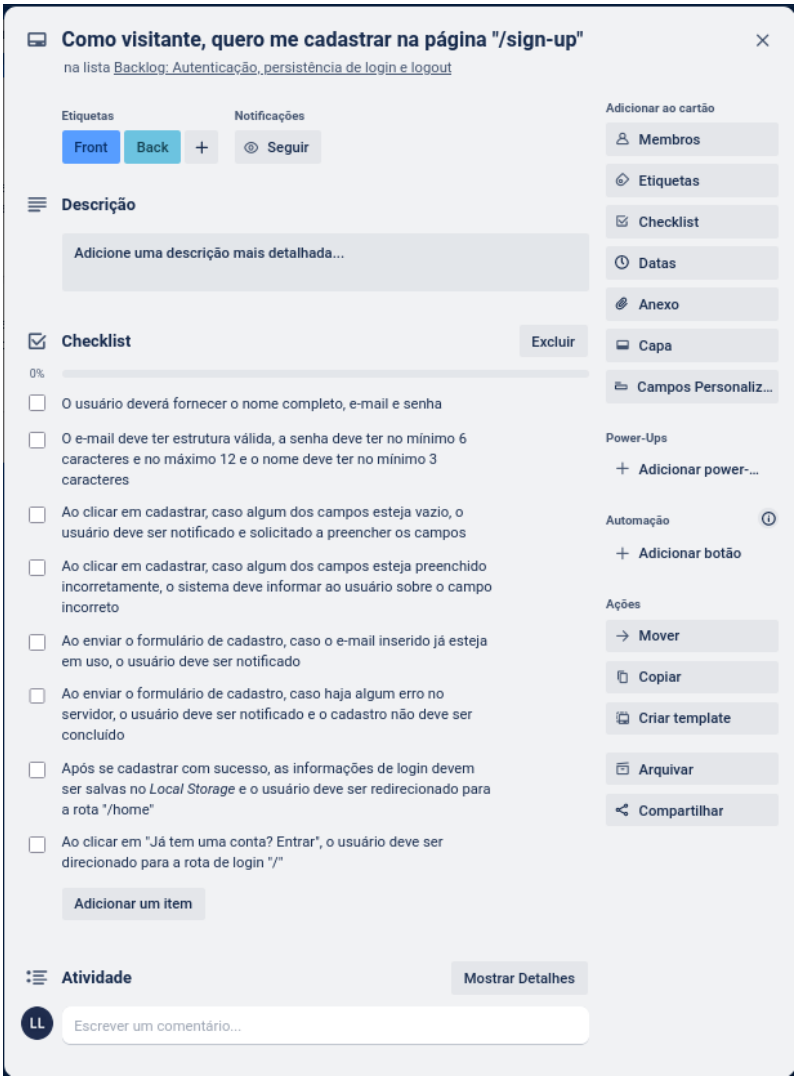
Fonte: Autoria própria.

3.2.3 Definição das regras de negócio

A estruturação das regras de negócio da aplicação *web* foi concebida visando garantir um fluxo consistente ao longo de toda a sua utilização. Para documentar tais regras, optou-se pela utilização da ferramenta Trello, que permitiu o registro delas no formato de tarefas relacionadas a cada página e/ou funcionalidade do sistema, englobando tanto o *back-end* quanto o *front-end*.

Foram criadas listas de tarefas segmentadas por funcionalidade da aplicação. Cada tarefa, representada por um cartão (elemento do Trello), incluiu um título conciso descrevendo sua natureza, uma *checklist* detalhando etapas menores e etiquetas "Back" ou "Front", para indicar em qual lado da aplicação (servidor ou cliente) a tarefa deveria ser desenvolvida. Um exemplo de tarefa no Trello contendo as regras de negócio de uma determinada funcionalidade é exibido na Figura 19. Já a visão geral do quadro do Trello pode ser consultada no apêndice A.

Figura 19 - Exemplo de tarefa contendo regras de negócio no Trello



Fonte: Autoria própria.

A seguir, são detalhadas as regras de negócio relacionadas a cada funcionalidade da aplicação.

3.2.3.1 Autenticação do usuário

Foi estabelecido que o usuário deve fornecer seu nome completo, *e-mail* e senha para se cadastrar na plataforma. Caso ele tente enviar campos em branco, o sistema deve exibir um alerta solicitando o preenchimento adequado. Da mesma forma, caso haja erro de preenchimento, como um *e-mail* inválido, senha fora do padrão (no mínimo 6 caracteres e no máximo 12) ou nome com estrutura inválida (no mínimo 3 caracteres), o sistema deve informar ao usuário sobre o campo incorreto.

Além disso, se o *e-mail* fornecido já estiver em uso, deve ser exibido um alerta ao usuário. Em caso de erro no servidor, o usuário deve ser notificado e o cadastro não deve ser efetuado. Após um cadastro bem-sucedido, o sistema deve realizar o *login* automaticamente e redirecionar o usuário para a página de módulos, armazenando as informações de *login* no *Local Storage* (forma de armazenamento de dados persistente disponível para aplicativos *web*) do navegador. Por fim, foi determinado que o usuário pode navegar da página de cadastro para a de *login* por meio de um *link*.

Já na página de *login*, foi definido que o usuário deve fornecer seu *e-mail* e senha para acessar a plataforma. Da mesma forma que na página de cadastro, caso seja enviado algum campo em branco ou haja erro de preenchimento, o usuário deve ser notificado e o *login* não deve ser efetuado. Em caso de *e-mail* e/ou senha incorretos, o sistema deve exibir um alerta indicando a invalidez das credenciais. Após um *login* bem-sucedido, o usuário deve ser redirecionado para a página de módulos, com as informações de *login* armazenadas no *Local Storage* e, por último, o usuário deve poder navegar da página de *login* para a de cadastro por meio de um *link*.

Por fim, ao fechar e reabrir o navegador, a aplicação deve manter o usuário logado, utilizando as credenciais armazenadas no *Local Storage*. Para efetuar *logout*, o usuário deve selecionar o botão "Sair" no cabeçalho, o que deve resultar na remoção das credenciais do *Local Storage* e no redirecionamento automático para a página de *login*.

3.2.3.2 Módulos

Na página de módulos, o usuário administrador deve visualizar todos os módulos cadastrados, independentemente de estarem habilitados ou não. Já o usuário comum deve visualizar apenas os módulos habilitados. Cada módulo deve apresentar um nome, uma imagem e uma breve descrição.

O usuário administrador deve visualizar o menu *meatballs* no *card* de cada módulo, o qual deve exibir as opções "Desabilitar/habilitar módulo", "Editar módulo" e "Deletar módulo".

Ao desabilitar um módulo, deve haver alguma indicação visual sobre seu *card* que indique o seu *status* e*,* caso um módulo esteja desabilitado, este não deve ser exibido para o usuário comum.

Ao editar um módulo, o administrador deve ser redirecionado para a página de edição de módulo. Já ao deletar um módulo, este deve ser removido da página de módulos, tanto para o usuário administrador quanto para o comum.

O administrador deve visualizar na página de módulos o botão "Cadastrar módulo". Ao clicar nesse botão, ele deve ser direcionado para a página de cadastro de módulo.

Por fim, ao clicar em um módulo, o usuário deve ser redirecionado para a página das aulas referentes ao mesmo.

3.2.3.3 Cadastro e edição de módulo

Nas páginas de criação e edição de módulo, foi definido que o campo “Link do Vídeo” deve receber um *link* do YouTube, e o “Link do Resumo”, um *link* do Notion.

Já no campo "Arquivo dos exercícios" deve ser possível anexar somente um arquivo no formato JSON e ao lado do título deste campo deve haver um ícone que, ao ser clicado, exiba um *card* contendo informações de como deve ser a estrutura do JSON. Ademais, após anexar um arquivo, suas informações, como nome e tamanho do arquivo, devem ser exibidas em um elemento visual. Neste elemento, deve haver algum ícone perto das informações do arquivo que, ao ser clicado, limpe o campo do arquivo e, consequentemente, “esconda” o elemento.

Também foi estabelecido que no formulário de cadastro de módulo, os campos devem ser obrigatórios e inicializados vazios, enquanto que no de edição, os campos devem ser opcionais

e inicializados com as informações salvas no banco de dados. Além disso, no formulário de edição, o botão "+" ao final da seção "Aulas" não deve ser exibido.

No formulário de cadastro, ao clicar no botão "+" ao final da seção "Aulas", deve ser exibida uma nova seção com todos os campos em branco.

Por fim, ao clicar no botão "Cadastrar" ou “Editar”, as informações devem ser enviadas para a API e o administrador deve ser direcionado para a página de módulos, onde o novo módulo ou o módulo editado deve ser exibido.

3.2.3.4 Aulas

Assim como foi definido para os módulos, na página de aulas, o usuário administrador deve visualizar todas as aulas cadastradas, independente de estarem habilitadas ou não. Já o usuário comum deve visualizar apenas as aulas habilitadas. Cada aula deve apresentar um nome e uma imagem.

O administrador deve visualizar o menu *meatballs* no *card* de aula, o qual deve exibir as opções "Desabilitar/habilitar aula", "Editar aula" e "Deletar aula".

Ao desabilitar uma aula, deve haver alguma indicação visual sobre o seu *card* que indique o seu *status*. Caso uma aula esteja desabilitada, esta não deve ser exibida para o usuário comum.

Ao editar uma aula, o administrador deve ser redirecionado para a página de edição de aula. Já ao deletar uma aula, esta deve ser removida da página de aulas, tanto para o usuário administrador quanto para o comum.

O administrador deve visualizar na página de aulas o botão "Cadastrar aula". Ao clicar nesse botão, ele deve ser direcionado para a página de cadastro de aula.

Por fim, ao clicar em uma aula, o administrador deve ser redirecionado para a página da videoaula referente à mesma.

3.2.3.5 Cadastro e edição de aula

As regras de negócio relacionadas aos campos da seção “Aulas” nas páginas de cadastro e edição de módulo também se aplicam às de cadastro e edição de aula.

Dessa forma, foi estabelecido que no formulário de cadastro de aula, os campos devem ser obrigatórios e inicializados vazios, enquanto que no de edição, os campos devem ser opcionais e inicializados com as informações salvas no banco de dados.

Ao clicar no botão "Cadastrar" ou “Editar”, as informações devem ser enviadas para a API e o administrador deve ser direcionado para a página de aulas, onde a nova aula ou a aula editada deve ser exibida.

3.2.3.6 Vídeo

Na página de vídeo de uma determina aula, o usuário deve visualizar um *player* do YouTube contendo a videoaula. Ao clicar nele, o usuário deve ser capaz de visualizar o vídeo com todas as funcionalidades características de um *player* do YouTube.

3.2.3.7 Resumo

Ao acessar a página de resumo de uma determinada aula, o usuário deve visualizar um *link* do Notion. Ao clicar nesse *link*, o usuário deve ser redirecionado para o resumo da aula em questão.

3.2.3.8 Exercícios e envio de resoluções

Na página de exercícios, o usuário deve ser capaz de visualizar o nome, o enunciado e o prazo de entrega do exercício, além de um editor de código e um terminal de comando.

No editor de código, deve ser exibida inicialmente a estrutura do corpo da função, contendo o nome, o tipo de dado que ela recebe e o tipo de dado que ela retorna.

Ao clicar no botão “Checar exercício”, os testes devem ser executados na resolução fornecida pelo discente. Em seguida, deve ser exibido um modal, contendo a porcentagem de acerto dos exercícios e opções para o discente escolher se deseja enviar ou não a resolução.

Caso o aluno opte por não enviar a resolução, a porcentagem de acerto não deve ser persistida no banco de dados, nem enviada à API, e o *status* do exercício deve ser definido como “Não enviado”. Por outro lado, se o aluno optar por enviar, a resolução deve ser persistida, o

status do exercício deve ser alterado para “Enviado” e os resultados dos testes devem ser exibidos no terminal.

Se o aluno sair da página e a resolução tiver sido enviada, ao retornar, a mesma deve ser exibida no editor de código, bem como a porcentagem de acerto e o *status* do exercício na página.

O aluno deve poder enviar uma nova resolução enquanto o prazo de entrega não tiver expirado, independentemente de já ter enviado uma resolução anteriormente e de ter passado em todos os testes ou não. Em contrapartida, caso o prazo de entrega tenha expirado, o aluno não deve poder enviar uma nova resolução.

Por fim, o aluno deve poder navegar entre os exercícios por meio de um *Segment Control* (componente que permite aos usuários selecionar uma aba dentre um conjunto de abas mutuamente exclusivas), o qual também deve apresentar um elemento visual que identifique quais exercícios foram enviados.

3.2.3.9 Dashboard

O botão “Dashboard” deve ser exibido no cabeçalho apenas para o administrador e deve ser acessível de qualquer página que o usuário esteja.

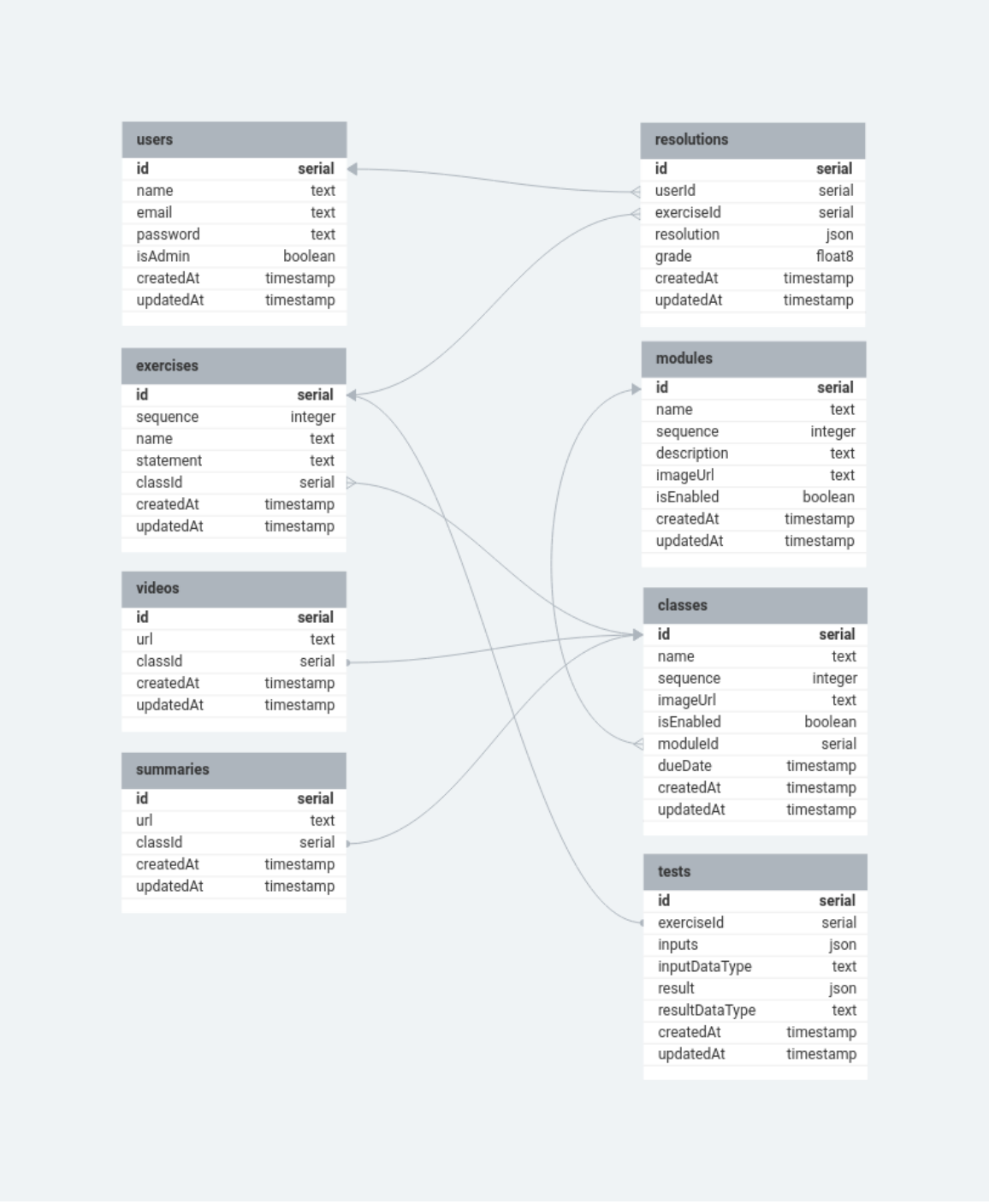
Ao clicar no botão “Dashboard”, o usuário deve ser redirecionado para a página de *dashboard*, a qual deve apresentar uma visão geral das entregas dos alunos, como o nome e a porcentagem de entregas de cada um. A porcentagem de entregas deve se referir às entregas em si, independentemente da porcentagem de acerto das resoluções enviadas.

Ao clicar no nome de um dos discentes, o administrador deve ser redirecionado para a página de *dashboard* do aluno. Essa página deve conter uma lista com a identificação de cada exercício, contendo o seguinte padrão de nomenclatura: nome do exercício (sequência do módulo/sequência da aula/sequência do exercício). Ademais, deve exibir o *status* (“Entregue” ou “Não Entregue”) e a porcentagem de acerto de cada exercício.

3.2.4 Estruturação das tabelas do banco de dados

O banco de dados foi estruturado em 8 tabelas: *users*, *modules*, *classes*, *videos*, *summaries*, *exercises*, *tests* e *resolutions*. O diagrama entidade-relacionamento do banco de dados pode ser visualizado na Figura 20.

Figura 20 - Diagrama entidade-relacionamento do banco de dados



Fonte: Autoria própria.

Na tabela “users”, o campo “email” deve ser único. Esta tabela possui um relacionamento de 1:N (lê-se, de um para N) com a tabela “resolutions”, utilizando os campos “id” da tabela “users” e “userId” da tabela “resolutions”, uma vez que cada usuário pode ter várias resoluções.

Já na tabela “modules”, os campos “name” e “sequence” devem ser únicos, pois não deve haver repetição de nomes e sequências de módulos. Além disso, há um relacionamento de 1:N com a tabela “classes”, utilizando os campos “id” da tabela “modules” e “moduleId” da tabela “classes”, pois um módulo poder conter múltiplas aulas.

Quanto à tabela “classes”, as combinações entre os campos “name” e “moduleId”, e entre “sequence” e “moduleId” devem ser únicas, garantindo que não haja repetição de nomes e sequências de aulas em cada módulo. Essa tabela possui dois relacionamentos de 1:1 (lê-se, de um para um) com as tabelas “videos” e “summaries”, utilizando os campos “id” na tabela “classes” e “classId” nas tabelas “videos” e “summaries”, pois cada aula deve ter apenas um vídeo e um resumo associados a ela. Além disso, há um relacionamento de 1:N com a tabela “exercises”, utilizando os campos “id” na tabela “classes” e “classId” na tabela “exercises”, uma vez que cada aula pode ter múltiplos exercícios associados.

As tabelas “videos” e “summaries” possuem uma estrutura semelhante, com a restrição de que o campo “classId” deve ser único em ambas, garantindo assim que cada aula tenha apenas um vídeo e um resumo associado a ela.

Na tabela “exercises”, as combinações entre os campos “name” e “classId”, e entre “sequence” e “classId” devem ser únicas, assegurando que cada aula contenha apenas exercícios com nomes e sequências distintos. Além do relacionamento com a tabela “classes” mencionado anteriormente, a tabela “exercises” possui dois relacionamentos de 1:N com as tabelas “tests” e “resolutions”, utilizando os campos “id” na tabela “exercises” e “exerciseId” nas tabelas “tests” e “resolutions”, visto que cada exercício pode ter múltiplos testes e resoluções associadas a ele.

A tabela “tests” possui campos para armazenar valores do tipo JSON, como "inputs" e "result", que representam as entradas e saídas esperadas dos testes, respectivamente. Além disso, há campos do tipo *string*, como "inputDataType" e "resultDataType", que armazenam os tipos de dados das entradas e saídas dos testes, respectivamente, auxiliando assim na lógica de geração dos testes no *front-end*.

Por fim, a tabela “resolutions” possui uma restrição que garante que a combinação dos campos "userId" e "exerciseId" seja sempre única, certificando que cada aluno tenha apenas

uma resolução associada a ele por exercício. O campo "resolution" da tabela armazena a resolução em si, representada no formato JSON, enquanto o campo "grade", do tipo *float*, armazena a nota da resolução.

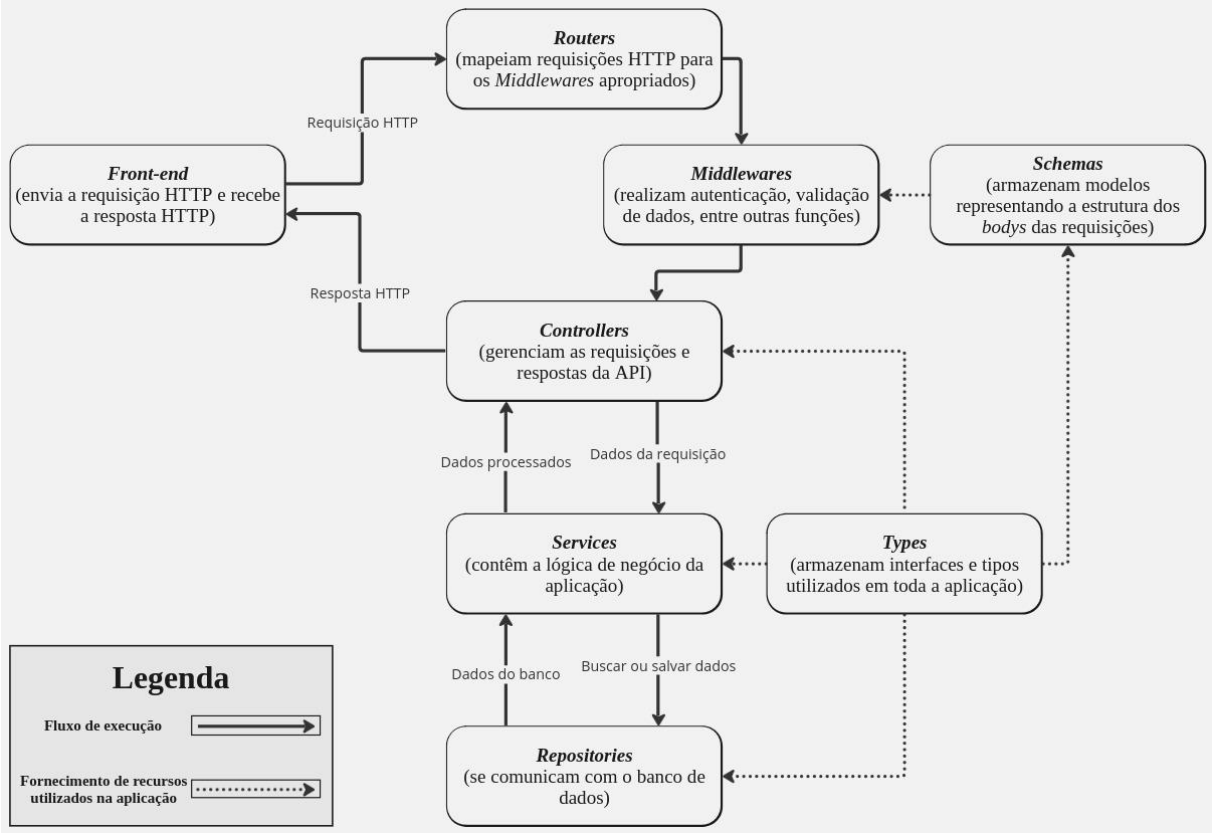
3.2.5 Arquiteturas

As arquiteturas do *back-end* e do *front-end* da aplicação *web* foram cuidadosamente estruturadas para garantir uma organização eficiente e escalável, conforme é explicitado a seguir.

3.2.5.1 Arquitetura do back-end

No *back-end*, adotou-se uma abordagem baseada na arquitetura *Model-View-Controller* (MVC), cuja representação pode ser vista no diagrama da Figura 21.

Figura 21 - Diagrama representando a arquitetura do back-end da aplicação



Fonte: Autoria própria.

No contexto apresentado na Figura 21, os *Routers* constituem a primeira camada, responsável por lidar com os *endpoints* (URLs específicas em um servidor que são acessadas por programas ou sistemas externos para interagir com uma API) e direcionar as requisições HTTP. Após a chegada da requisição nos *Routers*, estas são encaminhadas para os *Middlewares*, os quais realizam diversas validações, como autenticação de usuários e validação dos *bodies* (parte da mensagem HTTP que contém os dados enviados pelo cliente para o servidor) das requisições, entre outras.

Uma vez validadas, as informações relevantes são passadas para os *Controllers*, responsáveis por gerenciar as requisições e respostas da API, e em seguida, encaminhadas por meio destes para a camada dos *Services*. Nessa camada residem todas as regras de negócio da aplicação, sendo responsável por toda a lógica da API. Quando necessário acessar o banco de dados para recuperar ou salvar informações, os *Services* interagem com a camada de *Repositories*, que se comunica diretamente com o banco de dados e retorna os resultados necessários.

Além dessas camadas, no *back-end* também são utilizados diretórios como *Schemas*, que armazenam modelos representando a estrutura dos *bodys* das requisições para serem utilizados nas validações realizadas pelos *Middlewares*. Similarmente, o diretório *Types* armazena interfaces e tipos utilizados em toda a aplicação.

3.2.5.2 Arquitetura do front-end

No *front-end*, optou-se por uma estrutura baseada em diversos diretórios distintos, cada um com uma função específica, sendo eles: *assets*, *components*, *contexts*, *hooks*, *layouts*, *pages*, *services* e *utils*.

O diretório de *assets* armazena os recursos estáticos da aplicação, sendo subdividido em *images* e *styles*. O primeiro contém as imagens usadas na interface, enquanto o segundo, os arquivos CSS relativos aos estilos globais.

O diretório de *components* contém pequenos elementos que compõem as páginas da interface, cada um com sua própria estrutura, estilização e lógica.

Já os diretórios de *contexts* e *hooks* armazenam os contextos e *custom hooks* da aplicação React, respectivamente, oferecendo assim uma maneira de gerenciar e compartilhar estados e lógicas entre componentes.

O diretório *layouts* armazena os estilos utilizados por diversos componentes da interface, proporcionando uma padronização visual.

Em *pages* estão as páginas desenvolvidas, cada uma sendo uma composição dos componentes mencionados anteriormente, também com sua própria estrutura, estilização e lógica.

O diretório *services* é responsável por lidar com as chamadas HTTP, desde o envio até o recebimento, manipulação dos dados e tratamento de erros.

Por fim, o diretório *utils* contém funções auxiliares da aplicação, como a que valida a estrutura do arquivo JSON e a que gera os testes a serem executados nos exercícios.

3.2.6 Usabilidade do usuário

Foram implementadas diversas melhorias na interface da plataforma *web* com o objetivo de aprimorar a experiência do usuário e sua usabilidade.

A primeira delas foi o armazenamento dos dados de autenticação. Após realizar *login* na plataforma, o *token* de autenticação e o nível de acesso do usuário (administrador ou usuário comum) são armazenados no *Local Storage* do navegador. Isso evita a necessidade de autenticação a cada acesso à plataforma. Ao clicar no botão "Sair", essas informações são apagadas do *Local Storage* e o usuário é redirecionado automaticamente para a página de *login*, oferecendo assim a opção de *logout*.

Para o usuário administrador, foi implementado a indicação visual de módulos ou aulas desabilitados. Ao desabilitar um módulo ou aula, o *card* deste apresenta um filtro escuro sobre ele, sinalizando seu estado desabilitado.

Outra melhoria implementada no contexto do usuário administrador foram as informações relativas ao arquivo de exercícios. Nas páginas de cadastro e edição de módulo ou aula, foi adicionado um elemento para exibir informações como nome e tamanho do arquivo de exercícios carregado, a fim de indicar que o carregamento foi concluído com sucesso e qual arquivo foi selecionado. Além disso, adicionou-se um ícone ao lado da seção "Arquivo dos exercícios" para mostrar a estrutura esperada do arquivo JSON, auxiliando na sua estruturação.

Melhorias foram implementadas também em relação ao *feedback* recebido após a execução dos exercícios. Uma vez finalizado a compilação e execução dos testes na página de

exercícios, um modal com fundo escuro informa ao usuário a porcentagem de testes executados com sucesso e oferece as opções de enviar ou não a resolução, dando ao usuário o poder de escolha e o impedindo de executar outras ações na interface enquanto não escolher uma opção. Os testes são exibidos no terminal, sendo que os casos de falha são destacados em vermelho e os de sucesso em verde, facilitando a diferenciação destes por parte do usuário. Além disso, após o envio de um exercício, a aba correspondente no *Segment Control* muda para a cor azul, indicando visualmente que o exercício foi enviado sem que o usuário precise clicar na aba para visualizar seu *status*.

Ao longo da aplicação foram adicionados *loaders* (ícones de carregamento) com fundos escuros durante a execução de requisições HTTP à API, informando o usuário sobre o carregamento em curso e impedindo a execução de outras ações até sua conclusão.

Por fim, foi implementado *feedback* visual em elementos interativos da aplicação. Ao passar o *mouse* sobre elementos como, por exemplo, botões habilitados, foi adicionado um filtro escuro e um ícone diferente ao cursor para indicar sua interatividade. Já botões desativados tiveram sua coloração alterada para indicar seu estado desabilitado.

3.2.7 Compilação do código em C

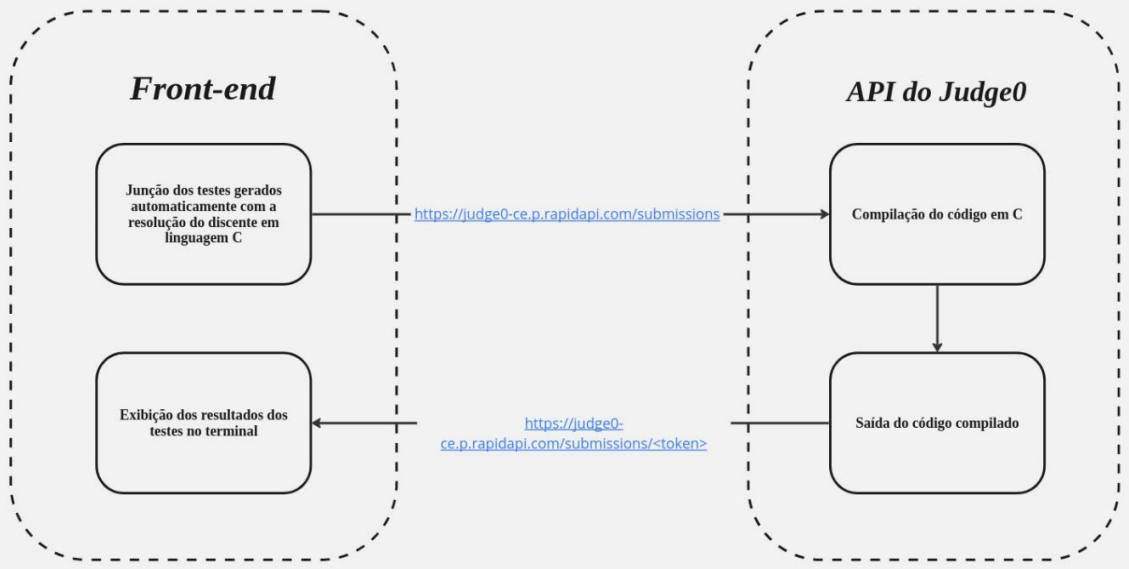
A compilação do código em linguagem C foi realizada por meio da integração com o Judge0, uma API que oferece compiladores para várias linguagens de programação. Para desenvolver essa integração, foi necessário criar uma conta no site do Judge0 e armazenar o *token* gerado como uma variável de ambiente (estrutura que armazena informações de configuração ou parâmetros específicos do ambiente de execução de uma aplicação) no *front-end.*

Então, utilizando o *endpoint* "https://judge0-ce.p.rapidapi.com/submissions", o código em C é enviado para a API do Judge0 juntamente com parâmetros de configuração, como informações do usuário e a linguagem de programação a ser compilada. Por meio do *endpoint* "https://judge0-ce.p.rapidapi.com/submissions/token_gerado_de_forma_automática", a API retorna a saída do código compilado, que é então exibida no terminal da interface.

Já em relação aos testes, ao clicar em "Checar exercício", o código fornecido pelo aluno é agregado a um código em C gerado automaticamente pela interface. Esse código contém a lógica de execução de cada teste e a exibição dos resultados no terminal. O código resultante é

então enviado para a API, que realiza a compilação e retorna os resultados dos testes no formato esperado para exibição no terminal. O fluxo referente à compilação do exercício e execução dos testes é apresentado no diagrama da Figura 22.

Figura 22 - Diagrama do fluxo de compilação do exercício e execução dos testes



Fonte: Autoria própria.

3.2.8 Implementação do processo de validação do protótipo

Após finalizado o desenvolvimento do protótipo, foi feito o seu *deploy* (processo de disponibilizar um *site* na internet) e, em seguida, cadastrados três módulos: “Introdução”, “Operações Aritméticas na Linguagem C” e “Estruturas de Controle na Linguagem C”. No primeiro módulo foram criadas três aulas: uma apresentando a plataforma, outra sobre declaração e atribuição de variáveis e a última sobre comandos de entrada e de saída. Já no segundo módulo foi criada uma aula sobre operações aritméticas e no terceiro, duas, sendo uma sobre comandos de decisão e a outra sobre comandos de repetição. Além disso, em cada aula, foram cadastrados exercícios relacionados ao conteúdo ministrado.

Na sequência, o *link* de *deploy* foi disponibilizado para ex-alunos da disciplina de ICC, os quais foram instruídos a respeito do funcionamento e objetivo do protótipo, bem como orientados a testarem todas as funcionalidades da aplicação e resolverem os exercícios propostos em um período de dez dias.

Passado esse período, foi elaborado e disponibilizado para os ex-alunos um formulário de avaliação do protótipo.

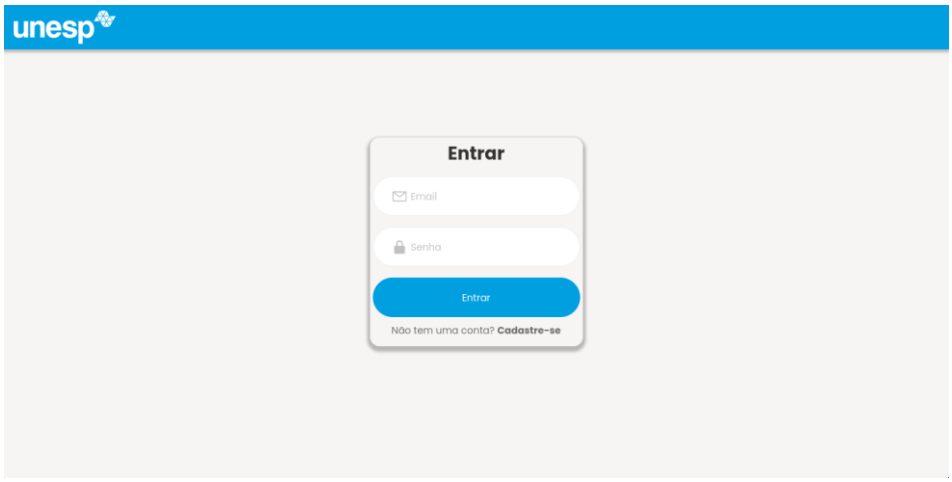
4 RESULTADOS E DISCUSSÕES

4.1 Apresentação da interface do protótipo

A interface do protótipo foi desenvolvida de forma a conter diversas páginas, as quais serão apresentadas nesta seção.

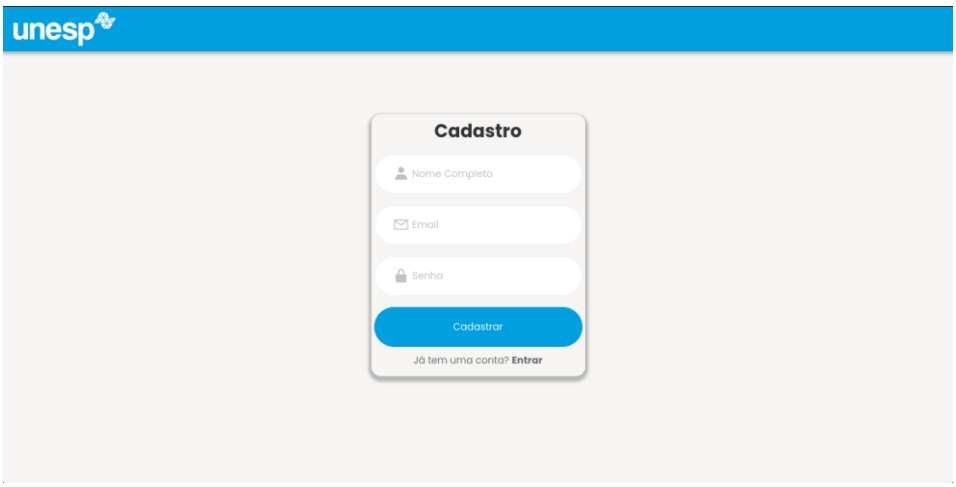
Foi possível realizar o desenvolvimento das páginas de *login* e cadastro em conformidade com as regras de negócio especificadas na seção 3.2.3.1 (Definição das regras de negócio - Autenticação do usuário), conforme mostrado na Figura 23 e na Figura 24.

Figura 23 - Página de *login* desenvolvida para o protótipo



Fonte: Autoria própria.

Figura 24 - Página de cadastro desenvolvida para o protótipo



Fonte: Autoria própria.

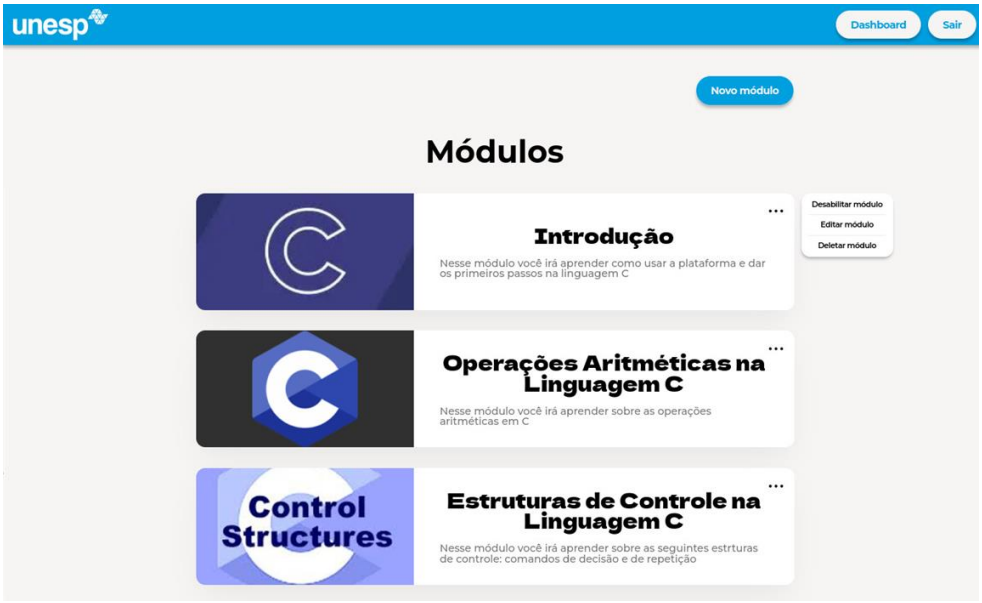
De maneira análoga, a página de módulos foi elaborada contemplando variações no seu *layout* conforme o nível de acesso do usuário (administrador ou usuário comum), conforme

mencionado na seção 3.2.3.2 (Regras de negócio - Módulos). A página de módulos visualizada pelo usuário comum é exibida na Figura 25, enquanto a visualizada pelo administrador é mostrada na Figura 26.

Figura 25 - Visualização que o usuário comum tem da página de módulos desenvolvida para o protótipo

Fonte: Autoria própria.

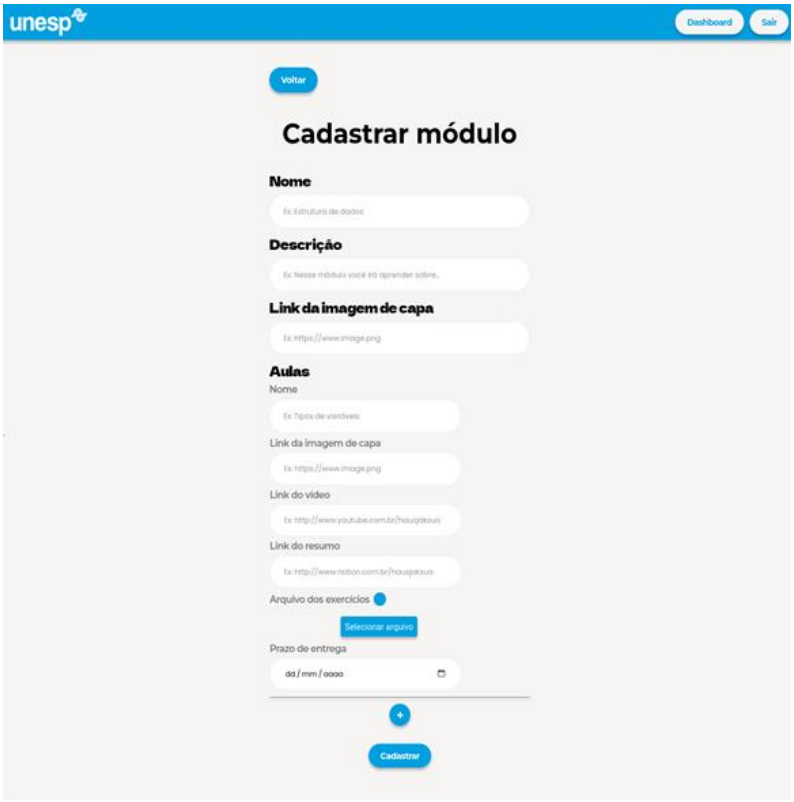
Figura 26 - Visualização que o administrador tem da página de módulos desenvolvida para o protótipo



Fonte: Autoria própria.

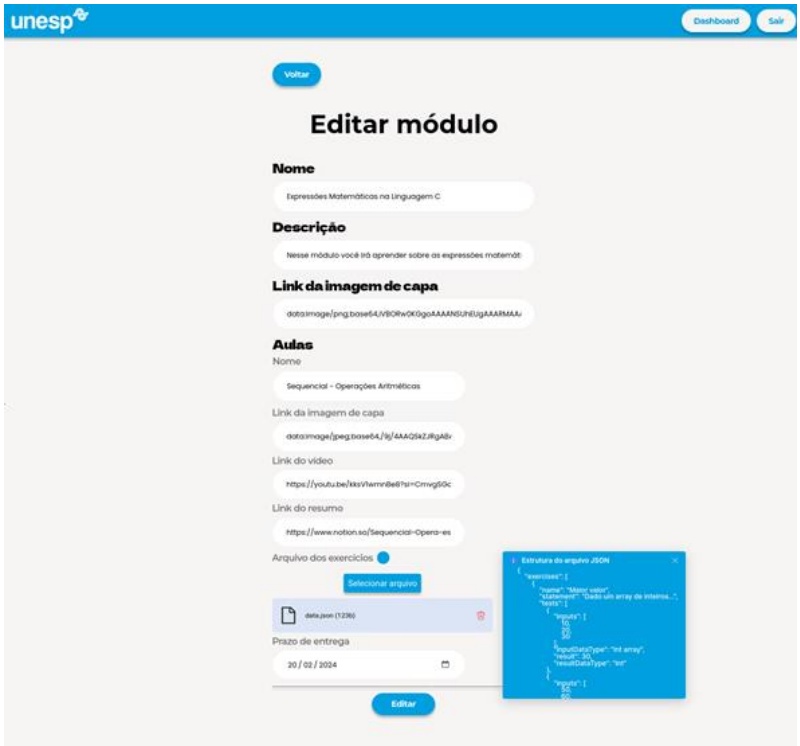
As páginas de cadastro e edição de módulo foram desenvolvidas seguindo as regras de negócio definidas na seção 3.2.3.3 (Regras de negócio - Cadastro e edição de módulo) e são apresentadas na Figura 27 e na Figura 28, respectivamente.

Figura 27 - Página de cadastro de módulo desenvolvida para o protótipo, acessível somente ao administrador



Fonte: Autoria própria.

Figura 28 - Página de edição de módulo desenvolvida para o protótipo, acessível somente ao administrador



Fonte: Autoria própria.

Assim como a página de módulos, a página de aulas foi elaborada considerando variações no *layout*, conforme o nível de acesso do usuário (administrador ou usuário comum), e que foi detalhado na seção 3.2.3.4 (Regras de negócio - Aulas). A página de aulas visualizada pelo usuário comum é mostrada na Figura 29, e a pelo administrador é apresentada na Figura 30.

Figura 29 - Visualização que o usuário comum tem da página de aulas desenvolvida para o protótipo

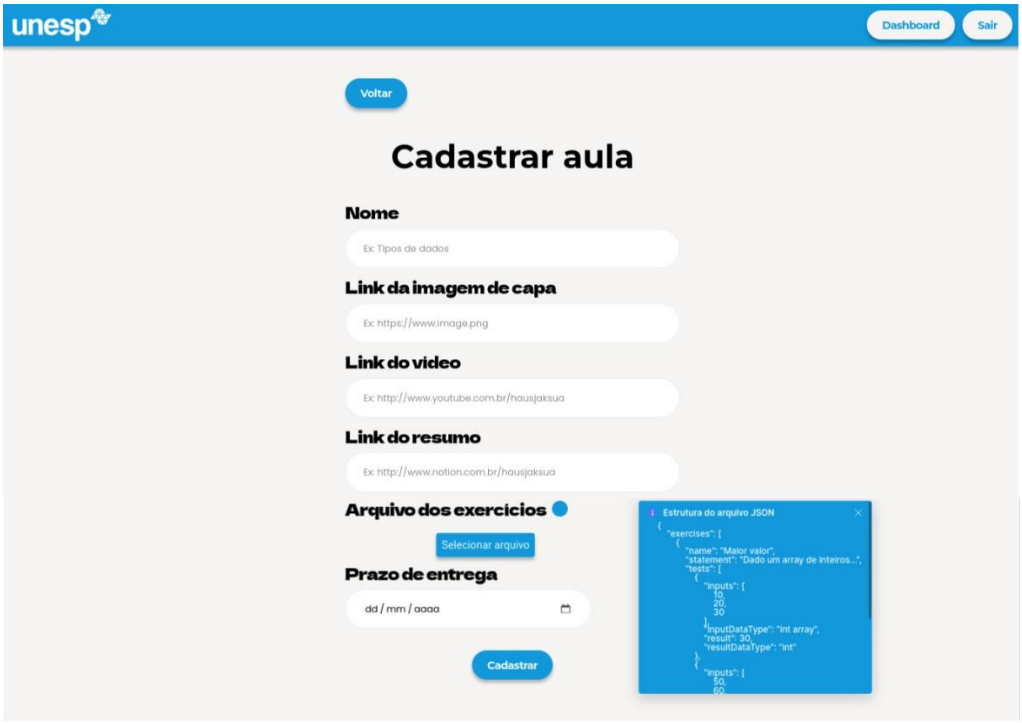
Fonte: Autoria própria.

Figura 30 - Visualização que o administrador tem da página de aulas desenvolvida para o protótipo

Fonte: Autoria própria.

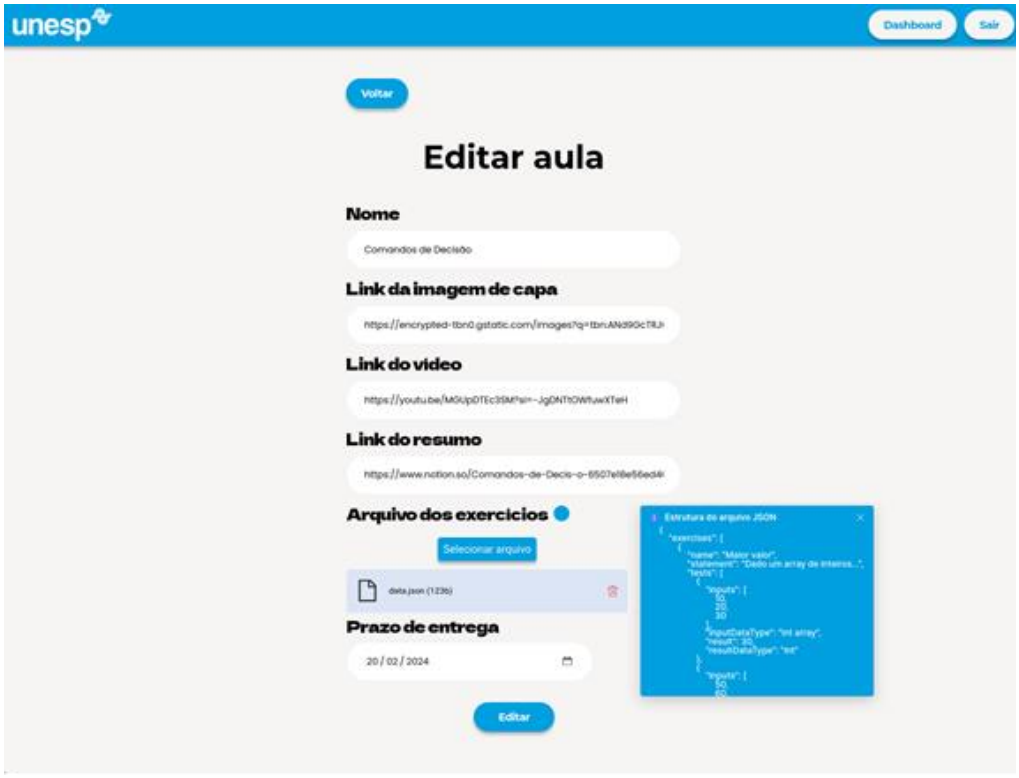
As páginas de cadastro e de edição de aula foram desenvolvidas em conformidade com as regras de negócio definidas na seção 3.2.3.5 (Regras de negócio - Cadastro e edição de aula) e são exibidas, respectivamente, na Figura 31 e na Figura 32.

Figura 31 - Página de cadastro de aula desenvolvida para o protótipo, acessível somente ao administrador



Fonte: Autoria própria.

Figura 32 - Página de edição de aula desenvolvida para o protótipo, acessível somente ao administrador



Fonte: Autoria própria.

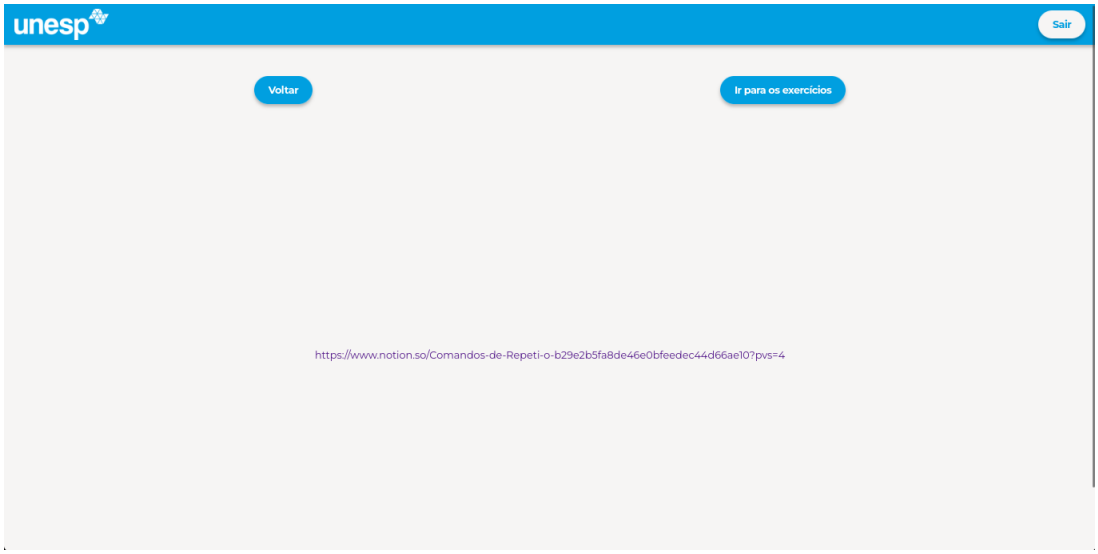
Além disso, foi implementada a página de videoaula em consonância com as regras de negócio estipuladas na seção 3.2.3.6 (Regras de negócio - Videoaula) (Figura 33).

Figura 33 - Página de videoaula desenvolvida para o protótipo

Fonte: Autoria própria.

A página de resumo foi elaborada considerando as regras de negócio descritas na seção 3.2.3.7 (Regras de negócio - Resumo), e é apresentada na Figura 34.

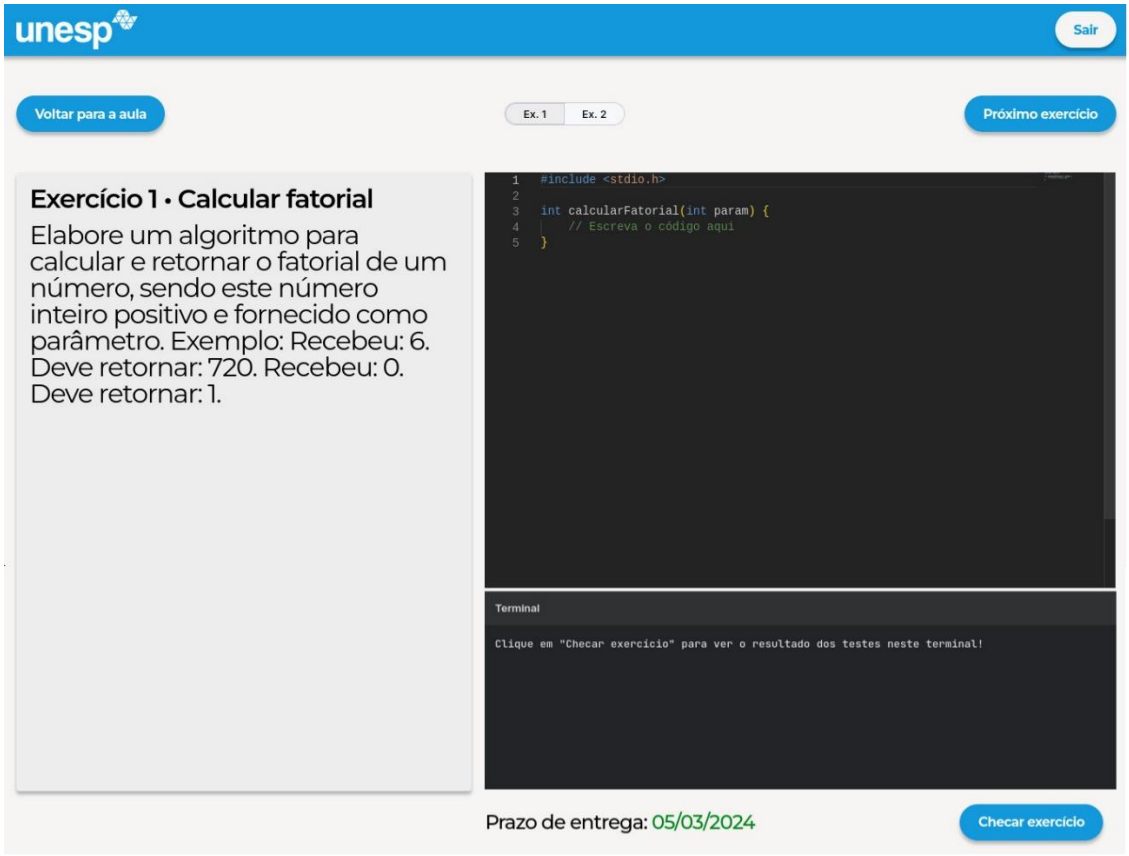
Figura 34 - Página de resumo desenvolvida para o protótipo



Fonte: Autoria própria.

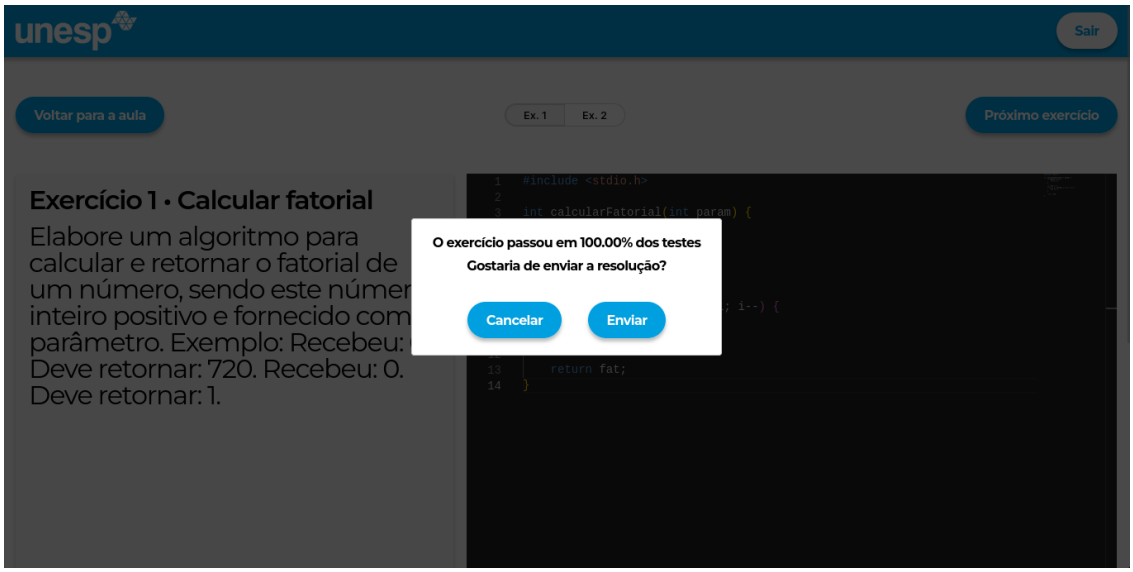
No prosseguimento do fluxo da aplicação, foi criada a página de exercícios, atendendo às regras de negócio descritas na seção 3.2.3.8 (Regras de negócio - Exercícios e envio de resoluções). O estado inicial da página de exercícios pode ser observado na Figura 35, sendo que o modal associado é apresentado na Figura 36.

Figura 35 - Estado inicial da página de exercícios desenvolvida para o protótipo



Fonte: Autoria própria.

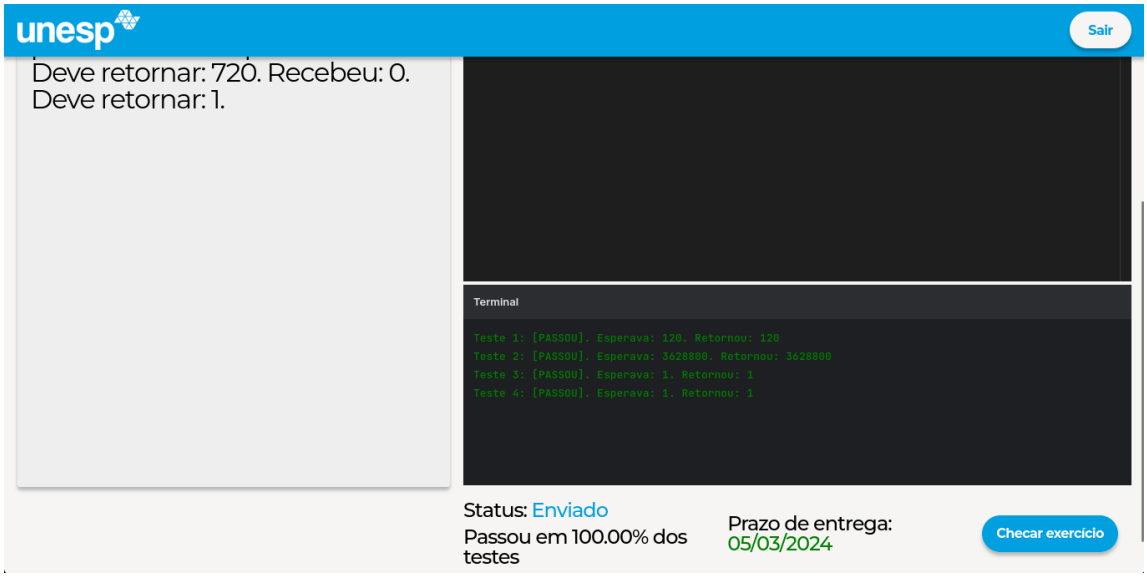
Figura 36 - Modal da página de exercícios desenvolvido para o protótipo



Fonte: Autoria própria.

Adicionalmente, o *layout* resultante da mesma página após o envio da resolução pelo discente, considerando um caso de sucesso no resultado dos testes é mostrado na Figura 37.

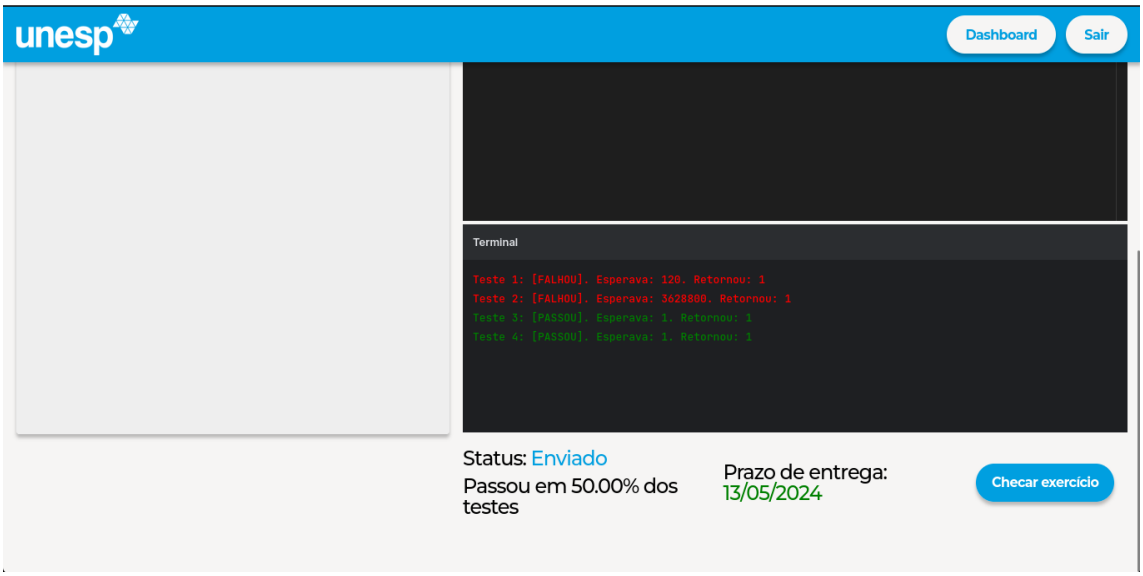
Figura 37 - Estado da página de exercícios do protótipo após o envio da resolução pelo discente, considerando um caso de sucesso no resultado dos testes



Fonte: Autoria própria.

Já o *layout* resultante considerando um cenário de falha nos resultados de alguns testes, ou seja, o aluno errou algum dos testes, é mostrado na Figura 38.

Figura 38 - Estado da página de exercícios do protótipo após o envio da resolução pelo discente, considerando um cenário de falha nos resultados de alguns testes



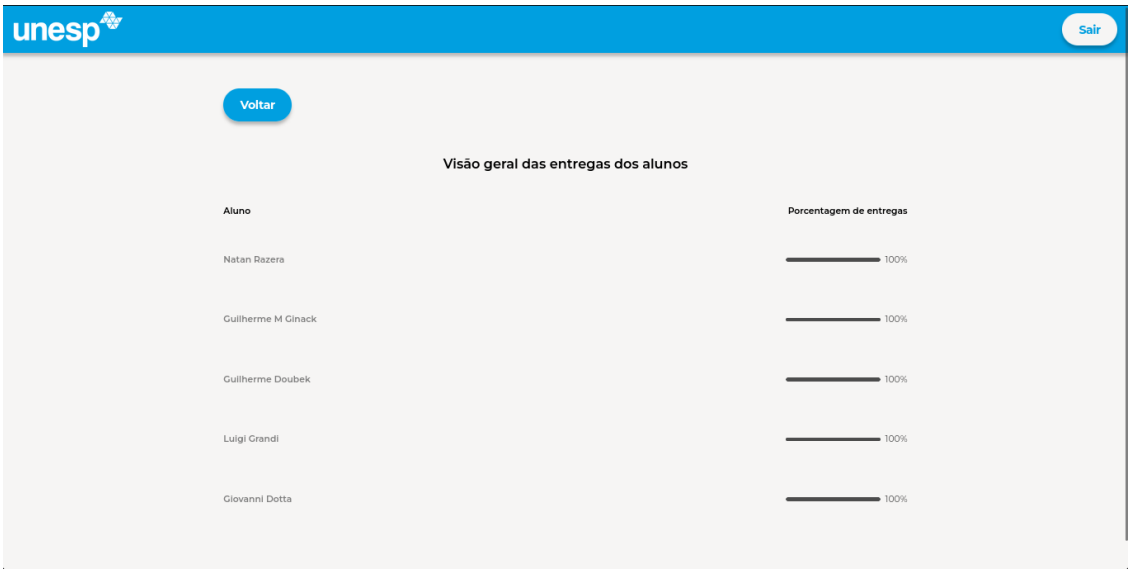
Fonte: Autoria própria.

Considerando a Figura 37 e a Figura 38, pode-se observar que o resultado de cada teste no terminal é composto pelo número de sua sequência (número do teste), o seu status final

(“PASSOU” ou “FALHOU”), o valor que era esperado como retorno da função desenvolvida e o valor que de fato foi retornado por ela.

Na etapa final do fluxo da aplicação, foi desenvolvida a página de *dashboard* e as regras de negócio delineadas na seção 3.2.3.9 (Regras de negócio - Dashboard) foram implementadas, conforme é apresentado na Figura 39.

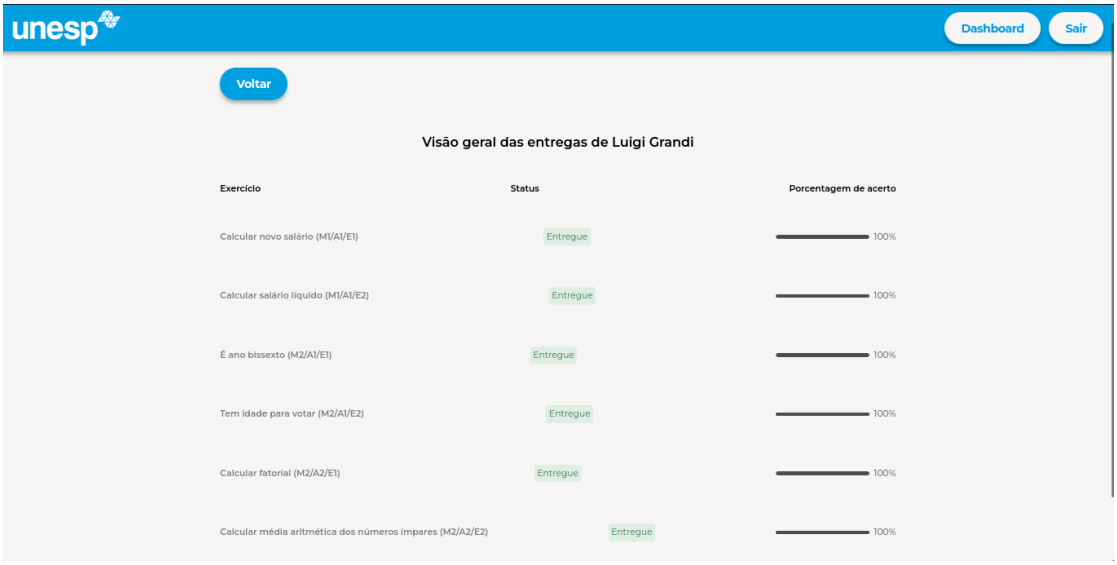
Figura 39 - Página de *dashboard* desenvolvida para o protótipo, acessível somente ao administrador



Fonte: Autoria própria.

Por fim, a página de *dashboard* do aluno foi desenvolvida em consonância com as regras de negócio descritas na seção 3.2.3.9 (Regras de negócio - *Dashboard*) (Figura 40).

Figura 40 - Página de *dashboard* do aluno desenvolvida para o protótipo, acessível somente ao administrador



Fonte: Autoria própria.

As páginas supracitadas podem ser visualizadas por meio do *link* de *deploy* da aplicação contido no apêndice B.

4.2 Apresentação e discussão das respostas do formulário de validação do protótipo

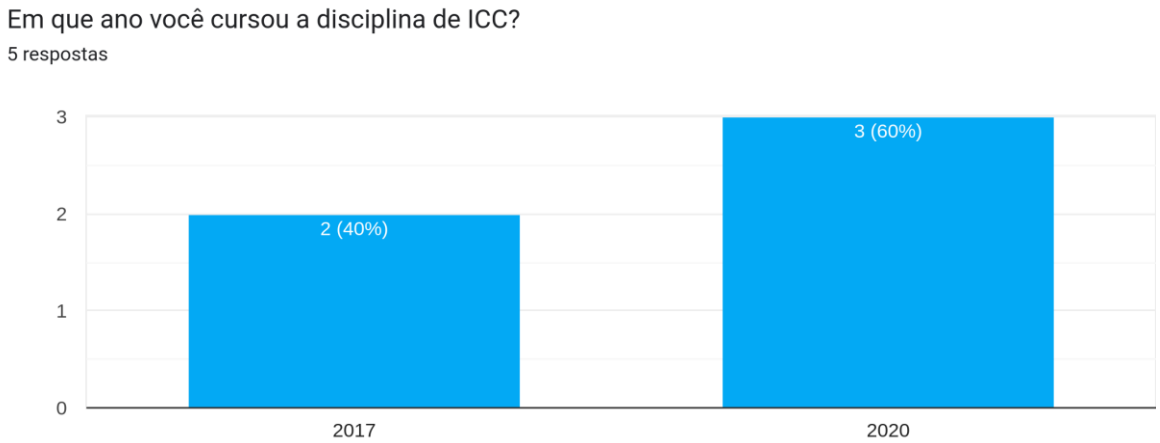
Nesta seção, as respostas coletadas no formulário de validação do protótipo são apresentadas e discutidas. Com o intuito de auxiliar a sua análise, esta seção está dividida em três tópicos: caracterização dos participantes no processo de validação; usabilidade do protótipo; relevância do protótipo para a disciplina de ICC.

4.2.1 Caracterização dos participantes no processo de validação

A questão 1 é exibida a seguir e suas respostas são apresentadas no gráfico da Figura 41.

- **Questão 1:** "Em que ano você cursou a disciplina de ICC?"

Figura 41 - Gráfico das respostas da questão 1



Fonte: Autoria própria.

A partir da Figura 41, pode-se perceber que 60% dos participantes cursaram a disciplina de ICC no ano de 2020, enquanto 40% cursaram no ano de 2017, evidenciando uma diferença significativa entre os mesmos com relação ao ano de ingresso, o qual corresponde ao ano em que a disciplina supracitada é ministrada. Tal diferença pode resultar em perspectivas, opiniões e respostas diferentes para as perguntas do formulário.

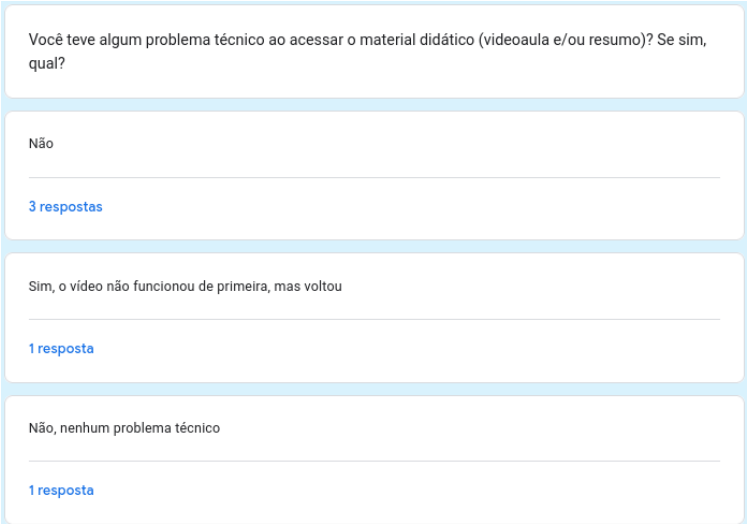
4.2.2 Usabilidade do protótipo

4.2.2.1 Recurso de material didático

O recurso de material didático consiste nas páginas que contêm a videoaula e o resumo da aula. Nesse contexto, as questões relacionadas à usabilidade deste recurso são mostradas a seguir e suas respostas são apresentadas na Figura 42 e na Figura 43, respectivamente.

- **Questão 2:** "Você teve algum problema técnico ao acessar o material didático (videoaula e/ou resumo)? Se sim, qual?"
- **Questão 3:** "Os recursos de material didático estão bem-organizados e são de fácil acesso dentro do protótipo"?

Figura 42 - Respostas da questão 2



Fonte: Autoria própria.

Figura 43 - Gráfico do nível de concordância com a questão 3

Fonte: Autoria própria.

Na Figura 42, nota-se que, exceto pela resposta de um indivíduo que presenciou uma alteração temporária na exibição da videoaula, a maioria dos participantes não enfrentou problemas técnicos com o recurso de material didático. Ademais, pode-se observar que apesar de ter sido constatado um problema técnico temporário no recurso de material didático, a sua usabilidade não foi comprometida, já que todos os participantes concordaram que este recurso está bem-organizado e que foi possível acessá-lo de forma bem-sucedida e com facilidade, conforme mostra a Figura 43.

4.2.2.2 Recurso de exercícios com feedback automático

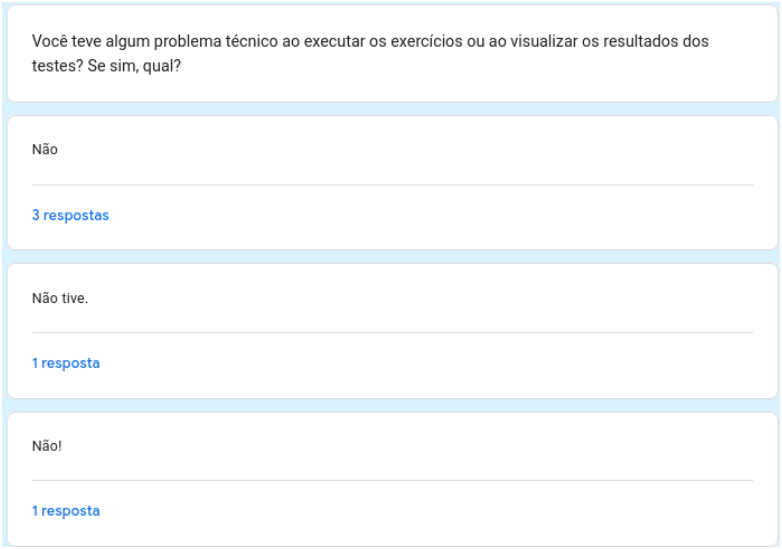
O recurso de exercícios com *feedback* automático consiste na página onde é possível resolver os exercícios e verificar se eles estão corretos ou não, a partir da realização automática de testes. As questões de usabilidade deste recurso são exibidas a seguir.

- **Questão 4:** "Você teve algum problema técnico ao executar os exercícios ou ao visualizar os resultados dos testes? Se sim, qual?"
- **Questão 5:** "Os exercícios são de fácil acesso dentro do protótipo e a forma de envio dos mesmos é intuitiva"?

As respostas obtidas referentes às questões supracitadas são exibidas na Figura 44 e na Figura 45, respectivamente.

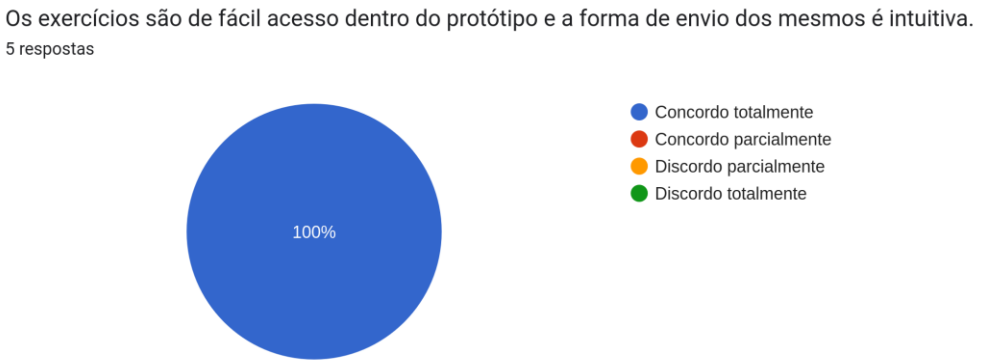
As respostas exibidas na Figura 44 evidenciam que nenhum participante enfrentou problemas técnicos ao executar os exercícios ou ao visualizar os resultados dos testes. Além disso, o alto índice de concordância total com a afirmação apresentada na Figura 45, mostra que o recurso de exercícios com feedback automático possui uma usabilidade excelente.

Figura 44 – Respostas da questão 4.



Fonte: Autoria própria.

Figura 45 - Gráfico do nível de concordância com a questão 5.



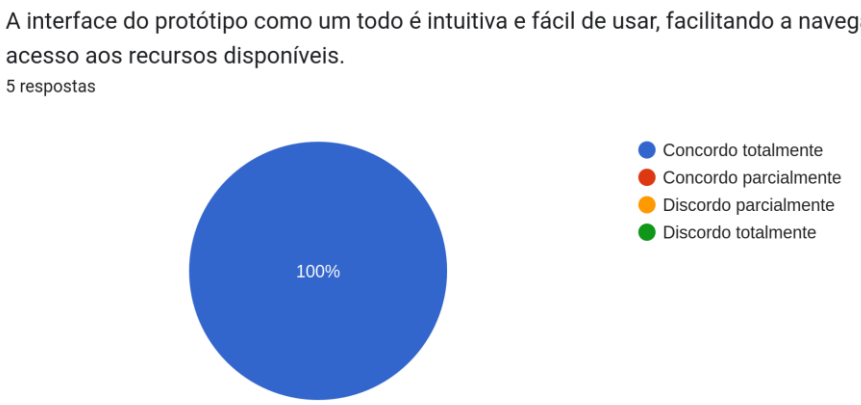
Fonte: Autoria própria.

4.2.2.3 Usabilidade geral do protótipo

A questão referente à usabilidade do protótipo como um todo é mostrada a seguir e as suas respostas são apresentadas no gráfico da Figura 46.

- **Questão 6:** “A interface do protótipo como um todo é intuitiva e fácil de usar, facilitando a navegação e o acesso aos recursos disponíveis”?

Figura 46 - Gráfico do nível de concordância com a questão 6



Fonte: Autoria própria.

Os dados mostrados na Figura 46 evidenciam que os recursos apresentados não só possuem boa usabilidade individualmente, mas também ao serem utilizados em conjunto, demonstrando facilidade de acesso aos materiais didáticos e exercícios, navegação fluída entre as páginas, intuitividade e facilidade de uso em todo o protótipo.

4.2.3 Relevância do protótipo para a disciplina de ICC

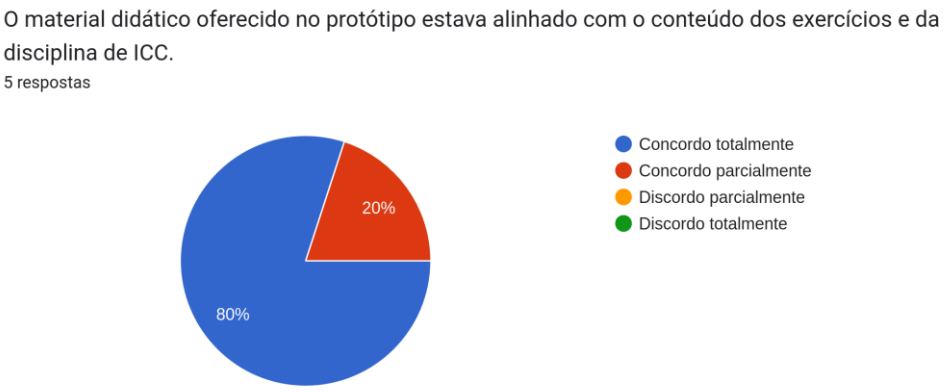
4.2.3.1 Recurso de material didático

As questões sobre a relevância do recurso de material didático para a disciplina de ICC são listadas a seguir.

- **Questão 7:** “O material didático oferecido no protótipo estava alinhado com o conteúdo dos exercícios e da disciplina de ICC”?
- **Questão 8:** “O protótipo oferece material didático claro e informativo para auxiliar no aprendizado dos conceitos de programação da disciplina de ICC”?

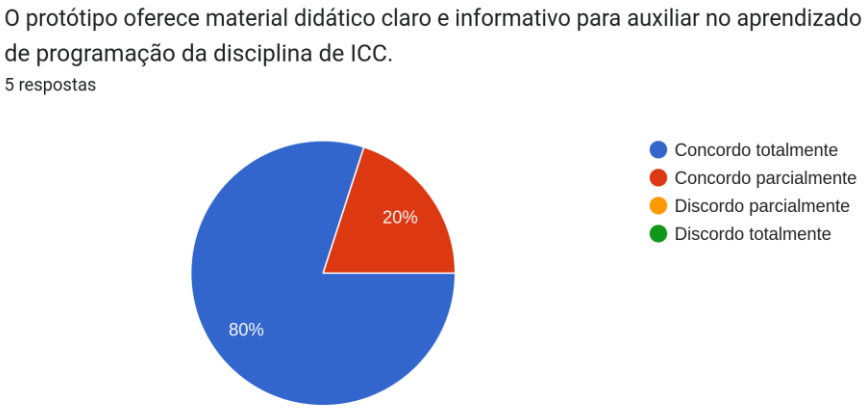
Já as respostas das mesmas são apresentadas na Figura 47 e na Figura 48.

Figura 47 - Gráfico do nível de concordância dos participantes com a questão 7



Fonte: Autoria própria.

Figura 48 - Gráfico do nível de concordância dos participantes com a questão 8



Fonte: Autoria própria.

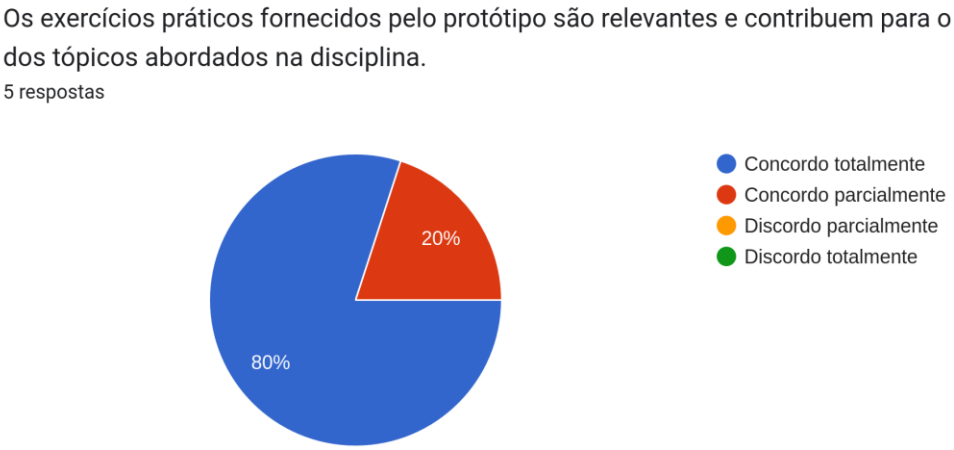
Com base nos dados da Figura 47 e da Figura 48, pode-se perceber que, mesmo tratando-se de discentes com anos de ingresso e, eventualmente, perspectivas diferentes, todos concordaram que o recurso de material didático estava alinhado e pode ser relevante para o ensino de ICC, de forma que tanto com relação ao alinhamento quanto com relação à relevância, apenas 20% não concordaram totalmente com as afirmações feitas.

4.2.3.2 Recurso de exercícios com feedback automático

As questões referentes à relevância do recurso de exercícios com feedback automático para a disciplina de ICC são mostradas a seguir e as suas respostas apresentadas na Figura 49 e na Figura 50, respectivamente.

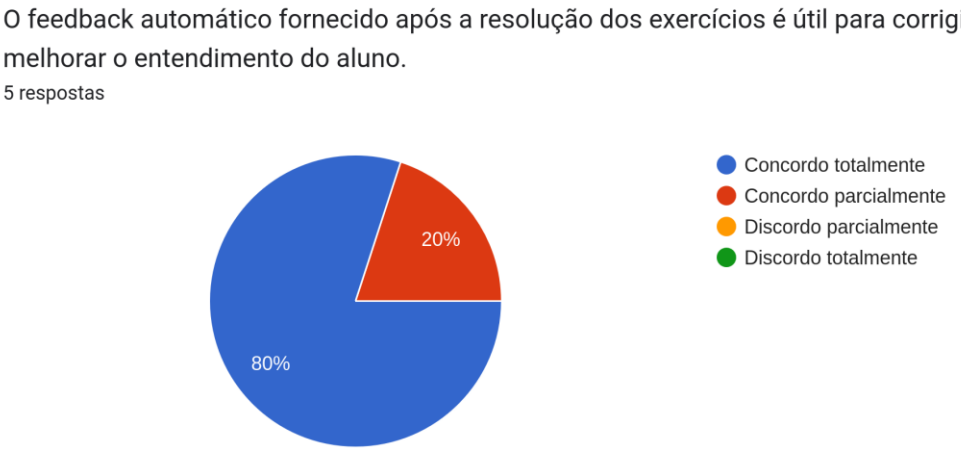
- **Questão 9:** “Os exercícios práticos fornecidos pelo protótipo são relevantes e contribuem para o entendimento dos tópicos abordados na disciplina”?
- **Questão 10:** “O feedback automático fornecido após a resolução dos exercícios é útil para corrigir erros e melhorar o entendimento do aluno”?

Figura 49 - Gráfico do nível de concordância dos participantes com a questão 9



Fonte: Autoria própria.

Figura 50 - Gráfico do nível de concordância dos participantes com a questão 10



Fonte: Autoria própria.

A partir dos dados da Figura 49 e da Figura 50, é possível observar que, todos os participantes concordam com a relevância e contribuição dos exercícios com *feedback*

automático no aprendizado da disciplina, já que em ambos os gráficos apenas 20% não concordaram totalmente com as afirmações feitas.

4.2.3.3 Relevância geral do protótipo

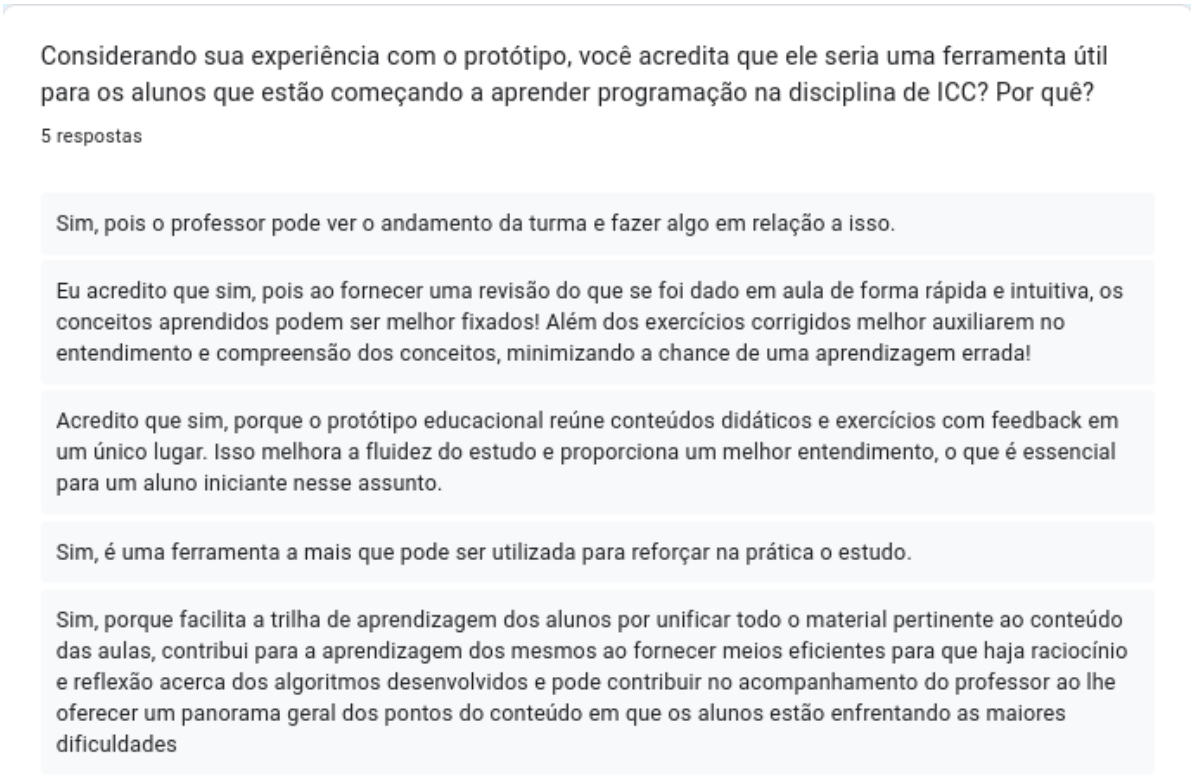
As questões sobre a relevância geral do protótipo para a disciplina de ICC são mostradas a seguir e as respostas obtidas são apresentadas na Figura 51 e na Figura 52, respectivamente.

- **Questão 11:** "Qual é o seu nível geral de satisfação com o protótipo de Software Educacional para a disciplina de ICC?"
- **Questão 12:** "Considerando sua experiência com o protótipo, você acredita que ele seria uma ferramenta útil para os alunos que estão começando a aprender programação na disciplina de ICC? Por quê?"

Figura 51 - Respostas da questão 11

Fonte: Autoria própria.

Figura 52 - Respostas da questão 12



Fonte: Autoria própria.

De forma geral, o nível de satisfação com o protótipo foi alto, já que 80% das respostas foram “Muito bom” e 20% foram “Bom”, conforme mostra a Figura 51. Já a partir das respostas da Figura 52 observa-se que, no geral, os participantes consideraram o protótipo relevante.

Dentre os comentários mostrados na Figura 52, 60% reforçam a importância do material didático, 80% enfatizam o valor agregado pelos exercícios com feedback automático e 40% ressaltam a importância da possibilidade de acompanhamento do docente.

Além disso, a unificação dos recursos supracitados em uma única ferramenta e o foco na revisão e no reforço dos conhecimentos foram pontos adicionais destacados como relevantes no protótipo.

4.2.4 Apresentação do recurso de monitoramento do docente

A visão geral das entregas dos participantes no processo de validação do protótipo, apresentada na página de *dashboard*, após finalizado o período de testes, pode ser visualizada na Figura 53.

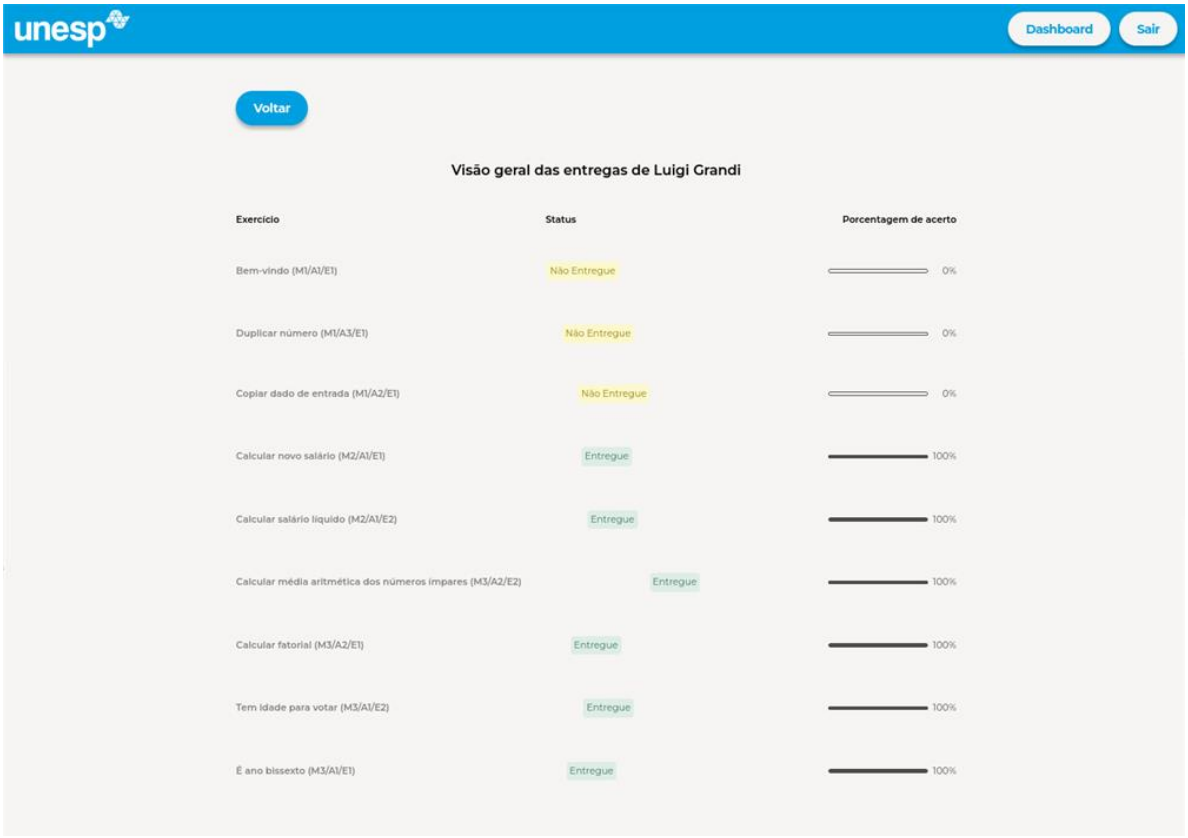
Figura 53 - Página de dashboard após finalizado o período de validação do protótipo

Fonte: Autoria própria.

Na Figura 53, pode-se observar que o docente é capaz de visualizar as porcentagens de entregas de cada aluno, possibilitando um monitoramento geral da turma.

É possível também visualizar individualmente as entregas dos participantes, conforme pode ser visto no *dashboard* mostrado na Figura 54, referente a um dos discentes que participou do processo de validação.

Figura 54 - Página de dashboard do aluno após finalizado o período de validação do protótipo



Fonte: Autoria própria.

Analisando-se a Figura 54, é possível notar que o docente é capaz de monitorar individualmente o desempenho de um determinado aluno, tendo acesso ao *status* de entrega e a porcentagem de acerto de cada exercício proposto ao mesmo.

5 CONCLUSÃO

Os resultados mostraram que a usabilidade e a relevância do recurso de material didático no protótipo foram satisfatórias, uma vez que os conteúdos foram apresentados de forma organizada, facilitando o acesso e a visualização dos mesmos. Além disso, eles se mostraram alinhados aos conceitos teóricos da disciplina e capazes de contribuir de maneira eficaz para a compreensão do conteúdo da matéria de ICC, demonstrando assim que a sua implementação foi bem planejada.

O recurso de exercícios com *feedback* automático se apresentou como uma excelente praticidade devido à ausência de problemas técnicos e ao acesso fácil e intuitivo às funcionalidades oferecidas. Ele se mostrou relevante por conta da utilidade do recurso na correção de erros, da possibilidade de aprimorar a compreensão dos alunos e de seu papel significativo no ensino do conteúdo da disciplina. Por conseguinte, também se obteve êxito na implementação do recurso em questão.

Já o mecanismo de monitoramento das atividades pelo docente mostrou-se ser um instrumento de apoio ao processo educacional com informações valiosas para promover intervenções pedagógicas adequadas e direcionadas às necessidades dos alunos, facilitando a identificação de pontos onde os discentes apresentam maior dificuldade de aprendizagem.

Com relação a interface da ferramenta, ela teve um alto nível de satisfação relatada pelos ex-alunos de ICC no processo de sua validação, pois, foram exaltadas características relativas à organização das páginas, relevância das funcionalidades implementadas e boa usabilidade, não só de todos os recursos apresentados individualmente, mas também da interação entre os mesmos, mostrando que a aplicação funciona bem como um todo.

Conclui-se que o protótipo de Software Educacional desenvolvido integra de forma clara, personalizada e abrangente, recursos de material didático de apoio, exercícios práticos com *feedback* automático e monitoramento das atividades pelo docente para a disciplina de Introdução à Ciência da Computação (ICC), mostrando que ela pode ser utilizada como material didático de apoio para o aprendizado da referida disciplina.

Trabalhos futuros

O protótipo desenvolvido pode ser estendido para outros módulos do conteúdo programático da disciplina de ICC. Além disso, por meio de pequenas modificações na interface

do usuário, o seu uso também pode ser ampliado e aplicado para outras disciplinas, como Física, Cálculo Diferencial e Integral, Álgebra Linear, entre outras. Além disso, este projeto pode servir de referência para futuros estudos ou trabalhos que desejem aprimorar e/ou incrementar as funcionalidades da aplicação desenvolvida.

REFERÊNCIAS

ALMEIDA, A. *et al*. Indicadores para avaliaçao de software educacional com base no guia gdsm (goal driven software measurement). In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, 2018. v. 29, n. 1, p. 21. Disponível em: https://www.researchgate.net/publication/328735886_Indicadores_para_Avaliacao_de_Software_Educacional_com_base_no_guia_GDSM_Goal_Driven_Software_Measurement. Acesso em: 30 jan. 2024.

ARORA, A. *How to Get Started with Node.js – Beginner's Guide to Node*. 2022. Disponível em: https://www.freecodecamp.org/news/introduction-to-nodejs/. Acesso em: 31 jan. 2024.

ATLASSIAN. *O que é o Trello: conheça recursos, usos e muito mais*. 2023. Disponível em: https://trello.com/tour. Acesso em: 15 jan. 2024.

BEECROWD. *Beecrowd Academic: Guide Your Students*. 2021. Disponível em: https://academic.beecrowd.com/pt/professors/landing. Acesso em: 20 fev. 2024.

CODECADEMY. Learn to Code - for Free | Codecademy. 2024. Disponível em: https://www.codecademy.com/. Acesso em: 21 fev. 2024.

CHACON, S.; STRAUB, B. *Pro git*. [S.l.]: Springer Nature, 2014. Disponível em: https://git-scm.com/book/en/v2. Acesso em: 15 jan. 2024.

COSTA, T. Análise dos problemas enfrentados por alunos de programação. *Trabalho de Conclusão de Curso. UEPB*, 2013. Disponível em: http://dspace.bc.uepb.edu.br/jspui/bitstream/123456789/2312/1/PDF%20-%20T%C3%BAlio%20Henriques%20Costa.pdf. Acesso em: 16 jan. 2024.

DB DESIGNER. *DB Designer: Online Database Schema Design and Modeling Tool*. 2023. Disponível em: https://www.dbdesigner.net/. Acesso em: 21 jan. 2024.

DEVMEDIA. *Tecnologia PostgreSQL*. 2015. Disponível em: https://www.devmedia.com.br/guia/tecnologia-postgresql/34328. Acesso em: 01 mar. 2024.

FONSECA, E. *O que é ORM?*. 2019. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-orm#google_vignette. Acesso em: 27 jan. 2024.

GOTARDO, R. *Linguagem de programação.* Rio de Janeiro: Seses, p. 34, 2015. Disponível em:

https://www.academia.edu/25554061/LIVRO_PROPRIETARIO_Linguagem_de_Programa. Acesso em: 16 jan. 2024.

GREIF, S.; BENITTE, R.; RAMBEAU, M. *The state of JavaScript 2018: Front-end frameworks – Overview*, 2018. Disponível em: https://2018.stateofjs.com/front-end-frameworks/overview/. Acesso em: 19 jan. 2024.

HACKERRANK. *HackerRank - Online Coding Tests and Technical Interviews. 2024*. Disponível em: https://www.hackerrank.com/. Acesso em: 18 fev. 2024.

HAHN, E. Express in Action: Writing, building, and testing Node. js applications. [S.l.]: Simon and Schuster, 2016. Disponível em: https://books.google.com.br/books?id=czkzEAAAQBAJ&printsec=frontcover&hl=pt-BR&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false. Acesso em: 02 fev. 2024.

JORGE, J. Git the impact on software development. 2018. Disponível em: https://blog.codacy.com/the-impact-of-git-on-software-development. Acesso em: 05 fev. 2024.

JÚNIOR, J. *et al*. Ensino de algoritmos e programação: uma experiência no nível médio. In: *XIII Workshop de Educação em Computação (WEI'2005). São Leopoldo, RS, Brasil*, 2005. Disponível em: https://www.researchgate.net/publication/301292800_Ensino_de_Algoritmos_e_Programacao_Uma_Experiencia_no_Nivel_Medio. Acesso em: 15 jan. 2024.

LEASE, D. *TypeScript: What is it when is it useful?*, 2018. Disponível em: https://medium.com/front-end-weekly/typescript-what-is-it-when-is-it-useful-c4c41b5c4ae7. Acesso em: 05 fev. 2024.

META PLATFORMS. *React - A JavaScript library for building user interfaces*. 2024. Disponível em: https://legacy.reactjs.org/. Acesso em: 06 fev. 2024.

MICROSOFT. *Visual Studio Code - Code Editing. Redefined*. 2024. Disponível em: https://code.visualstudio.com/. Acesso em: 18 jan. 2024.

MIR, M. A. *What are the advantages and disadvantages of using Visual Studio Code or Atom?*. 2023. Disponível em: https://medium.com/@ssc.ahmed.926748/what-are-the-advantages-and-disadvantages-of-using-visual-studio-code-or-atom-d3132bf1af85. Acesso em: 18 jan. 2024.

NOTION LABS. *Notion - The next gen of notes docs*. 2024. Disponível em: https://www.notion.so/product/docs. Acesso em: 21 jan. 2024.

OPENJS FOUNDATION. *Introduction to Node.js*. 2024. Disponível em: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs. Acesso em: 28 jan. 2024.

PRISMA DATA. *What is Prisma ORM?* 2024. Disponível em: https://www.prisma.io/docs/orm/overview/introduction/what-is-prisma. Acesso em: 28 jan. 2024.

RAPOSO, E. H. S.; DANTAS, V. O desafio da serpente-usando gamification para motivar alunos em uma disciplina introdutória de programação. In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE), 2016. v. 27, n. 1, p. 577. Disponível em: https://www.researchgate.net/publication/309884276_O_Desafio_da_Serpente_-_Usando_gamification_para_motivar_alunos_em_uma_disciplina_introdutoria_de_programacao. Acesso em: 16 jan. 2024.

RIBEIRO, A. L. S. *Trello: o que é, como funciona e os principais recursos.* 2022. Disponível em: https://www.alura.com.br/artigos/trello. Acesso em: 15 jan. 2024.

ROVEDA, U. *Desenvolvimento web: o que é e como ser um desenvolvedor web*. 2020. Disponível em: https://kenzie.com.br/blog/desenvolvimento-web/. Acesso em: 02 fev. 2024.

SOARES, E. Sistemas de Controle de Versão. 2012. Disponível em: https://www.devmedia.com.br/sistemas-de-controle-de-versao/24574. Acesso em: 02 fev. 2024.

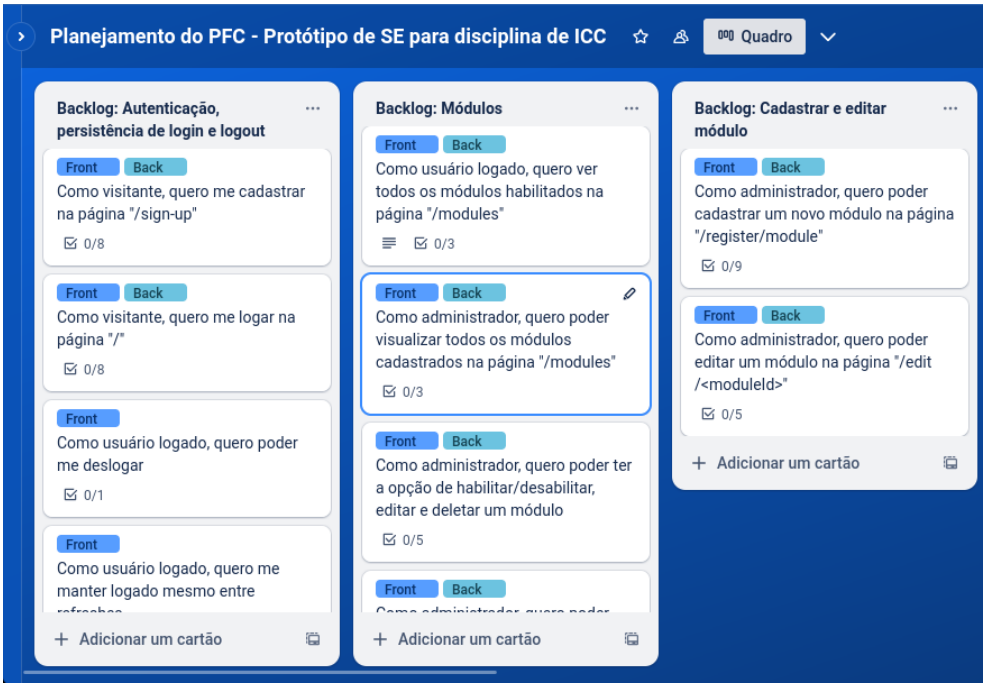
SOUZA, T. T. D. *Gestão de versão de software: um estudo de caso*. Engenharia de Projetos de Software-Florianópolis, 2015. Disponível em: https://repositorio.animaeducacao.com.br/items/d470b3e6-6fad-486c-bae3-cb2c1f391dc7. Acesso em: 20 jan. 2024.

SOUZA, M. de; SILVA, E. A. da. Estudo comparativo de tecnologias de desenvolvimento front-end para web. *Anais do Computer on the Beach*, v. 12, p. 201–208, 2021. Disponível em: https://periodicos.univali.br/index.php/acotb/article/view/17402. Acesso em: 24 jan. 2024.

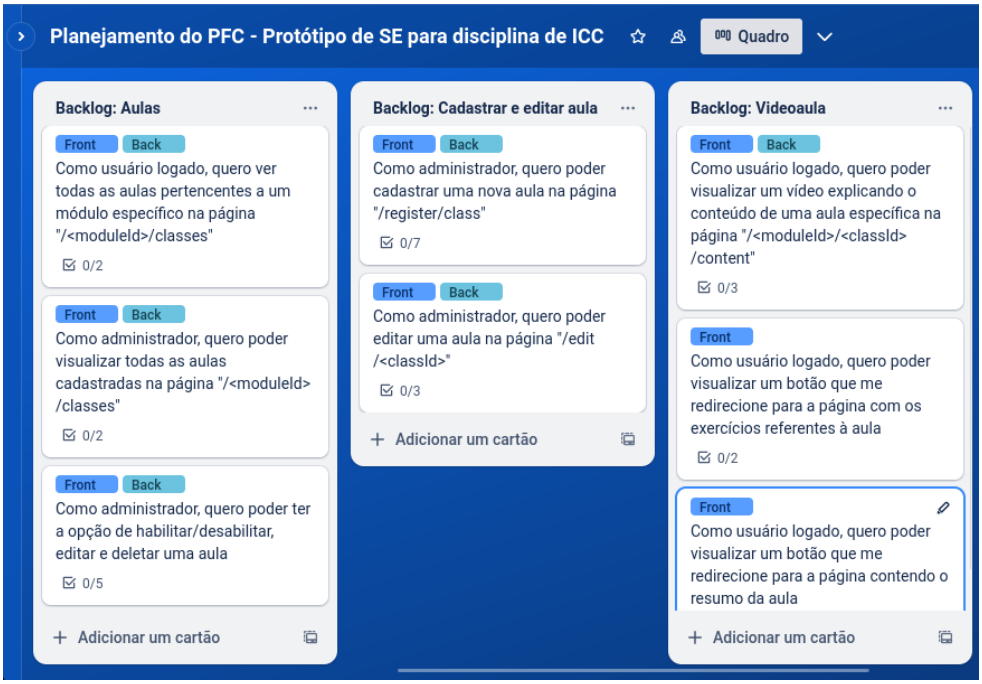
STRONGLOOP; IBM. *Express - Fast, unopinionated, minimalist web framework for Node.js*. 2017. Disponível em: https://expressjs.com/. Acesso em: 31 jan. 2024.

VILLAIN, M.; SILVEIRA, M. I. *Figma: o que é a ferramenta, Design e uso.* 2023. Disponível em: https://www.alura.com.br/artigos/figma. Acesso em: 18 jan. 2024.

APÊNDICE A – VISÃO GERAL DO QUADRO DO TRELLO CONTENDO AS REGRAS DE NEGÓCIO DEFINIDAS PARA O PROTÓTIPO



(a)



(b)


Planejamento do PFC - Protótipo de SE para disciplina de ICC
Quadro
Backlog: Resumo da aula
Front
Back
Como usuário logado, quero poder visualizar o resumo da aula na página "/<moduleId>/<classId> /<summaryd>"
0/2
Front
Como usuário logado, quero poder ir para os exercícios
0/2
Adicionar um cartão
Backlog: Exercícios
Front
Back
Como usuário logado, quero poder visualizar o enunciado do exercício e um editor de código na página "/<moduleId>/<classId> /<exerciseId>"
0/4
Front
Back
Como usuário logado, quero poder verificar se a minha resolução do exercício está correta
0/6
Front
Back
Como usuário logado, quero poder enviar minha resolução do exercício para o professor
0/5
Adicionar um cartão
Backlog: Dashboard
Front
Back
Como administrador, quero poder ter um visão geral do desempenho dos alunos na página "/dashboard"
0/3
Front
Back
Como administrador, quero poder ter um visão geral do desempenho de um aluno específico na página "/dashboard/<studentId>"
0/4
Adicionar um cartão

(c)


Planejamento do PFC - Protótipo de SE para disciplina de ICC
Quadro
Backlog - Deploy
Front
Back
O usuário deve poder acessar a plataforma em um link da internet
Adicionar um cartão
Em Progresso
Adicionar um cartão
Concluído
Adicionar um cartão

(d)

APÊNDICE B – *LINK* DO *DEPLOY* DO PROTÓTIPO

Link de acesso ao protótipo: https://pfc-front.vercel.app/