UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA AUXÍLIO AOS CANDIDATOS DOS VESTIBULARES DA UFSC

Bruno Barreto

Florianópolis – SC
2024

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA AUXÍLIO AOS CANDIDATOS DOS VESTIBULARES DA UFSC

Trabalho de Conclusão de Curso de Graduação em Sistemas de Informação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.

Bruno Barreto

Orientador:
Prof. Dr. Frank Augusto Siqueira

Bruno Barreto

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA AUXÍLIO AOS CANDIDATOS DOS VESTIBULARES DA UFSC

Trabalho de Conclusão de Curso de Graduação em Sistemas de Informação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Sistemas de Informação, sob apreciação da seguinte Banca Examinadora:

Prof. Dr. Alex Sandro Roschildt Pinto (UFSC)

Prof. Dr. Antonio Carlos Mariani (UFSC)

Aprovado em …

Prof. Dr. Frank Augusto Siqueira (UFSC)

AGRADECIMENTOS

Agradeço, em primeiro lugar, à minha família — Carina, Andrey e Brayan — por todo o apoio, carinho e incentivo ao longo dessa jornada. Sem vocês, nenhuma conquista teria o mesmo valor. A cada passo, a força e compreensão de vocês foram fundamentais para que eu pudesse chegar até aqui.

Também dedico um agradecimento especial à minha companheira, Brenda, por sua presença constante, por estar sempre ao meu lado e por me apoiar em todos os momentos, compartilhando comigo os desafios e as vitórias.

Por fim, expresso minha gratidão à Universidade Federal de Santa Catarina (UFSC) e aos professores com os quais tive o privilégio de aprender. A experiência proporcionada por esta instituição foi essencial para minha formação, ampliando meus horizontes e me guiando em direção a novos conhecimentos e realizações.

RESUMO

A busca pela entrada em instituições de ensino superior envolve múltiplas etapas, como o processo de inscrição e a obtenção de informações atualizadas. Este trabalho propõe o desenvolvimento de um aplicativo móvel para simplificar e aprimorar a experiência dos candidatos no vestibular da Universidade Federal de Santa Catarina (UFSC). O aplicativo foi projetado para oferecer uma interface intuitiva e funcionalidades que facilitem o acesso a informações detalhadas sobre os vestibulares, incluindo datas e locais de prova, boletim de desempenho individual, classificação e convocação para matrícula. Para avaliar a usabilidade do aplicativo, será utilizado o questionário SUS (*System Usability Scale*), um método amplamente adotado que medirá a facilidade de uso, a clareza das informações e a satisfação geral dos usuários, fornecendo dados quantitativos e qualitativos para validar e aprimorar a experiência do usuário.

Palavras-chave: Aplicativo Móvel. Vestibular. Experiência do Candidato. Processo de Inscrição. Avaliação de Desempenho.

ABSTRACT

The pursuit of entry into higher education institutions involves multiple stages, including the application process and access to updated information. This work proposes the development of a mobile application to simplify and enhance the experience of candidates applying for the entrance exam at the Federal University of Santa Catarina (UFSC). The application will be designed to offer an intuitive interface and features that facilitate access to detailed information about the exams, such as exam dates and locations, individual performance reports, rankings, and enrollment calls. To assess the application's usability, the System Usability Scale (SUS) questionnaire will be employed. This widely used method will measure ease of use, clarity of information, and overall user satisfaction, providing both quantitative and qualitative data to validate and improve the user experience.

Keywords: Mobile Application. Entrance Exam. Candidate Experience. Registration Process. Performance Evaluation.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Neste capítulo, será apresentado o contexto e a motivação para o desenvolvimento deste projeto, assim como os objetivos gerais e específicos que ele busca atingir. Em seguida, serão abordadas as delimitações do escopo e a metodologia de pesquisa adotada para embasar e estruturar o trabalho. Por fim, é descrita a organização do restante do texto, destacando os capítulos subsequentes que compõem este documento.

1.1. CONTEXTUALIZAÇÃO

A evolução dos processos educacionais é fortemente impulsionada pela tecnologia, e a otimização da experiência dos candidatos nos vestibulares é um exemplo claro dessa tendência. Em um cenário onde a entrada em instituições de ensino superior é um marco significativo na trajetória acadêmica, aprimorar o processo de inscrição e disponibilizar informações relevantes torna-se crucial.

No contexto atual, o vestibular da Universidade Federal de Santa Catarina (UFSC) adota um sistema Web que centraliza todas as informações pertinentes, como inscrições, datas e locais de prova, além de outros dados relevantes. Uma pesquisa realizada pela empresa americana eMarketer (2020) revela que as pessoas utilizam aplicativos cerca de oito vezes mais do que sites, atribuindo essa preferência à praticidade e responsividade dos aplicativos em dispositivos móveis.

Para os estudantes imersos no período de vestibular, focados intensamente nos estudos, acompanhar seu processo de inscrição no vestibular pode ser muito mais interessante via dispositivo móvel, visto que, segundo dados da pesquisa da FGV (2023), o Brasil conta com aproximadamente 464 milhões de dispositivos móveis, o dobro do número de computadores.

Diante desse contexto, o presente trabalho propõe a construção de um aplicativo móvel destinado a auxiliar os candidatos no processo de inscrição de vestibulares da UFSC. Este aplicativo será desenvolvido utilizando tecnologias modernas que possibilitem a criação de aplicativos multiplataforma, com o intuito de notificar e fornecer informações atualizadas aos candidatos sobre sua inscrição, datas de prova, locais de aplicação do exame, entre outras informações de relevância durante todo o processo seletivo.

1.2. OBJETIVOS

Esta seção define os objetivos que orientaram o desenvolvimento do projeto. O objetivo geral destaca a finalidade principal do aplicativo, enquanto os objetivos específicos detalham as etapas e metas que contribuirão para o alcance desse propósito. Esses objetivos foram estabelecidos para assegurar que o aplicativo atenda às necessidades dos candidatos aos vestibulares da UFSC, proporcionando uma experiência prática e informativa durante o processo de inscrição e acompanhamento de seu desempenho.

1.2.1. OBJETIVO GERAL

O objetivo principal deste trabalho é desenvolver um aplicativo móvel que tem como objetivo aprimorar a experiência dos candidatos durante o processo de inscrição para os vestibulares da Universidade Federal de Santa Catarina (UFSC).

O sistema fornecerá informações detalhadas sobre inscrições, datas, locais de prova, entre outras informações essenciais para auxiliar os candidatos em sua jornada para ingressar na UFSC. Também será possível receber notificações a respeito de um vestibular no qual o candidato esteja inscrito ou vestibulares em aberto.

1.2.2. OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho que podem ser citados são:

- Analisar a experiência dos candidatos no processo de inscrição dos vestibulares da UFSC.
- Desenvolver um aplicativo móvel para facilitar a inscrição dos candidatos nos vestibulares da UFSC.
- Analisar se o projeto obteve o resultado esperado.

1.3. DELIMITAÇÕES

Para garantir a viabilidade e o foco do projeto, serão estabelecidas as seguintes delimitações no escopo do desenvolvimento do aplicativo:

- **INSCRIÇÕES NO VESTIBULAR**

O aplicativo não permitirá que os candidatos efetuem a inscrição no vestibular diretamente pelo APP. A funcionalidade de inscrição continuará a ser realizada exclusivamente pelo site oficial do Vestibular da UFSC. Isso se deve a questões de segurança e integração com o sistema existente de inscrições.

- **ACESSO À INTERNET**

O aplicativo requererá acesso à internet para seu funcionamento. Todas as funcionalidades e informações disponibilizadas pelo APP dependerão de uma conexão ativa com a internet. Isso inclui acesso a dados atualizados, notificações e sincronização com os sistemas da UFSC.

- **PLATAFORMA DE TESTES**

Os testes e validações do aplicativo serão realizados exclusivamente em smartphones com o sistema operacional Android. Não será contemplado, nesta fase do projeto, o desenvolvimento e testes para outras plataformas como iOS ou Windows Phone. Esta delimitação permite concentrar recursos e esforços na plataforma mais amplamente utilizada pelos potenciais usuários.

1.4. MÉTODO DE PESQUISA

O desenvolvimento deste projeto será guiado por uma metodologia estruturada e completa, que envolve várias etapas essenciais, desde a compreensão inicial dos requisitos até a validação do produto final. As etapas

descritas a seguir detalham o processo metodológico que será seguido, abrangendo desde a análise de requisitos, design da interface, implementação técnica, até os testes de usabilidade e validação com usuários. As etapas descritas a seguir detalham o processo metodológico que será seguido:

- **ESTUDO DE CASO INICIAL**

Será realizado um estudo de caso com o objetivo de compreender o contexto e as necessidades dos usuários envolvidos no processo do vestibular da UFSC. Isso incluirá a análise dos processos atuais e a identificação de possíveis melhorias e funcionalidades essenciais para o aplicativo.

- **CONSULTAS COM RESPONSÁVEIS**

Serão conduzidas entrevistas e consultas detalhadas com os funcionários responsáveis pelo vestibular na UFSC. Estas sessões visam obter informações técnicas e operacionais críticas sobre o funcionamento atual do sistema. Através dessas interações, será possível mapear os requisitos do sistema, identificar desafios e coletar insights valiosos para o desenvolvimento do aplicativo.

- **PROTOTIPAÇÃO DO APLICATIVO**

Com base nas informações coletadas durante as etapas anteriores, será realizada a prototipação do aplicativo. Esta fase envolverá a criação de *wireframes* e *mockups* que serão baseados no site atual do vestibular da UFSC. A prototipação permitirá a visualização preliminar das funcionalidades e a estrutura do aplicativo, facilitando a identificação de melhorias antes do desenvolvimento final.

- **DESENVOLVIMENTO DO APLICATIVO**

Após a aprovação e refinamento dos protótipos, será iniciado o desenvolvimento efetivo do aplicativo. Esta etapa incluirá a codificação, integração de funcionalidades, testes internos e correção de bugs. Serão utilizadas tecnologias

modernas e métodos ágeis para garantir eficiência e adaptabilidade ao longo do processo de desenvolvimento.

- **TESTES DE USABILIDADE E COLETA DE FEEDBACKS**

Com o aplicativo desenvolvido, serão realizados testes de usabilidade abrangentes para garantir que o produto seja intuitivo e eficaz para todos os usuários. Serão selecionados usuários com diferentes níveis de afinidade tecnológica para testar o aplicativo. Os *feedbacks* coletados durante esta fase serão essenciais para realizar ajustes finais e aprimorar a experiência do usuário.

- **PESQUISA CONTÍNUA DE TECNOLOGIAS E MÉTODOS**

Durante todas as fases do projeto, será mantida uma pesquisa contínua sobre novas tecnologias, métodos de desenvolvimento, e melhores práticas da indústria. Esta pesquisa garantirá que o projeto esteja alinhado com as tendências atuais e utilize as ferramentas mais eficientes e adequadas.

1.5. ESTRUTURA DO TRABALHO

A estrutura do trabalho será composta da seguinte forma:

- **Capítulo 1 - Introdução**: Este capítulo apresenta o contexto e a motivação para o desenvolvimento do aplicativo, destacando os desafios enfrentados pelos candidatos nos processos de inscrição e acesso a informações sobre vestibulares. Também são definidos o problema, os objetivos gerais e específicos, e a justificativa para a criação deste sistema.
- **Capítulo 2 - Fundamentação Teórica**: Este capítulo aborda os conceitos e teorias que embasam o desenvolvimento do aplicativo, incluindo usabilidade, acessibilidade e escalas de avaliação de sistemas, como o SUS (System Usability Scale). Também é explorado o contexto dos processos de inscrição em instituições de ensino superior, com foco nas necessidades informacionais dos candidatos.

- **Capítulo 3 - Metodologia**: No capítulo de metodologia, são descritos o planejamento e as etapas seguidas para o desenvolvimento do aplicativo, incluindo a escolha das tecnologias, a definição do público-alvo e o método de coleta de dados para a avaliação da usabilidade. São detalhadas as ferramentas e os métodos utilizados, bem como os critérios de avaliação aplicados durante o desenvolvimento e teste do sistema.
- **Capítulo 4 - Desenvolvimento do Sistema e Resultados**: Este capítulo descreve o processo de desenvolvimento do aplicativo, abordando aspectos técnicos da implementação, como a estrutura do sistema, as funcionalidades implementadas e as tecnologias utilizadas. Os resultados do questionário SUS, que avaliam a usabilidade do sistema, também são apresentados e analisados, fornecendo uma visão sobre a aceitação do aplicativo pelos usuários.
- **Capítulo 5 - Conclusão e Trabalhos Futuros**: No capítulo final, são apresentadas as conclusões a partir dos resultados obtidos, avaliando o cumprimento dos objetivos propostos e o impacto do sistema na experiência do usuário. Também são sugeridas direções para trabalhos futuros, abordando possíveis melhorias no sistema, expansão de funcionalidades e novos estudos para aprofundar a análise de usabilidade e eficácia do aplicativo.

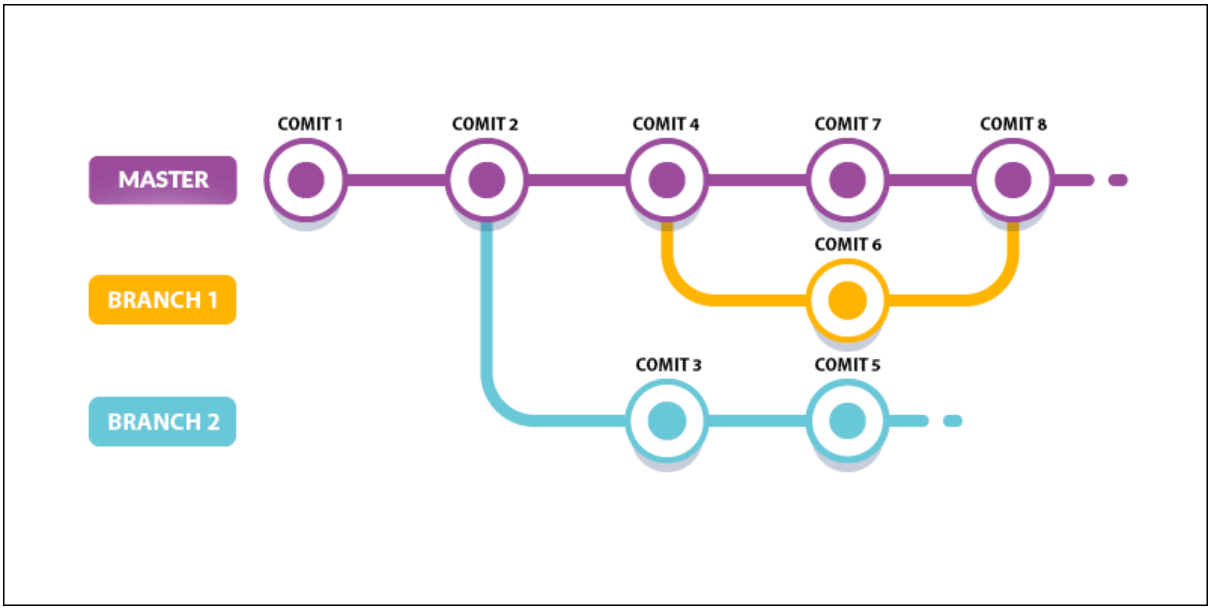
2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão exploradas as principais tecnologias e conceitos relacionados que serão utilizadas no presente projeto. Também serão citados trabalhos relacionados seguidamente de uma comparação com o presente projeto.

2.1. GIT E GITHUB

Git é o sistema de controle de versão mais popular atualmente, criado em 2005 por Linus Torvalds, também criador do sistema operacional Linux. É uma ferramenta de código aberto, desenvolvida para gerenciar projetos de software de pequeno, médio e grande porte com velocidade e eficiência. Com o Git, é possível manter o histórico de alterações separados por versões (*branches*) de um mesmo projeto de forma distribuída em diferentes computadores, e também cada branch pode ter uma sequência de versões (*commits*).

Figura 1: Versionamento do Git utilizando branches.



Fonte: Full Cycle (2019).

GitHub é uma plataforma de hospedagem de códigos e arquivos em nuvem com controle de versão utilizando o Git. Criado em 2008 e comprado pela Microsoft em 2018, o GitHub é a plataforma mais conhecida e utilizada por desenvolvedores para armazenamento de projetos de software.

Essas ferramentas serão úteis para manter o código do projeto seguro e proporcionar o seu versionamento.

2.2. JAVASCRIPT, TYPESCRIPT E NODEJS

JavaScript é uma linguagem de programação criada em 1995 por Brendan Eich. Inicialmente foi desenvolvida para ser executada apenas no navegador, porém hoje já existem soluções que permitem o JavaScript ser executado e interpretado do lado do servidor. O JavaScript não é uma linguagem tipada, o que acarreta algumas dificuldades em projetos de grande escala, como por exemplo, a depuração de código. Em busca de resolver esse problema, a Microsoft lançou em 2012 o TypeScript, com o objetivo de trazer uma tipagem estática para o JavaScript, permitindo uma melhor robustez e confiabilidade aos projetos.

NodeJS é um ambiente de execução de código JavaScript do lado do servidor, construído sobre o motor V8 do Google Chrome. Ele permite que os desenvolvedores usem JavaScript para escrever códigos do lado do servidor, em vez de utilizar apenas para interatividade do lado do cliente em navegadores Web.

Para o projeto, será utilizado o TypeScript tanto no Front-End, quanto no Back-End, e para o servidor, será utilizado o NodeJS.

2.3. SQLITE E PRISMA ORM

SQLite é um poderoso Sistema de Gerenciamento de Banco de Dados (SGBD) que utiliza a Linguagem de Consulta Estruturada (SQL), tendo suas origens nos anos 2000 por Dwayne Richard Hipp.

PrismaORM é um framework JavaScript de Object-Relational Mapping (ORM) que possui integração com bancos de dados relacionais e não relacionais, incluindo SQLite e MongoDB. O PrismaORM é capaz de abstrair a comunicação entre o servidor e o banco de dados através do mapeamento entre modelos de objetos (aplicação) e modelos relacionais (banco de dados), sendo possível a criação de tabelas, migrações, consultas, entre outras operações.

O SQLite será utilizado como banco de dados para o projeto, principalmente por ser um banco simples de ser configurado, para armazenar informações úteis, como por exemplo os tokens de notificação dos usuários. Em conjunto, será

utilizado o PrismaORM, para facilitar a interação entre servidor e banco de dados, oferecendo também o suporte às migrações.

2.4. REACT NATIVE E EXPO

React Native é um framework JavaScript, lançado em 2015 pela Meta, que é capaz de criar interfaces multiplataforma utilizando o mesmo código base, gerando o código nativo, por exemplo Java (Android) ou Swift (iOS), a partir de um código React escrito com JavaScript.

Expo é uma ferramenta capaz de facilitar o desenvolvimento, implantação e manutenção de um aplicativo desenvolvido em React Native. Com Expo, é possível executar o aplicativo com muito mais praticidade e agilidade através do Expo GO, e também compilar e gerar os executáveis do projeto através do Expo Application Services (EAS).

O aplicativo do projeto será desenvolvido com Expo, a fim de facilitar o desenvolvimento e implantação do aplicativo. Também será utilizado o Expo GO para execução do código em ambiente de desenvolvimento.

2.5. TRABALHOS RELACIONADOS

Nesta seção, serão apresentados aplicativos móveis com finalidade semelhante à do aplicativo que está sendo proposto neste trabalho de conclusão de curso. Por fim, será feita uma análise comparativa entre os trabalhos relacionados já existentes com o aplicativo proposto.

2.5.1 EUVESTIBULANDO

O Trabalho de Conclusão de Curso (TCC) apresentado por Bergária, Reis e Roland (2021), introduz uma solução para um problema comum entre os estudantes que se preparam para vestibulares: a organização e gerenciamento de datas importantes. O projeto, intitulado EUVESTIBULANDO, é um aplicativo móvel projetado para auxiliar os vestibulandos a manterem o controle sobre os prazos de

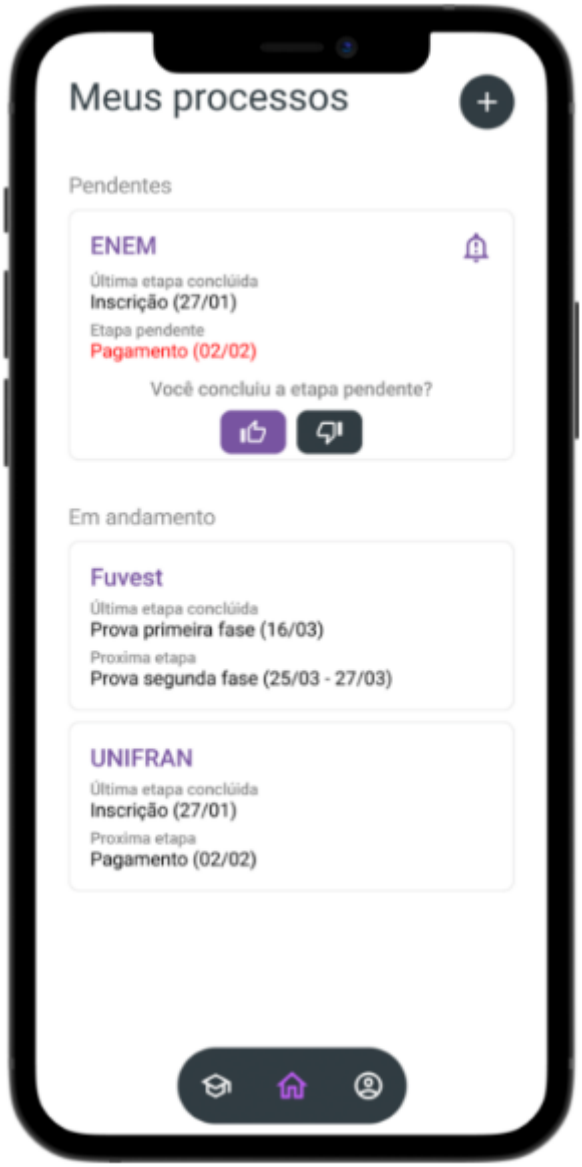
inscrição e datas de provas dos diversos processos seletivos, um desafio que muitos enfrentam durante esse período crítico de suas vidas acadêmicas.

A motivação por trás do desenvolvimento do EUVESTIBULANDO surge da observação dos autores sobre a necessidade de uma ferramenta que centralize informações e notificações importantes, permitindo aos estudantes focar no que realmente importa: seus estudos e bem-estar. Para alcançar esse objetivo, o aplicativo foi construído utilizando uma combinação de tecnologias modernas e práticas de engenharia de software, incluindo React Native, Node.js e PostgreSQL. A metodologia ágil foi adotada para o desenvolvimento do projeto, garantindo flexibilidade e eficiência ao longo do processo.

O aplicativo não apenas fornece um meio para os estudantes acompanharem as datas dos vestibulares, mas também oferece funcionalidades como a personalização de alertas e um sistema de notificações para atualizações em tempo real. A implementação detalhada no TCC abrange desde a interface do usuário até o backend, incluindo a API e um sistema de monitoramento que verificam mudanças nos sites dos exames, assegurando que os usuários estejam sempre informados sobre as últimas novidades.

A Figura 2, retirada do artigo, mostra uma prototipação da tela inicial do projeto.

Figura 2: Tela inicial do protótipo (EUVESTIBULANDO).



Fonte: EUVESTIBULANDO (2021).

Contudo, de acordo com os autores, o EUVESTIBULANDO representa um avanço significativo na maneira como os estudantes interagem com o calendário dos vestibulares, proporcionando uma ferramenta robusta e confiável que promete simplificar uma das facetas mais estressantes da vida de um vestibulando. Os autores do TCC demonstram com sucesso a aplicabilidade de seus conhecimentos em engenharia de software para criar uma solução prática que tem o potencial de beneficiar milhares de estudantes em todo o país.

2.5.2. ENEM GO

O Trabalho de Conclusão de Curso "Enem Go: Um Aplicativo de Estudo Gamificado para o Exame Nacional do Ensino Médio", desenvolvido por Matheus Antunes Vieira e Daniel Cardoso Moraes de Oliveira (2022), publicado na Universidade Federal Fluminense, aborda a criação de um aplicativo móvel para auxiliar os estudantes na preparação para o Exame Nacional do Ensino Médio (ENEM).

Os autores começam o artigo explicando o que é o Enem, sendo ele uma prova do Governo Federal que avalia o desempenho escolar do ensino médio dos participantes e proporciona a possibilidade de ingresso ao ensino superior. Devido à importância desse exame, Vieira e Oliveira propuseram, desenvolveram e publicaram o aplicativo móvel Enem Go na loja de aplicativos Google Play Store.

O Enem Go, conforme descrito pelos autores, utiliza conceitos de gamificação como motores motivacionais para auxiliar no processo preparatório para a realização da prova. A gamificação é uma estratégia que utiliza elementos de jogos em contextos que não são jogos, com o objetivo de aumentar o engajamento e a motivação dos usuários. No caso do Enem Go, a gamificação é usada para tornar o estudo para o Enem mais interessante e envolvente.

A Figura 3, retirada do artigo, mostra uma prototipação da tela inicial do projeto.

Figura 3: Tela inicial do protótipo (ENEM GO).

Fonte: ENEM GO (2022).

Também é destacado que o aplicativo está em conformidade com os critérios de eficiência e satisfação. Para validar isso, o aplicativo foi testado com os potenciais usuários através do teste de usabilidade com o suporte da escala SUS.

A escala System Usability Scale (SUS) é uma ferramenta simples e confiável para medir a usabilidade. Ela consiste em um questionário de 10 itens com cinco opções de resposta para os entrevistados de "Concordo totalmente" a "Discordo totalmente". Os resultados do teste, conforme relatado pelos autores, concluíram que a aplicação apresentou um resultado satisfatório em relação aos critérios.

2.5.3. ANÁLISE COMPARATIVA

Comparando os trabalhos apresentados com o presente projeto, podemos notar que o projeto atual traz objetivos e funcionalidades que os trabalhos apresentados não possuem, tendo como principal diferença, o fato de que nenhum deles se trata especificamente de um aplicativo para auxiliar os candidatos em um vestibular. Comparado com o EUVESTIBULANDO, o aplicativo proposto integra o usuário com o sistema real do vestibular, garantindo informações corretas e atualizadas, já o EUVESTIBULANDO seria apenas uma gestão de datas a serem cadastradas individualmente pelo usuário.

Tabela 1: Tabela de comparação entre trabalhos relacionados com o projeto proposto

	Envio de Notificações	Teste de Usabilidade	Integrado com o Vestibular	Gestão Personalizada
ENEM GO		X		
EUVESTIBULANDO	X			X
APP VESTIBULAR UFSC	X	X	X	

Fonte: Elaborado pelo autor.

3. PROPOSTA DO APLICATIVO

Neste capítulo, serão apresentadas as funcionalidades do aplicativo, desde a análise de requisitos até a criação de *wireframes* e protótipos das telas do aplicativo.

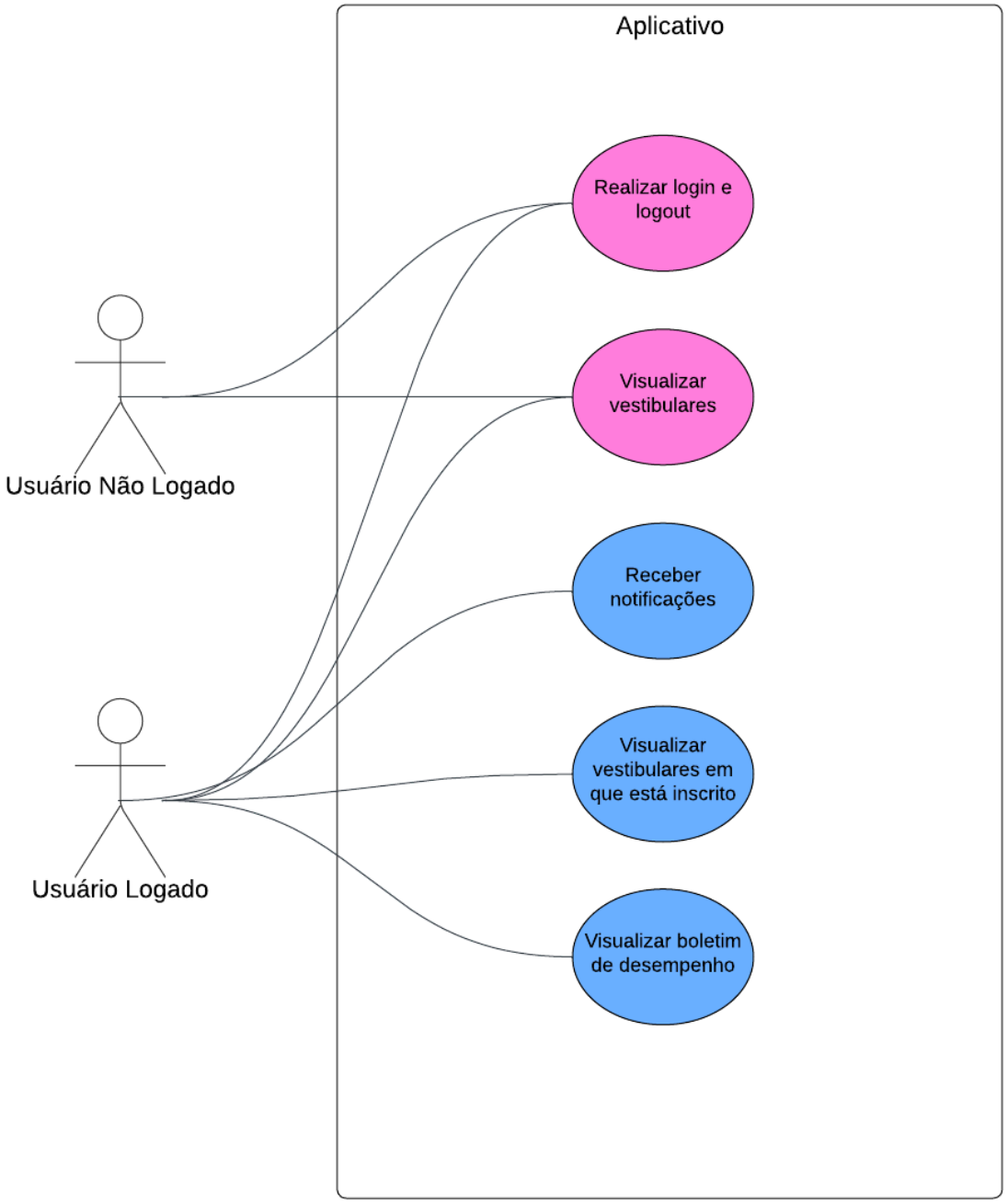
Ao inicializar o aplicativo, o usuário poderá visualizar uma lista de vestibulares atuais, obtendo informações iniciais ou fazer o *download* do edital do respectivo vestibular, mesmo sem estar logado. Ainda sem necessidade de login, o usuário também terá acesso ao módulo de validação de autenticidade de documentos, permitindo a validação de documentos através de identificadores únicos.

O *login* no aplicativo poderá ser realizado com CPF e senha previamente registrados no site oficial. Após efetuar o *login*, novas funcionalidades serão liberadas ao candidato, incluindo:

- **Visualização Detalhada de Vestibulares:** Acesso a informações detalhadas sobre os vestibulares da UFSC, como locais de prova, resultados, gabaritos, entre outros.
- **Notificações Importantes:** Recebimento e visualização de notificações sobre informações e atualizações importantes relacionadas ao vestibular no qual o candidato está participando ou vestibulares em aberto, como por exemplo, o anúncio de abertura de inscrições para um determinado vestibular ou alertas de chamadas de candidatos em lista de espera.

A Figura 4 se trata de um diagrama de casos de uso que exibe as funcionalidades base do aplicativo que foram citadas, com o usuário autenticado e não autenticado.

Figura 4: Diagrama de casos de uso com funções base do aplicativo.



Fonte: Elaborado pelo autor.

3.1. ANÁLISE DE REQUISITOS DE SOFTWARE

Requisitos de software são as funções que determinado sistema deve possuir, estando relacionados com os objetivos que o cliente tem com o produto final. Segundo a IEEE (1990), a análise de requisitos de software é um processo que envolve um estudo de necessidades dos usuários, sendo indispensável para o desenvolvimento de um sistema. Os requisitos podem ser definidos entre requisitos diretos, também denominados de funcionais e requisitos indiretos, também denominados não funcionais.

3.1.1. REQUISITOS FUNCIONAIS

Os requisitos funcionais ou requisitos diretos, são aquelas funções que o software precisa para cumprir o seu objetivo, ou seja, funções que definem o comportamento do software, por exemplo: cadastrar clientes. Os requisitos funcionais podem ser executados pelo usuário ou de forma automatizada pelo sistema.

Contudo, foram definidos os seguintes requisitos funcionais para o projeto:

- **Requisito funcional 1 –** Efetuar *login* utilizando CPF e senha;
- **Requisito funcional 2 –** Visualizar todos os vestibulares abertos;
- **Requisito funcional 3 –** Visualizar vestibulares passados e atuais no qual esteja inscrito;
- **Requisito funcional 4 –** Efetuar *download* de documentos;
- **Requisito funcional 5 –** Acompanhar a inscrição no vestibular;
- **Requisito funcional 6 –** Visualizar boletim de desempenho;
- **Requisito funcional 7 –** Visualizar locais de prova;
- **Requisito funcional 8 –** Visualizar datas de prova;
- **Requisito funcional 9 –** Receber notificações;
- **Requisito funcional 10 –** Visualizar notas finais.

3.1.2. REQUISITOS NÃO-FUNCIONAIS

Os requisitos não funcionais ou requisitos indiretos, são aqueles que não tem relação direta com funções do software, estando relacionados ao uso da aplicação em termos de manutenção, performance, segurança ou disponibilidade do software, por exemplo: o sistema deve estar disponível para Windows 11 e MacOS Monterey. Os requisitos não funcionais são implícitos no software, sendo dispensada a visualização direta dos mesmos.

Dito isso, foram estabelecidos requisitos não-funcionais:

- **Requisito não-funcional 1 –** Disponibilidade para a plataforma Android.
- **Requisito não-funcional 2 –** Assegurar os dados dos usuários.
- **Requisito não-funcional 3 –** Utilizar o *framework* React Native para a construção da interface gráfica.
- **Requisito não-funcional 4 –** Utilizar o ambiente de execução de código NodeJS para a construção do servidor da aplicação.

3.2. ARQUITETURA

A arquitetura do sistema foi concebida para ser uma solução eficiente e escalável, permitindo uma interação fluida entre o front-end e o back-end e garantindo uma comunicação segura e rápida com o banco de dados. O sistema foi dividido em duas partes principais, o front-end e o back-end, cada um com funções e responsabilidades distintas, facilitando a manutenção e o desenvolvimento independente de cada módulo.

3.2.1. VISÃO GERAL

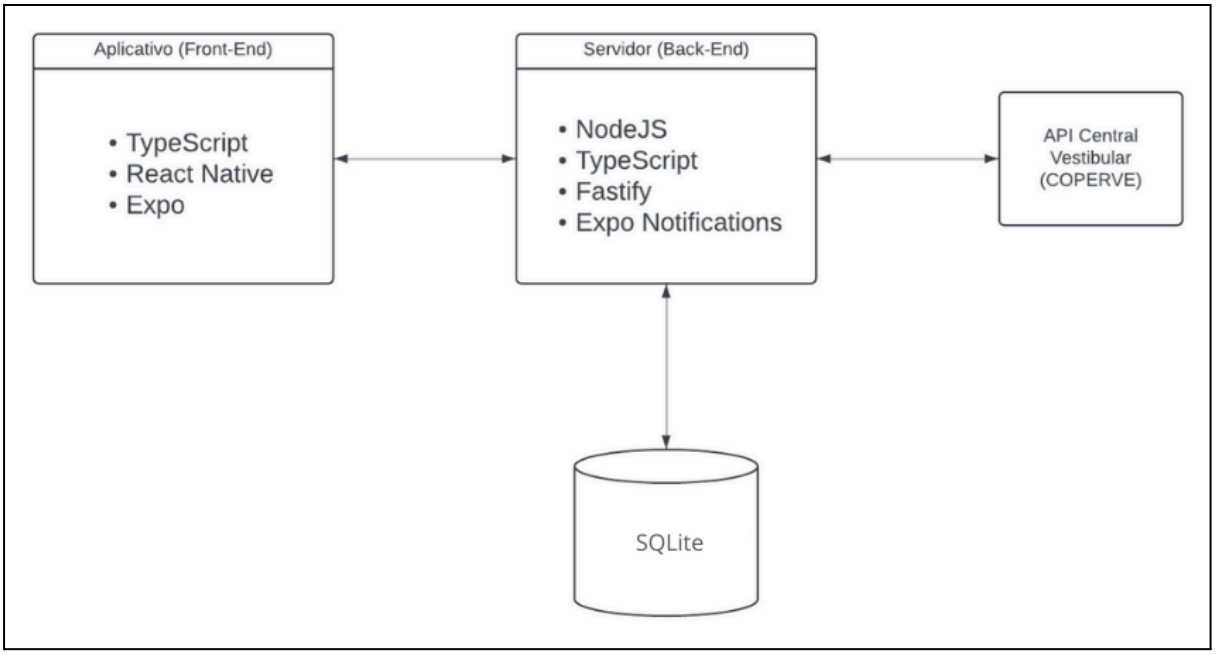
O sistema proposto segue uma arquitetura cliente-servidor, na qual o front-end é responsável pela interface do usuário, enquanto o back-end gerencia a lógica de negócios e o armazenamento de dados.

O front-end, desenvolvido com React Native, interage com o back-end por meio de chamadas de API, enviando e recebendo dados conforme necessário. O

back-end foi construído utilizando NodeJS com o framework Fastify para gerenciar as requisições HTTP e controlar o fluxo de dados para e do banco de dados. O banco de dados utilizado é o SQLite, que armazena as informações do sistema em tabelas estruturadas e relacionadas.

A figura 5 apresenta o diagrama de arquitetura do sistema que ilustra a comunicação entre o front-end e o back-end, e como o back-end interage com o banco de dados. Esse diagrama representa um fluxo simplificado da comunicação no sistema, onde o front-end realiza chamadas para a API no back-end, que por sua vez consulta ou atualiza o banco de dados conforme necessário, ou consulta a API Central da COPERVE, no qual contém todas as informações de usuários e vestibulares.

Figura 5: Arquitetura de comunicação do projeto.



Fonte: Elaborado pelo autor.

3.2.2. FRONT-END

O front-end foi construído com o objetivo de ser intuitivo, responsivo e fácil de usar. A tecnologia React Native foi escolhida devido à sua flexibilidade e à sua capacidade de criar interfaces dinâmicas com alta performance. A arquitetura do front-end é componentizada, ou seja, cada elemento da interface (como formulários, botões e listas) é criado como um componente independente e reutilizável, o que facilita a manutenção e a escalabilidade do sistema.

3.2.3. BACK-END

O back-end foi desenvolvido em NodeJS e é responsável por processar as requisições do front-end, executar a lógica de negócios e interagir com o banco de dados. O uso do Fastify proporciona uma estrutura eficiente para criação de APIs RESTful, que facilita a comunicação entre o front-end e o banco de dados. Essa arquitetura permite que o sistema seja escalável e facilmente mantido, pois o back-end está organizado em camadas distintas, separando responsabilidades como:

- **Roteamento**: Define as rotas da aplicação, que correspondem aos diferentes endpoints acessados pelo front-end.
- **Controladores**: Recebem as requisições, aplicam a lógica de negócios necessária e enviam respostas de volta ao cliente.
- **Serviços**: Implementam a lógica de negócio, como o gerenciamento de usuários, notificações, e validação de dados.

O back-end também é responsável por implementar a segurança da aplicação, garantindo que apenas usuários autenticados e autorizados tenham acesso a informações sensíveis. A autenticação é realizada, por exemplo, com tokens JWT (JSON Web Token), permitindo que o sistema controle o acesso aos recursos de maneira segura e eficiente.

3.2.4. BANCO DE DADOS

Para o armazenamento de dados, foi escolhido um banco de dados relacional que organiza as informações em tabelas relacionadas. O modelo de dados foi projetado para otimizar a consulta e a integridade dos dados, facilitando a

implementação de relacionamentos entre entidades do sistema, como usuários e notificações.

O banco de dados possui três tabelas principais:

- **tb_user**: Armazena informações dos usuários, como seu identificador, CPF e um possível token de autenticação.
- **tb_notification**: Armazena as notificações que são enviadas aos usuários, com campos para título e mensagem.
- **rl_user_notification**: Tabela de relacionamento entre tb_user e tb_notification, que permite que um usuário tenha várias notificações e uma notificação possa ser associada a vários usuários.

Essas tabelas estão inter-relacionadas para garantir que cada usuário possa receber várias notificações e que uma notificação possa ser vinculada a múltiplos usuários. Esse modelo de dados foi escolhido para assegurar flexibilidade e eficiência, de forma que o sistema possa gerenciar notificações de maneira prática.

3.3. WIREFRAMES

Wireframes são representações visuais simplificadas das telas de um aplicativo, focado na estrutura e no layout, sem se preocupar com aspectos de design detalhados, como cores e tipografia. Eles são ferramentas essenciais no processo de design de interfaces, pois permitem a visualização inicial do posicionamento e da organização dos elementos na tela, facilitando a comunicação entre designers, desenvolvedores e outros *stakeholders* do projeto.

Os *wireframes* servem como um guia para a construção do produto final, ajudando a identificar possíveis problemas de usabilidade e a ajustar a funcionalidade antes de investir tempo e recursos no desenvolvimento completo.

Por serem esquemáticos e de fácil modificação, os *wireframes* permitem ajustes rápidos e iterativos, garantindo que as necessidades dos usuários e os requisitos do projeto sejam atendidos de maneira eficaz.

Esses esboços são fundamentais para o planejamento do fluxo de navegação e para a definição das interações básicas entre as telas, proporcionando uma base sólida sobre a qual o design visual e o desenvolvimento funcional serão

construídos. Ao estabelecer claramente a hierarquia de informações e a disposição dos componentes, os *wireframes* ajudam a alinhar a equipe em torno de uma visão comum, acelerando o processo de design e desenvolvimento e reduzindo o risco de retrabalho.

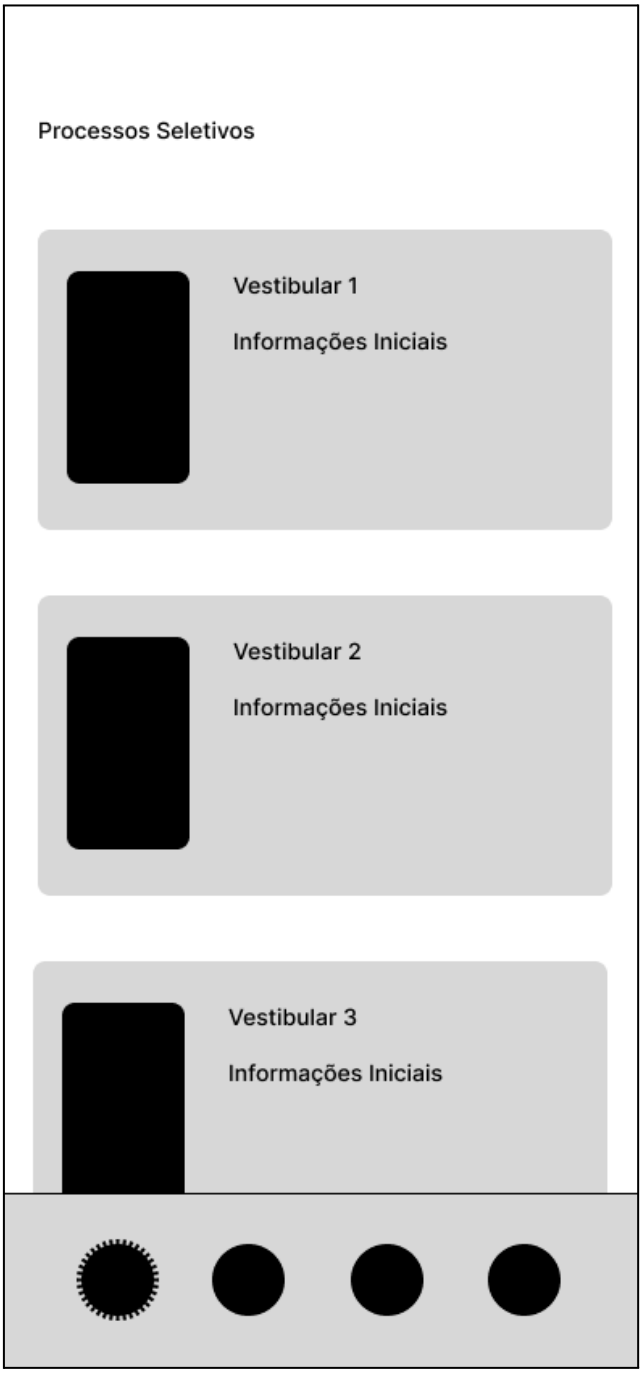
A seguir, serão apresentados os *wireframes* desenvolvidos para o projeto.

Figura 6: *Wireframe* da tela de abertura do aplicativo.

Fonte: Elaborado pelo autor.

Na Figura 6 é exibida a tela de abertura, que será apresentada apenas uma vez ao inicializar o aplicativo.

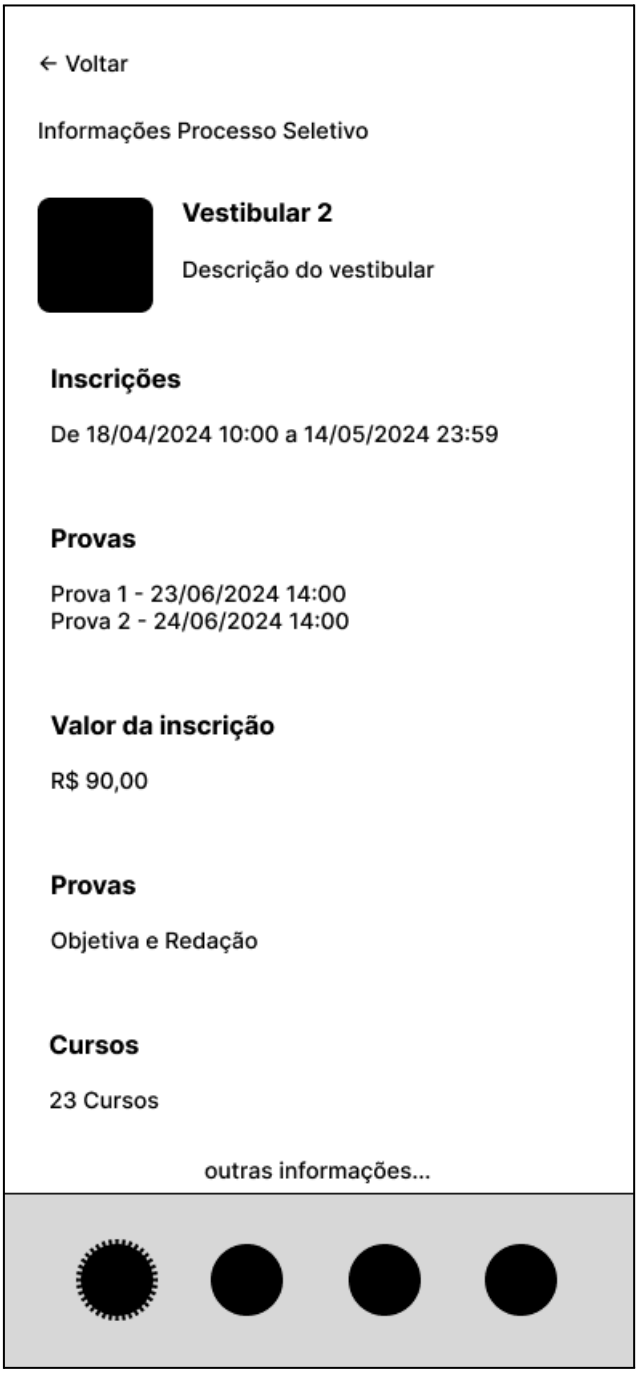
Figura 7: *Wireframe* da tela inicial do aplicativo.



Fonte: Elaborado pelo autor.

A tela inicial do aplicativo, ilustrada na Figura 7, será responsável por exibir os vestibulares atuais ou, em caso do usuário estar autenticado, exibir vestibulares em que ele esteja inscrito ou já participou.

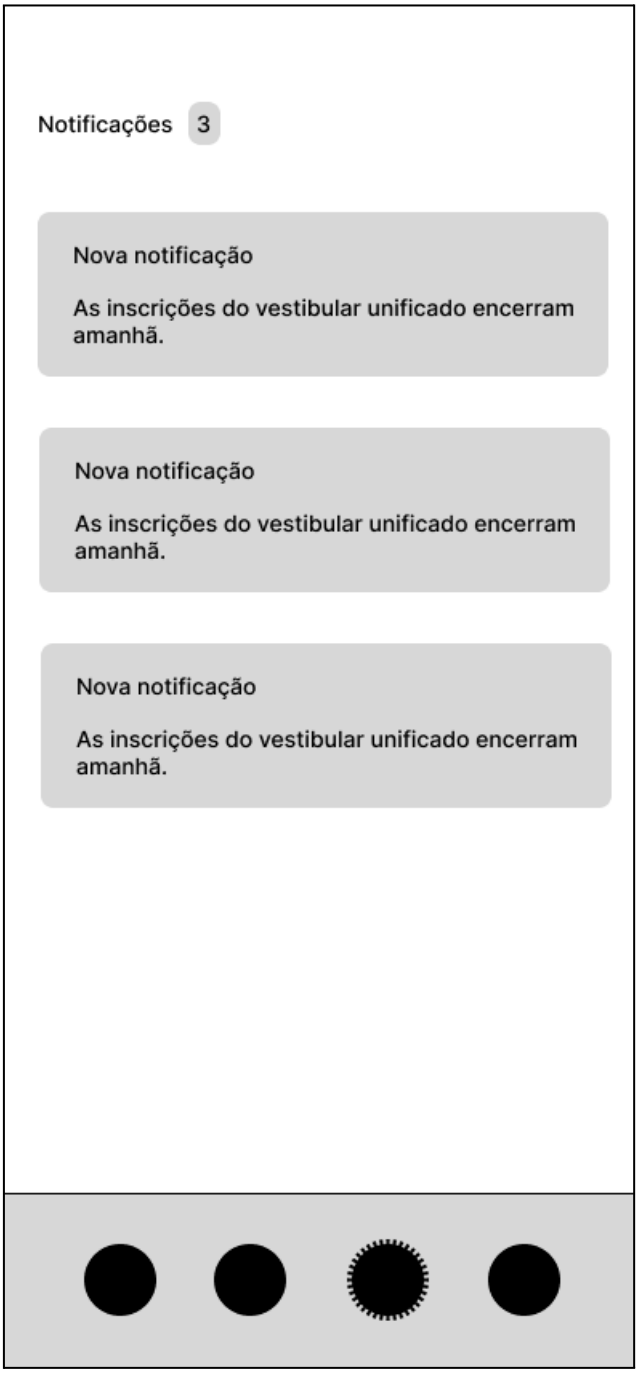
Figura 8: *Wireframe* da tela de informações detalhadas de um vestibular.



Fonte: Elaborado pelo autor.

Na Figura 8 é exibida a tela responsável por mostrar as informações detalhadas de um vestibular em específico, como a data de inscrição, datas de prova, custo de inscrição, entre outros.

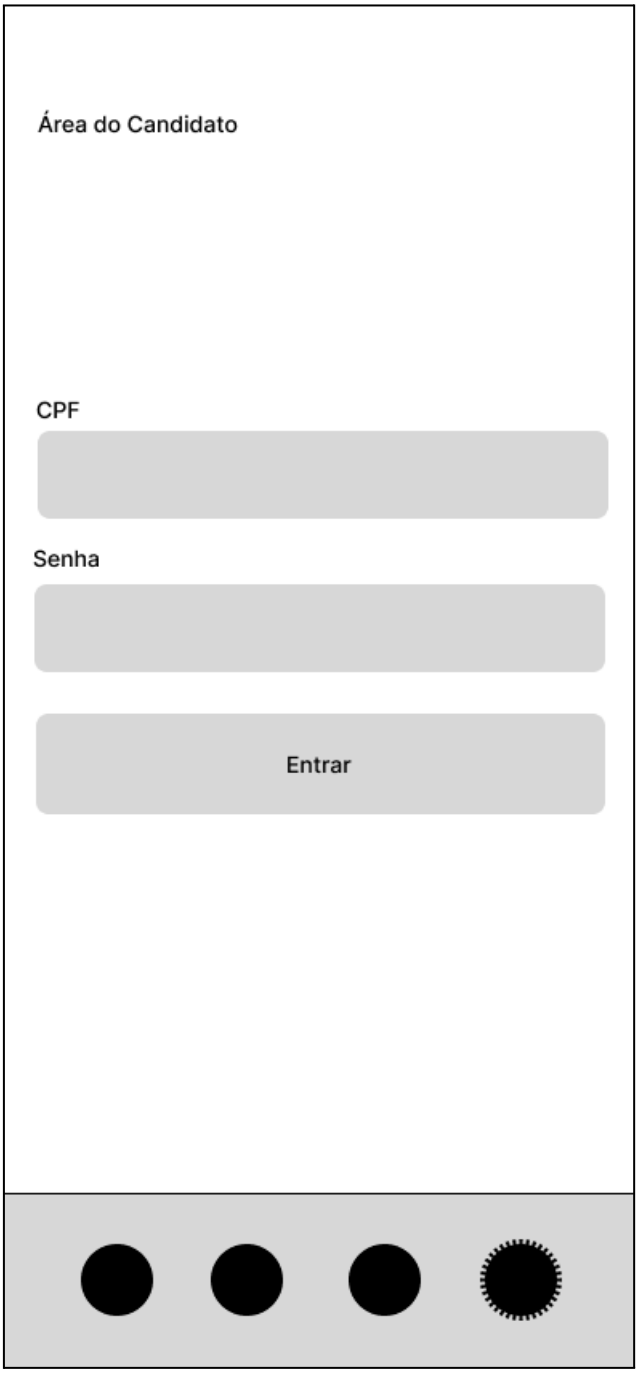
Figura 9: *Wireframe* da tela de notificações.



Fonte: Elaborado pelo autor.

A Figura 9 mostra a tela responsável por centralizar e exibir todas as notificações recebidas pelo usuário. Isso inclui avisos importantes, como encerramento de inscrições e alertas de aprovação em vestibulares. Dessa forma, o usuário poderá acompanhar todas as informações relevantes de maneira organizada e eficiente.

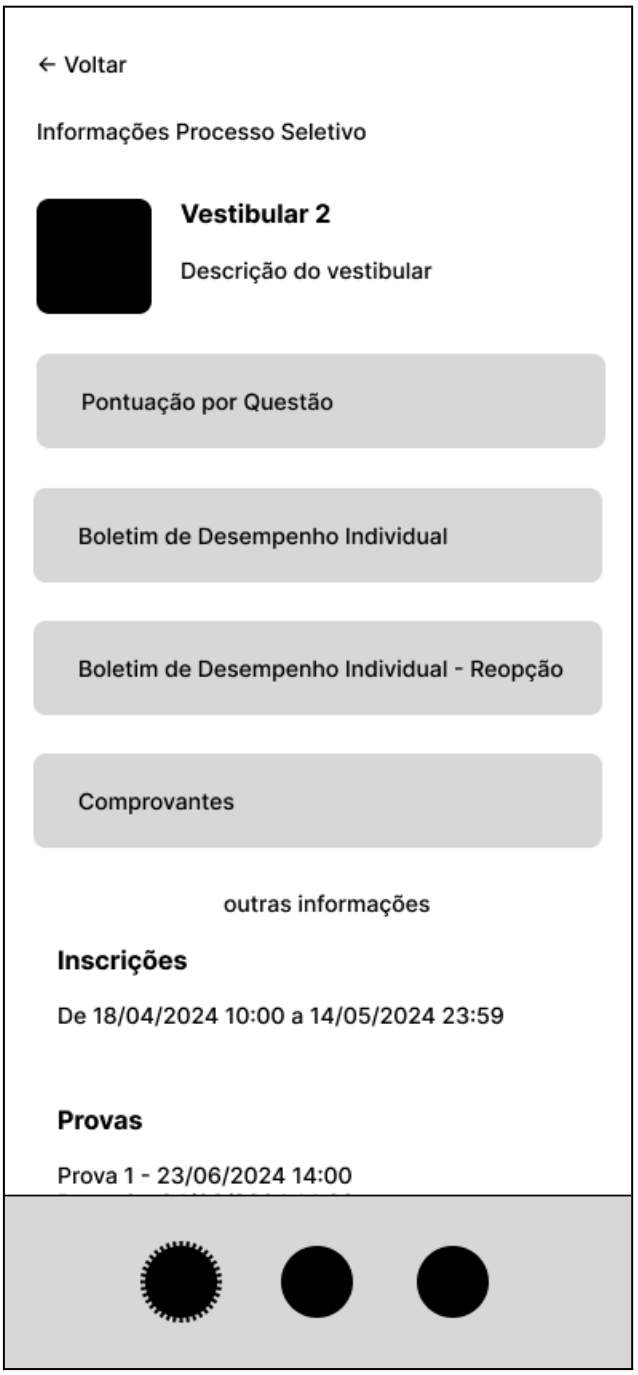
Figura 10: *Wireframe* da tela de login.



Fonte: Elaborado pelo autor.

A Figura 10 exibe a tela responsável por realizar a autenticação do usuário no aplicativo por meio do CPF e senha do candidato.

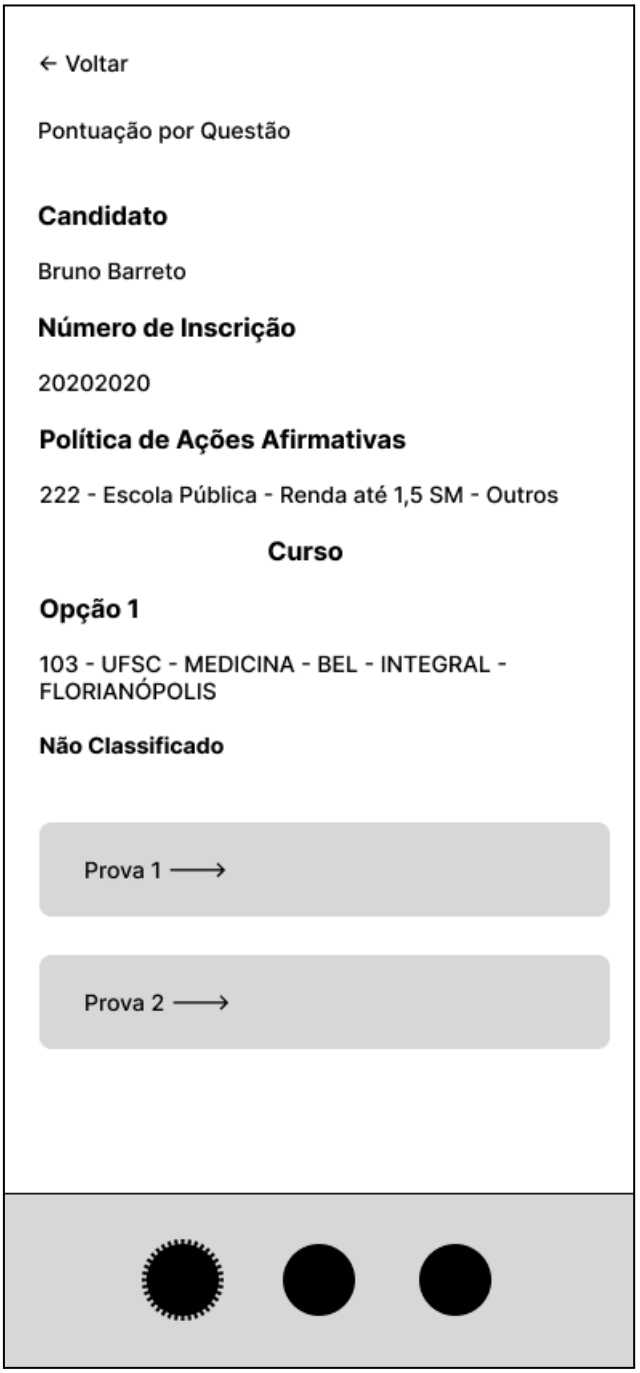
Figura 11: *Wireframe* da tela de informações detalhadas de um vestibular no qual o candidato está inscrito.



Fonte: Elaborado pelo autor.

Semelhante à tela de informações detalhadas de um vestibular, a Figura 11 representa a tela exibida caso o usuário esteja inscrito no vestibular. Ela pode apresentar novas informações, como comprovantes de inscrição, além de pontuações e desempenho, caso as provas já tenham sido realizadas.

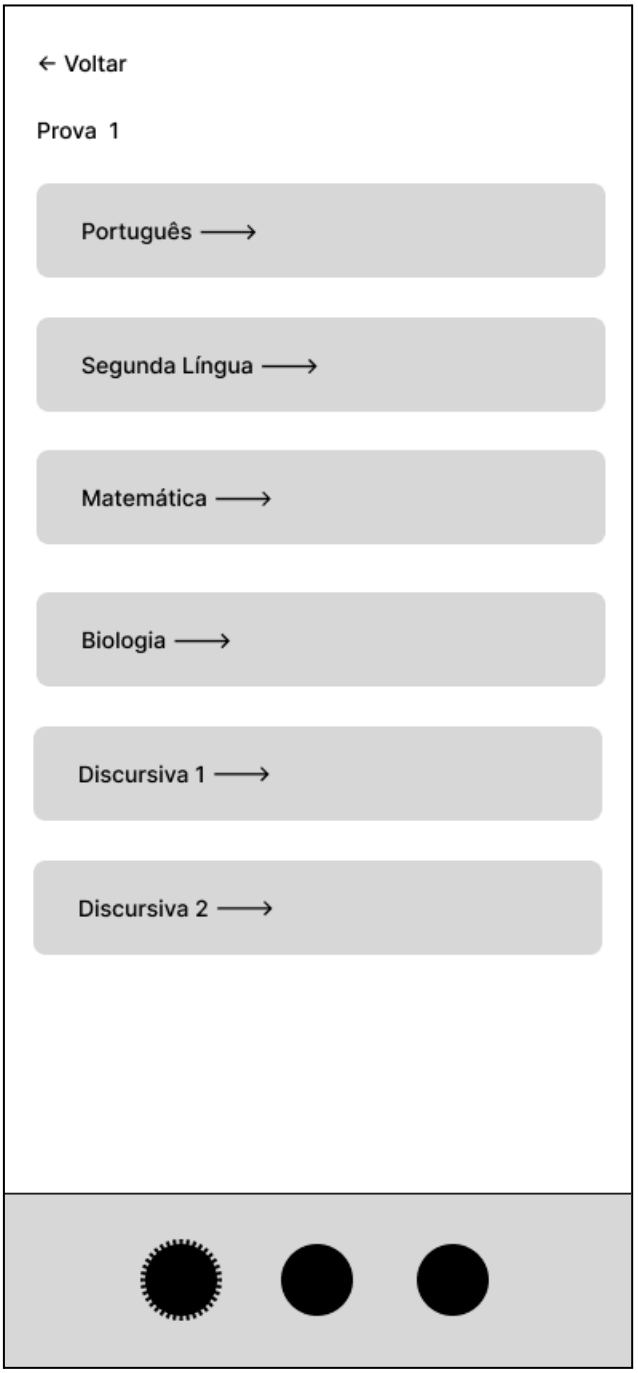
Figura 12: *Wireframe* da tela inicial de pontuação por questão.



Fonte: Elaborado pelo autor.

A Figura 12 ilustra a tela inicial de pontuação por questão, antecessora à tela de seleção de disciplina e utilizada para a escolha de provas a serem visualizadas.

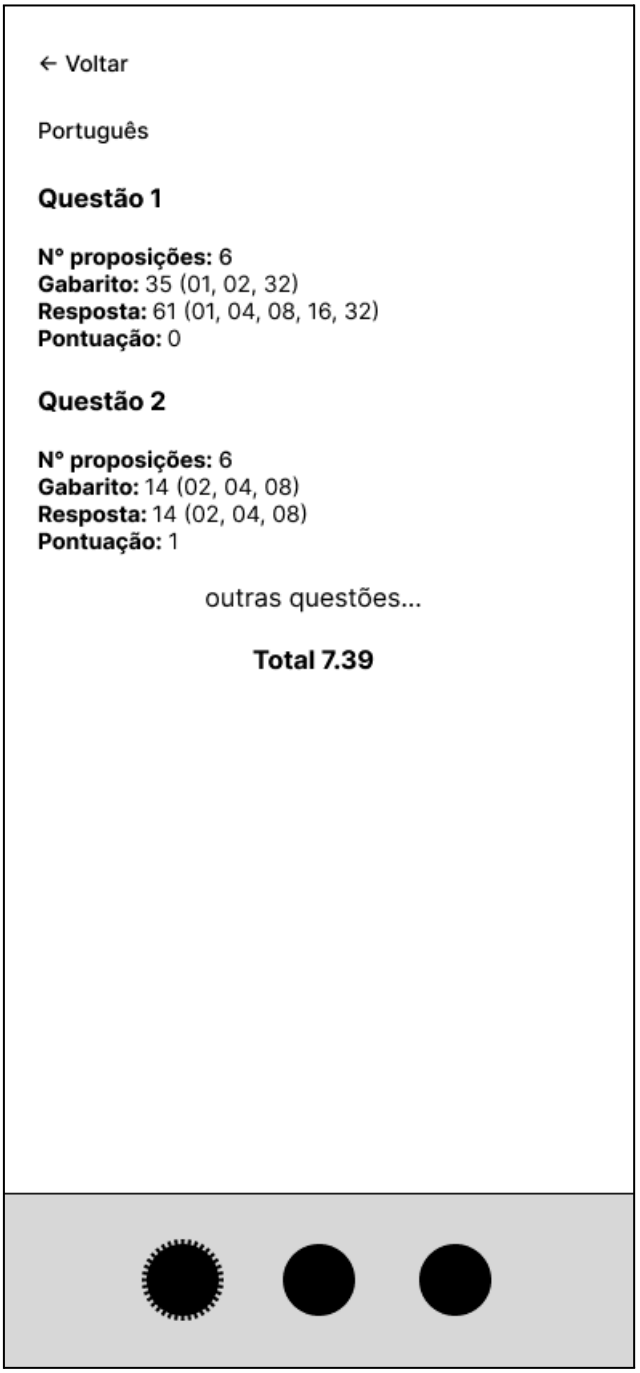
Figura 13: *Wireframe* da tela seleção de disciplina para pontuação.



Fonte: Elaborado pelo autor.

A Figura 13 exibe a tela de seleção de disciplina para a visualização da pontuação da mesma, responsável por organizar as pontuações das disciplinas individualmente.

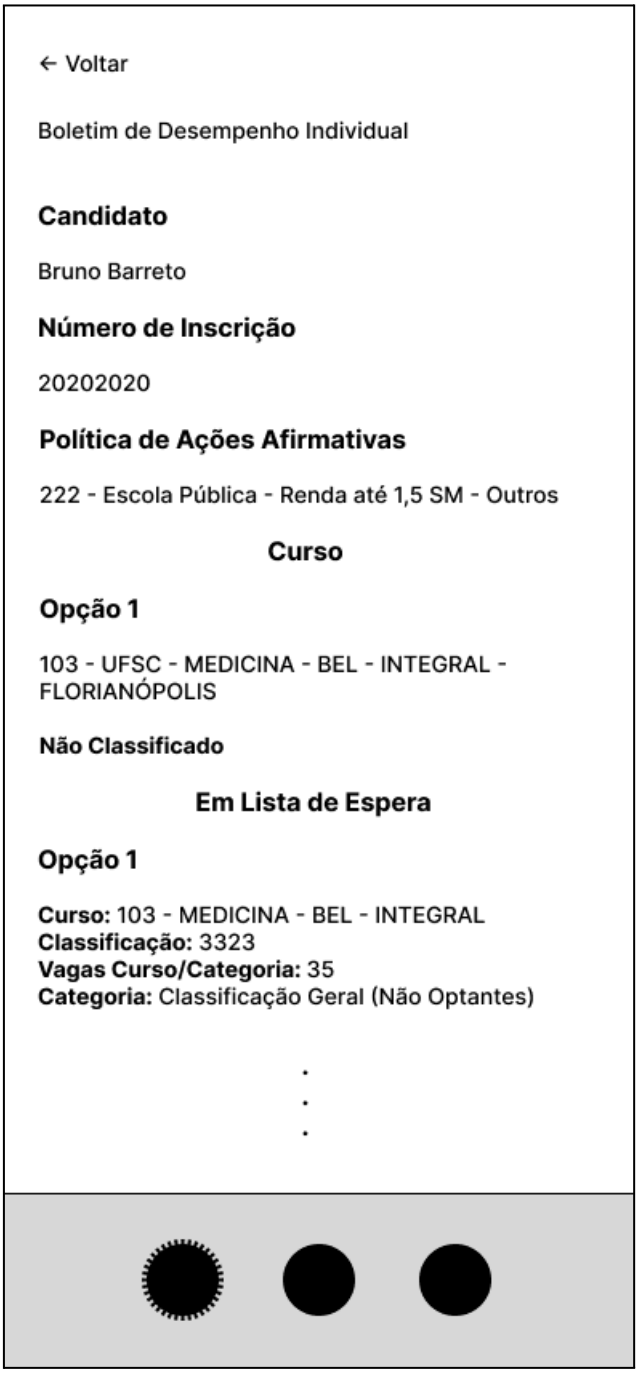
Figura 14: *Wireframe* da tela de pontuação por questão na disciplina.



Fonte: Elaborado pelo autor.

A tela principal de pontuação por questão, ilustrada na Figura 14, exibe todas as questões da disciplina que foi previamente selecionada. Nesta tela são mostradas as respostas do candidato por questão e o gabarito correto, assim como a nota final na disciplina correspondente.

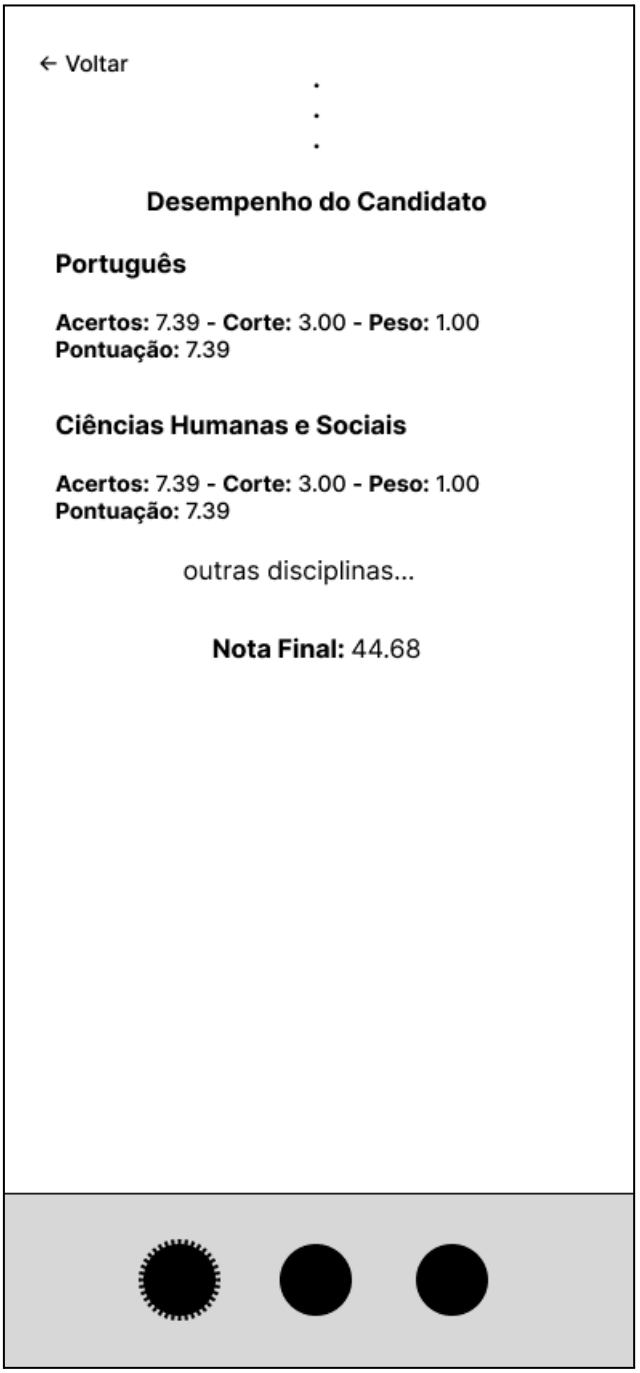
Figura 15: *Wireframe* da tela de boletim de desempenho individual - 1.



Fonte: Elaborado pelo autor.

A Figura 15 exibe a tela responsável por mostrar o desempenho em geral do candidato, indicando o curso e as notas finais em cada disciplina.

Figura 16: *Wireframe* da tela de boletim de desempenho individual - 2.



Fonte: Elaborado pelo autor.

A Figura 16 representa a continuação da tela anterior (Figura 17).

Figura 17: *Wireframe* da tela de download de comprovantes.

Fonte: Elaborado pelo autor.

A Figura 17 exibe a tela responsável por reunir todos os documentos disponíveis para *download* de um determinado vestibular.

4. DESENVOLVIMENTO E IMPLEMENTAÇÃO

Esse capítulo tem o objetivo de documentar o processo de desenvolvimento do sistema, abordando as escolhas de tecnologias, a estrutura do código, e as principais funcionalidades implementadas tanto no front-end quanto no back-end. Essa seção será mais detalhada e técnica, apresentando como a arquitetura definida anteriormente foi materializada.

4.1. AMBIENTE DE DESENVOLVIMENTO

Para o desenvolvimento do projeto, utilizamos a **IDE VSCode**, que proporciona uma experiência de desenvolvimento ágil e simples. O **npm**, ferramenta do Node.js, foi empregado para a criação e gerenciamento das dependências do projeto, tanto para o back-end quanto para o front-end. Para testar as rotas HTTP durante o desenvolvimento, antes mesmo de ter o front-end implementado, foi utilizada a ferramenta **Postman**. Além de ser útil para testar as rotas da API central da COPERVE, o Postman facilitou a comunicação entre o back-end e o front-end, permitindo a validação das funcionalidades do sistema.

4.2. MODELO DE DADOS E RELACIONAMENTOS

Após as configurações iniciais, foi modelado uma estrutura inicial do banco de dados, a fim de armazenar as informações necessárias no sistema, sendo elas o login do usuário e notificações a serem enviadas para os usuários.

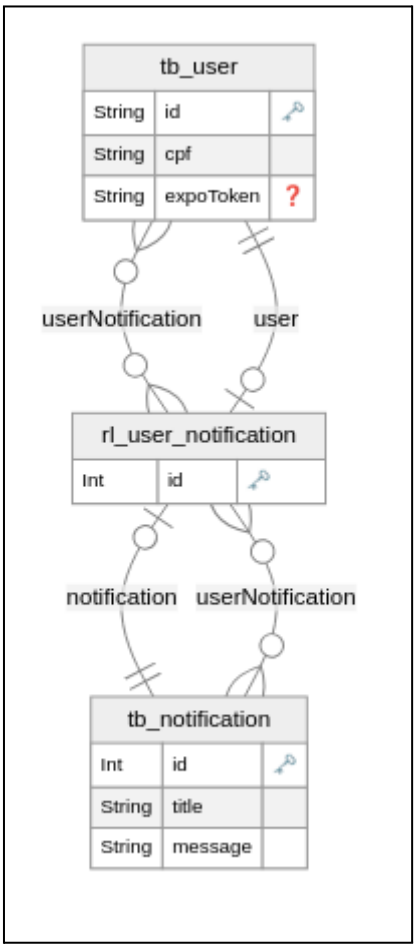
O sistema utiliza um banco de dados relacional, composto por três tabelas principais:

- **tb_user**: Tabela que armazena os dados dos usuários.
    - id: Identificador único de cada usuário, utilizado para referenciá-los em outros relacionamentos.
    - cpf: Número de CPF do usuário, utilizado para validação e identificação adicional.

    - expoToken: Token utilizado para o envio de notificações push ao dispositivo do usuário, caso o sistema integre com notificações móveis.
- **tb_notification**: Tabela que armazena informações sobre as notificações enviadas.
    - id: Identificador único de cada notificação.
    - title: Título da notificação, utilizado para resumir o assunto da mensagem.
    - message: Mensagem completa da notificação, contendo o conteúdo informativo que será exibido ao usuário.
- **rl_user_notification**: Tabela intermediária que cria um relacionamento many-to-many entre usuários e notificações, permitindo que um usuário receba várias notificações e que cada notificação seja enviada para múltiplos usuários.
    - id: Identificador único do relacionamento, que pode ser utilizado para auditoria ou controle do sistema.
    - userId: Identificador único do usuário.
    - notificationId: Identificador único da notificação.

A figura 18 apresenta o modelo ER (Entidade-Relacionamento) do banco de dados, exibindo as tabelas **tb_user**, **tb_notification** e **rl_user_notification**, bem como os relacionamentos entre elas.

Figura 18: Modelo ER do banco de dados do sistema.



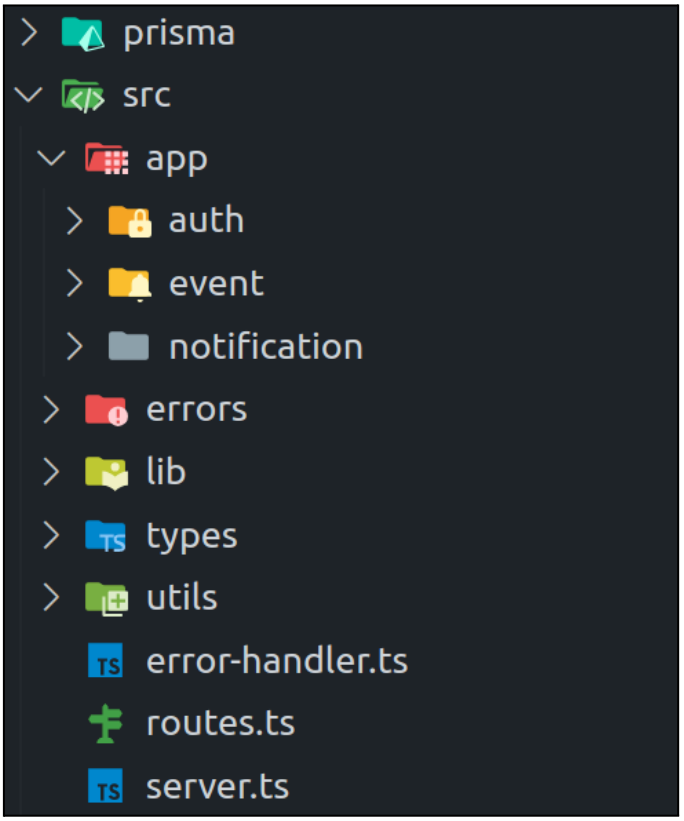
Fonte: Elaborado pelo autor.

4.3. IMPLEMENTAÇÃO DO BACKEND

Após a criação do projeto backend com **npm**, foram instalados pacotes essenciais para o desenvolvimento, com destaque para o **Fastify**, que é fundamental para a criação de um servidor HTTP de alto desempenho. Também foi utilizado o **Prisma ORM**, que facilita a criação automatizada das tabelas no banco de dados e o gerenciamento de migrações. Para realizar chamadas HTTP à API central da COPERVE, foi incorporado o **Axios**, permitindo a autenticação e a busca de informações. A instalação do **TypeScript** no projeto também foi necessária, oferecendo maior segurança no desenvolvimento através de tipagens.

Uma estrutura de pastas foi criada para garantir a organização e a manutenção do projeto. A figura 19 apresenta a estrutura de pastas definida para o backend.

Figura 19: Estrutura de pastas para o *Back-End*.



Fonte: Elaborado pelo autor.

- **prisma**: Contém as configurações do Prisma ORM, incluindo as migrações do banco de dados e os esquemas de tabelas.
- **src**: Pasta principal onde é armazenado o código do projeto:
    - **app**: Módulos do projeto, incluindo controllers, services e repositories.
    - **errors**: Funções de exceções personalizadas.
    - **lib**: Configurações básicas, como a comunicação com a API.
    - **types**: Tipagens de objetos que podem ser utilizadas globalmente.
    - **utils**: Funções úteis, reutilizáveis em diferentes partes do projeto.

4.3.1. AUTENTICAÇÃO

Para implementar a autenticação dos usuários, foi utilizada a biblioteca **@fastify/jwt**, que gerencia os tokens JWT. Durante o processo de autenticação, o backend recebe o CPF e a senha do usuário, além do token do Expo, caso o usuário tenha aceitado receber notificações push. Com as credenciais fornecidas, o sistema realiza a autenticação junto à API central da COPERVE. Caso a autenticação seja bem-sucedida, o login do usuário é registrado na tabela **tb_user**, juntamente com o token do Expo, para permitir o envio de notificações push.

4.3.2. EVENTOS

Para a consulta de eventos, foram implementadas algumas rotas:

- **findEvent**: Busca um evento específico pelo ID.
- **getEvents**: Retorna todos os eventos disponíveis.
- **getPerformanceReport**: Retorna as notas de um vestibular realizado por um candidato. Essa rota é privada, ou seja, só pode ser acessada por usuários autenticados.

As rotas **findEvent** e **getEvents** possuem um fluxo semelhante. Caso o usuário tenha fornecido um token de autenticação, o sistema busca e retorna informações do candidato no evento, caso existam. Se o token não for fornecido, o sistema retorna apenas as informações do evento. Todas as rotas realizam chamadas à API central, e as respostas são processadas para retornar apenas os dados necessários ao front-end.

Figura 20: Função de transformação de informações.

```
export const parseBasicEvent = async ({
  rawEvent,
  rawEventCandidate,
}: ParseBasicEventProps): Promise<BasicEvent> ⇒
  Promise.resolve({
    id: rawEvent.codigo_evento,
    eventName: rawEvent.nome,
    registrationStartDate: convertStringToDate(rawEvent.data_inicio_inscricao),
    registrationEndDate: convertStringToDate(rawEvent.data_fim_inscricao),
    examList: parseExamList(rawEvent.provas),
    registered: !!rawEventCandidate,
    registrationPaid: !!rawEventCandidate?.homologado,
    image: rawEvent.imagem,
  });
```

Fonte: Elaborado pelo autor.

4.3.3. NOTIFICAÇÕES

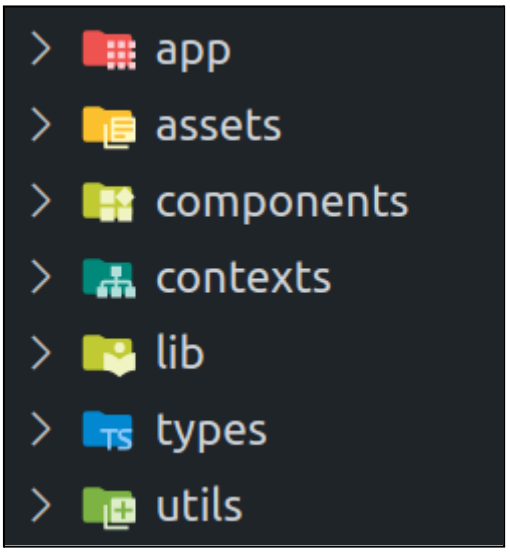
Para o envio de notificações push, foi utilizada a biblioteca **expo-server-sdk**, criada pelo próprio Expo. Com ela, é possível enviar notificações push para dispositivos que utilizam o Expo, sendo necessário apenas o **expoToken**, previamente armazenado no banco de dados.

Mesmo que o usuário não tenha permitido o envio de notificações, ou seja, não tenha fornecido seu **expoToken** durante a autenticação, as notificações serão exibidas na tela de notificações do aplicativo (Figura 9).

4.4. IMPLEMENTAÇÃO DO FRONTEND

Após a criação do projeto **Expo** com o comando **npx create-expo-app**, foi estabelecida a estrutura de pastas para garantir a organização do código do front-end. A figura 21 ilustra a estrutura de pastas definida para o front-end.

Figura 21: Estrutura de pastas para o Front-End.



Fonte: Elaborado pelo autor.

- **app**: Armazena as diferentes telas do aplicativo.
- **assets**: Arquivos como imagens e fontes utilizadas no projeto.
- **components**: Componentes React reutilizáveis.
- **context**: Armazena os contextos React para o gerenciamento de estados globais.
- **lib**: Contém as configurações básicas do projeto, como a comunicação com a API.
- **types**: Tipagens globais de objetos.
- **utils**: Funções utilitárias utilizadas em várias partes do projeto.

Após definir a estrutura de pastas, foram instaladas bibliotecas para otimizar o desenvolvimento:

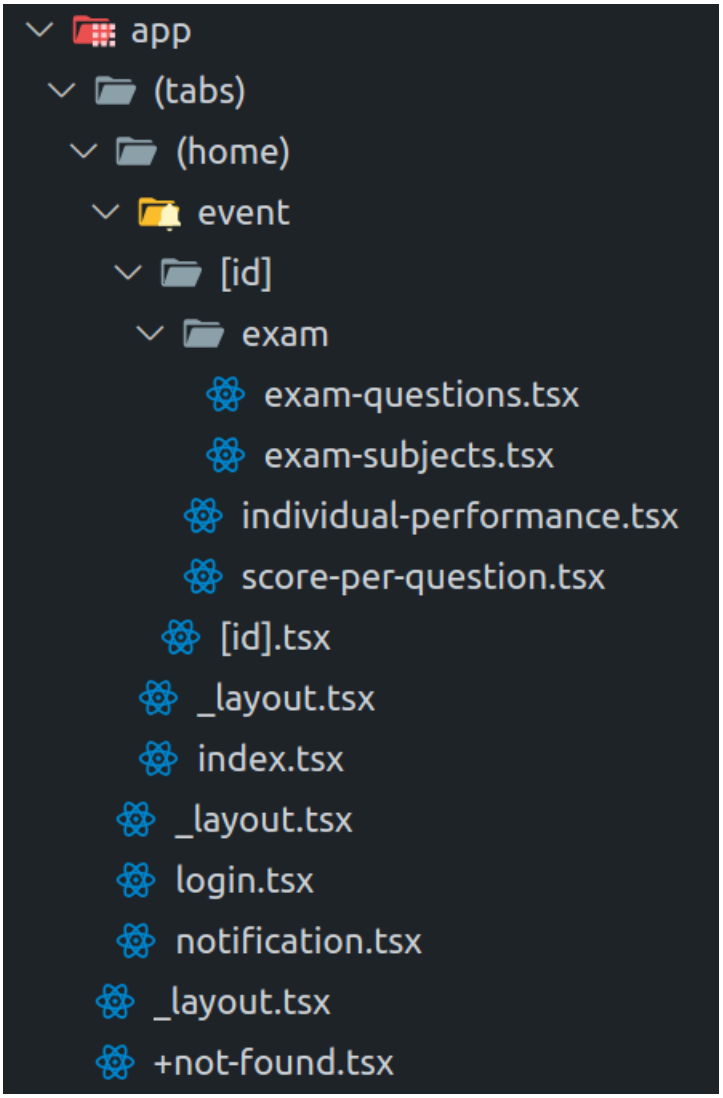
- **nativewind**: Facilita a estilização dos componentes visuais, utilizando classes do **Tailwind CSS** adaptadas para o React Native.
- **react-query** e **axios**: Utilizadas para as chamadas de API.
- **react-hook-form**: Facilita a criação de formulários e a validação de dados.

O **TypeScript** também foi integrado ao projeto, garantindo segurança no desenvolvimento por meio da tipagem estática.

4.4.1. ROTEAMENTO

O gerenciamento de rotas no aplicativo foi feito utilizando o **expo-router**, que está disponível nativamente a partir da versão 50 do Expo. Com esse sistema de roteamento baseado em arquivos, ao criar um arquivo de componente, a rota correspondente é automaticamente gerada.

Figura 22: Roteamento baseado em arquivos do aplicativo Expo.



Fonte: Elaborado pelo autor.

4.4.2. TELAS DO APLICATIVO

As telas do aplicativo foram estilizadas com a biblioteca **nativewind**, que permite a estilização através de classes, oferecendo uma abordagem prática e rápida para o desenvolvimento de interfaces.

Figura 23: Estilização utilizando nativewind.

```
<Image source={HeaderImage} className="absolute" />
      <View className="h-full p-4 pt-16 space-y-4">
        <Text className="text-base font-semibold text-white">
          Comissão Permanente do Vestibular
        </Text>

        <View className="flex-row items-center space-x-2">
          <Feather name="home" color={colors.white} size={16} />
          <Text className="text-white">•</Text>
          <Text className="text-white">Processos Seletivos</Text>
        </View>

        {isLoading ? <LoadingContainer /> : <EventList eventList={eventList} />}
      </View>
```

Fonte: Elaborado pelo autor.

As classes de estilo, como **text-white** para definir a cor do texto, são aplicadas diretamente nos componentes React, tornando o desenvolvimento frontend mais ágil e eficiente.

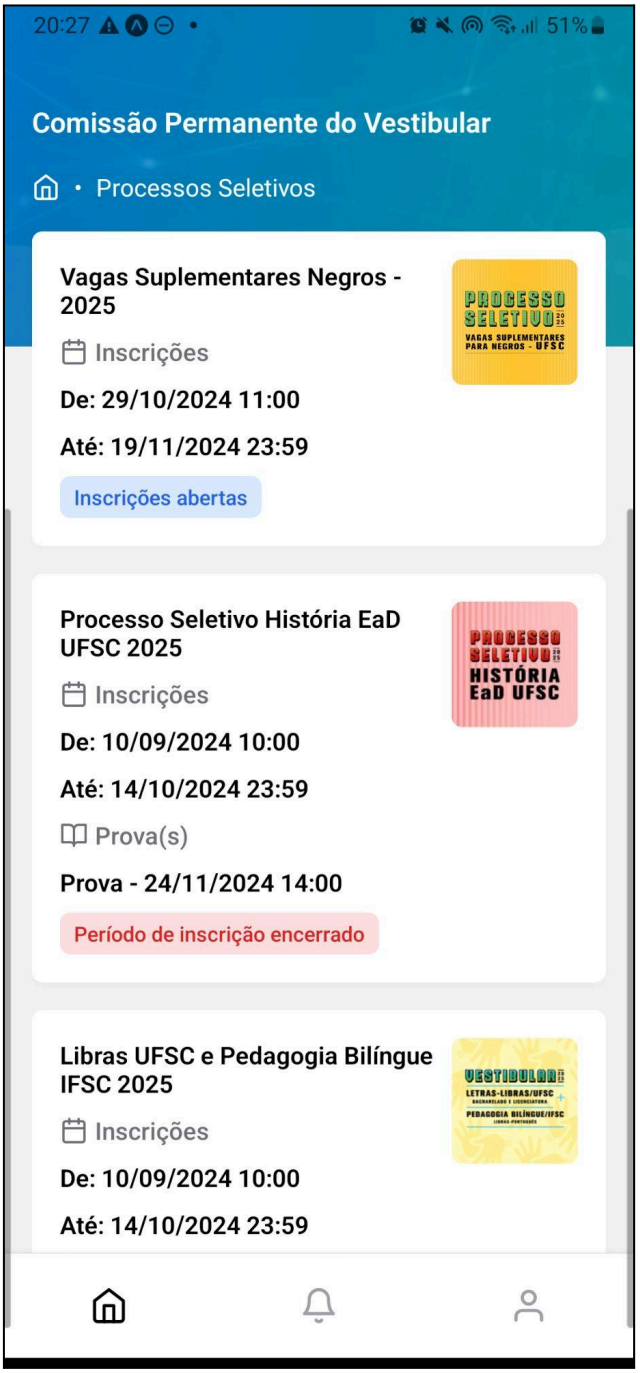
A seguir, é mostrado as telas do aplicativo, já com a estilização finalizada, inicialmente utilizando dados fictícios.

Figura 24:Tela de abertura do aplicativo.

Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela de abertura do aplicativo (Figura 6).

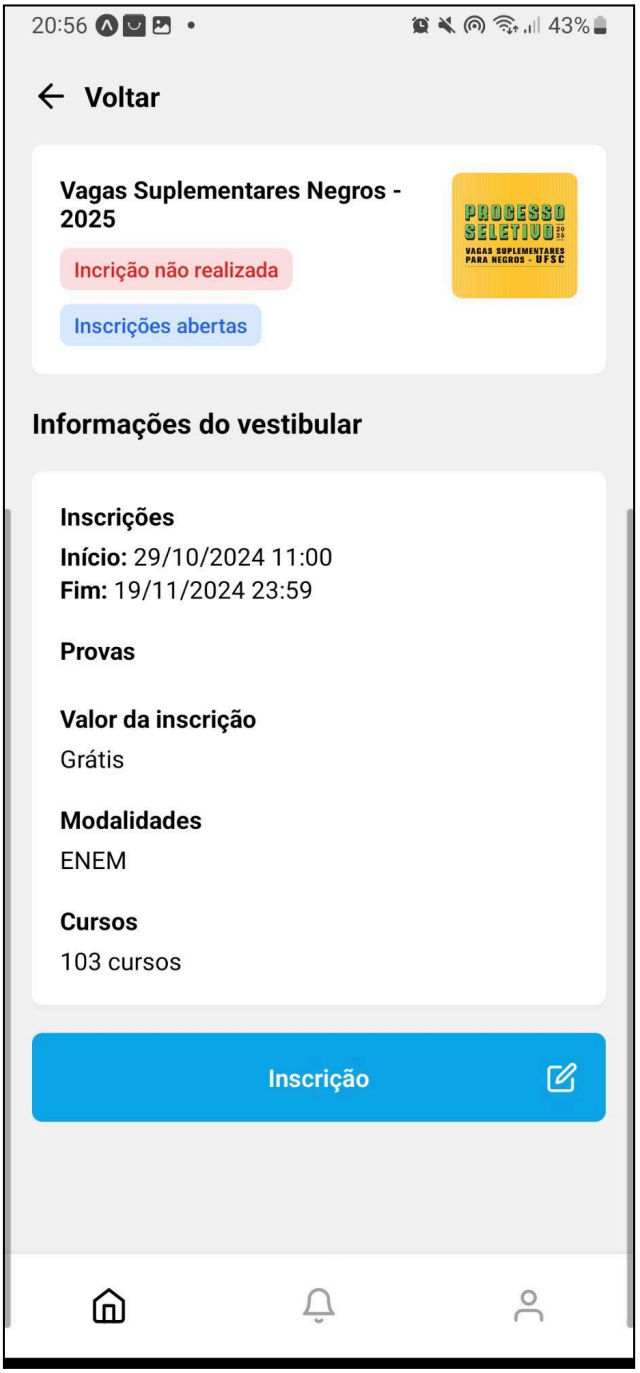
Figura 25: Tela inicial do aplicativo.



Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela inicial do aplicativo (Figura 7).

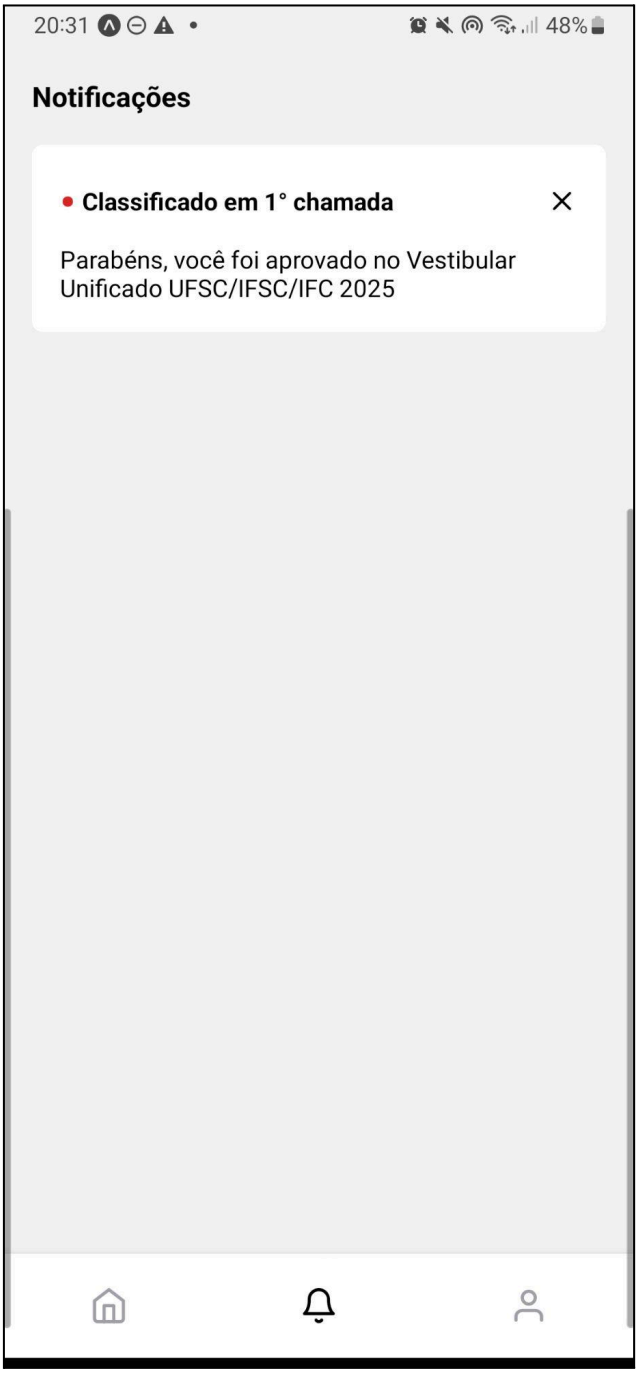
Figura 26: Tela de informações detalhadas de um vestibular.



Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela de informações detalhadas de um vestibular (Figura 8).

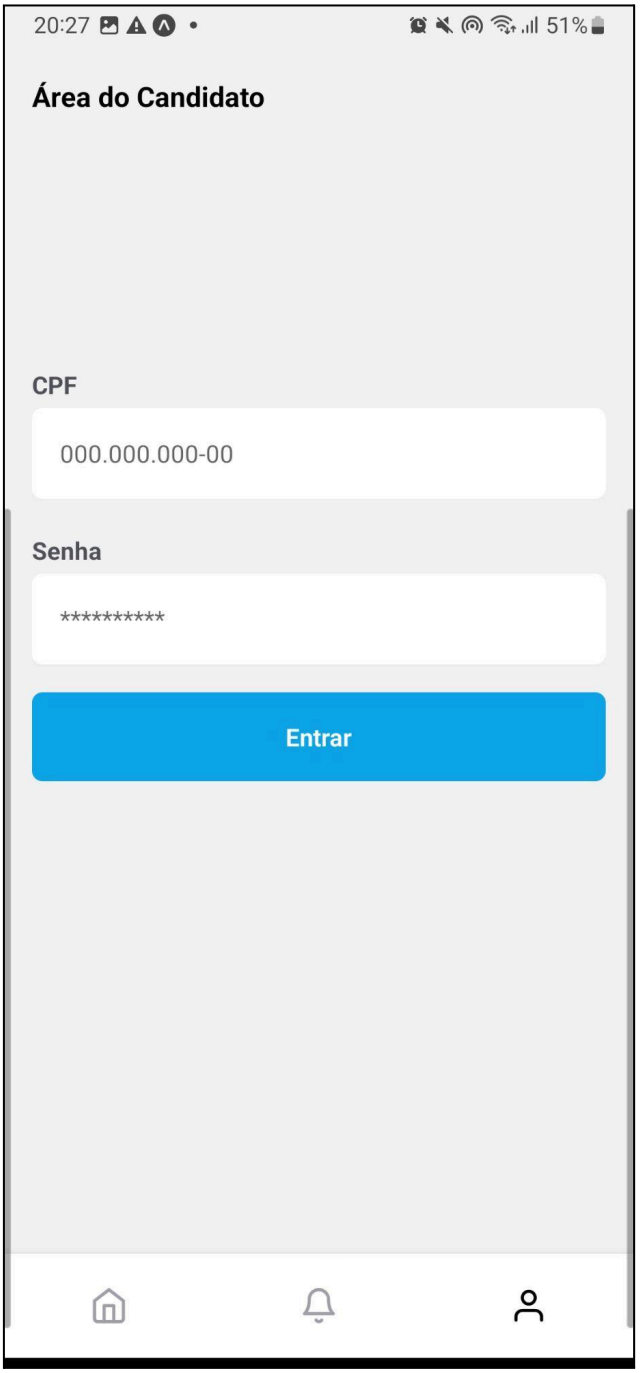
Figura 27: Tela de notificações.



Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela de notificações (Figura 9).

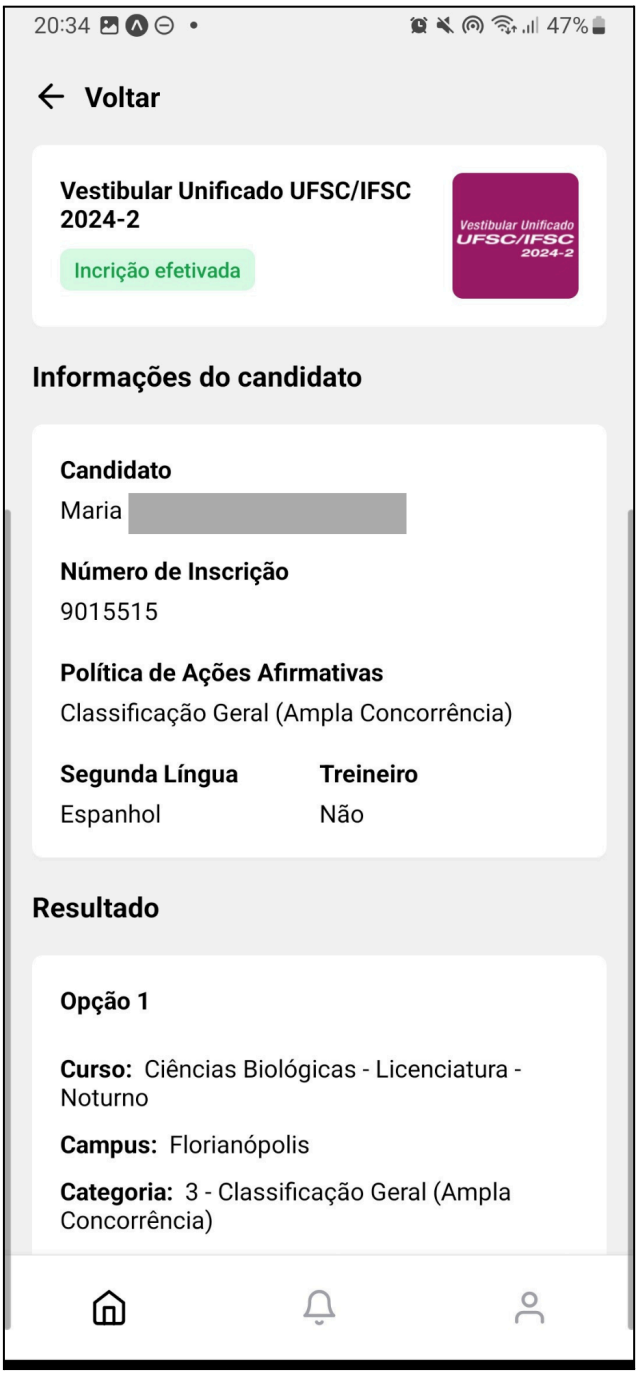
Figura 28: Tela de login.



Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela de login (Figura 10).

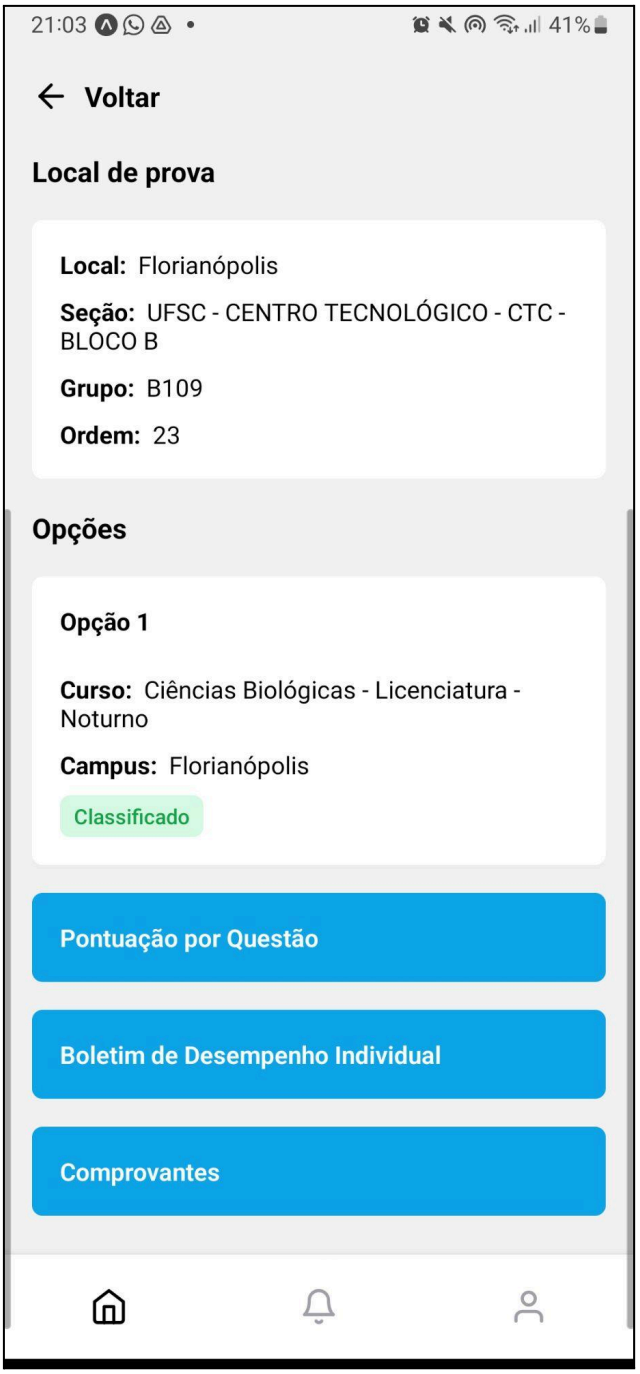
Figura 29: Tela de informações detalhadas de um vestibular no qual o candidato está inscrito - 1.



Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela de informações detalhadas de um vestibular no qual o candidato está inscrito - 1 (Figura 11).

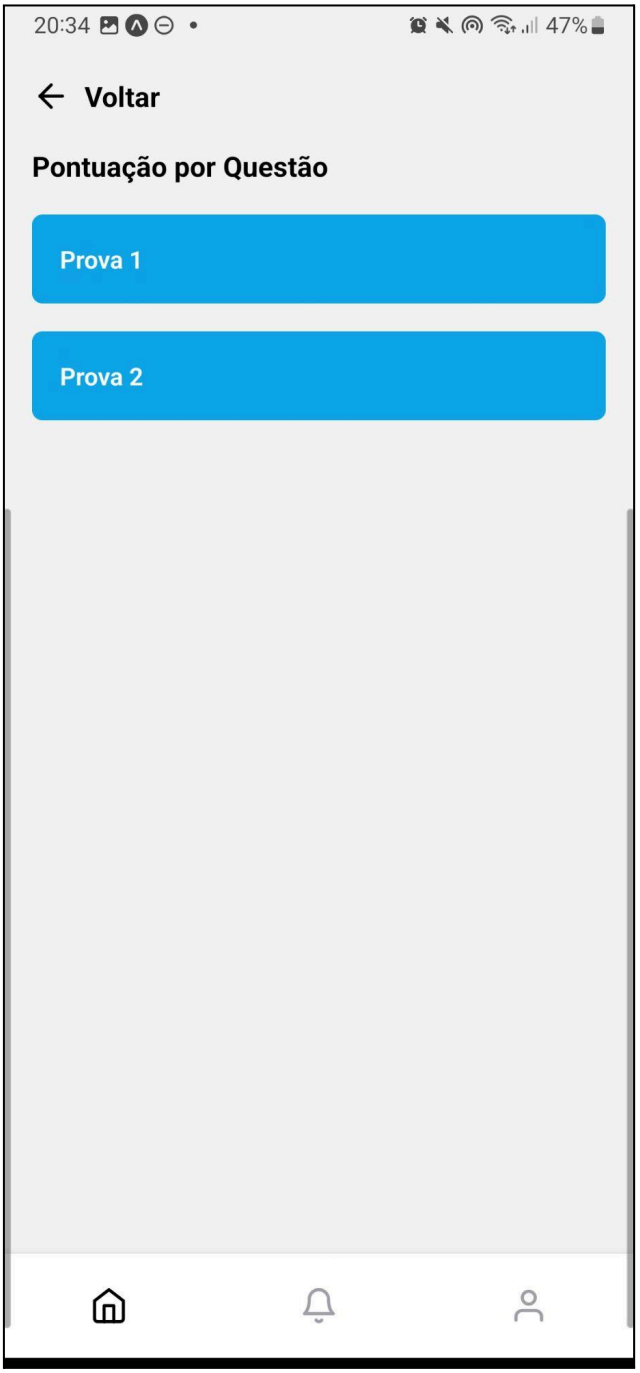
Figura 30: Tela de informações detalhadas de um vestibular no qual o candidato está inscrito - 2.



Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela de informações detalhadas de um vestibular no qual o candidato está inscrito - 2 (Figura 11).

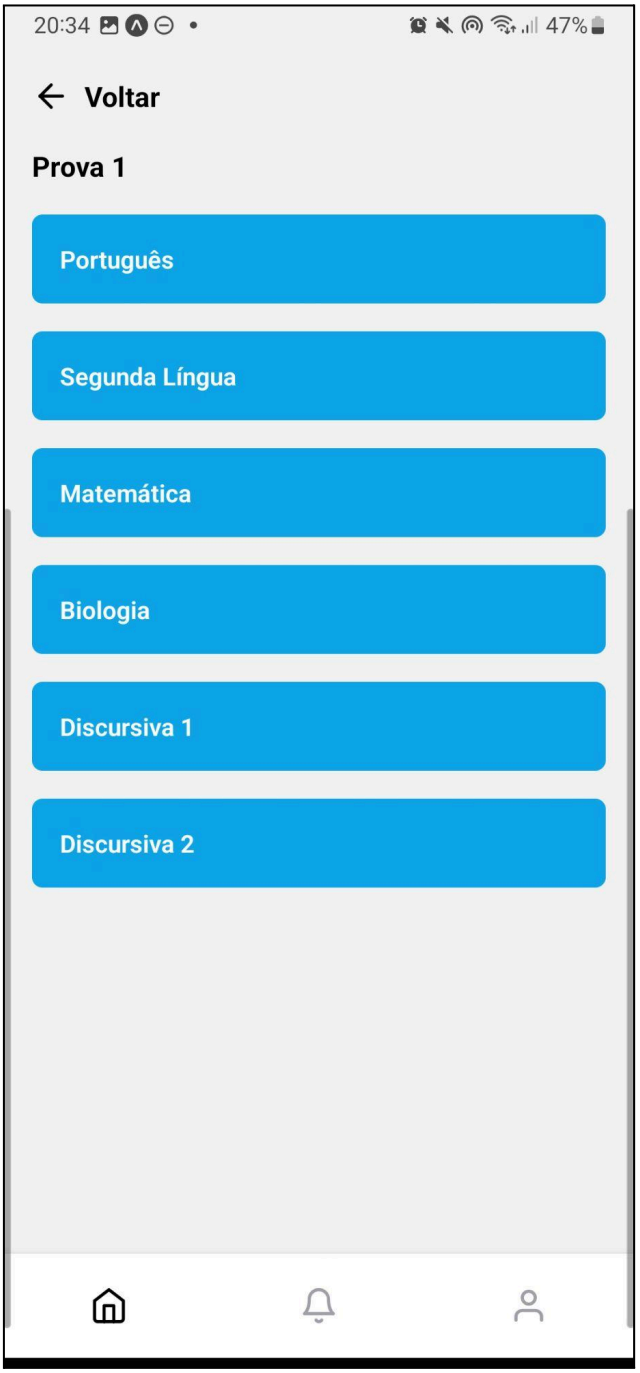
Figura 31: Tela inicial de pontuação por questão.



Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela inicial de pontuação por questão (Figura 12).

Figura 32: Tela seleção de disciplina para pontuação.



Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela seleção de disciplina para pontuação (Figura 13).

Figura 33: Tela de pontuação por questão na disciplina.

Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela de pontuação por questão na disciplina (Figura 14).

Figura 34: Tela de boletim de desempenho individual - 1.

Fonte: Elaborado pelo autor.

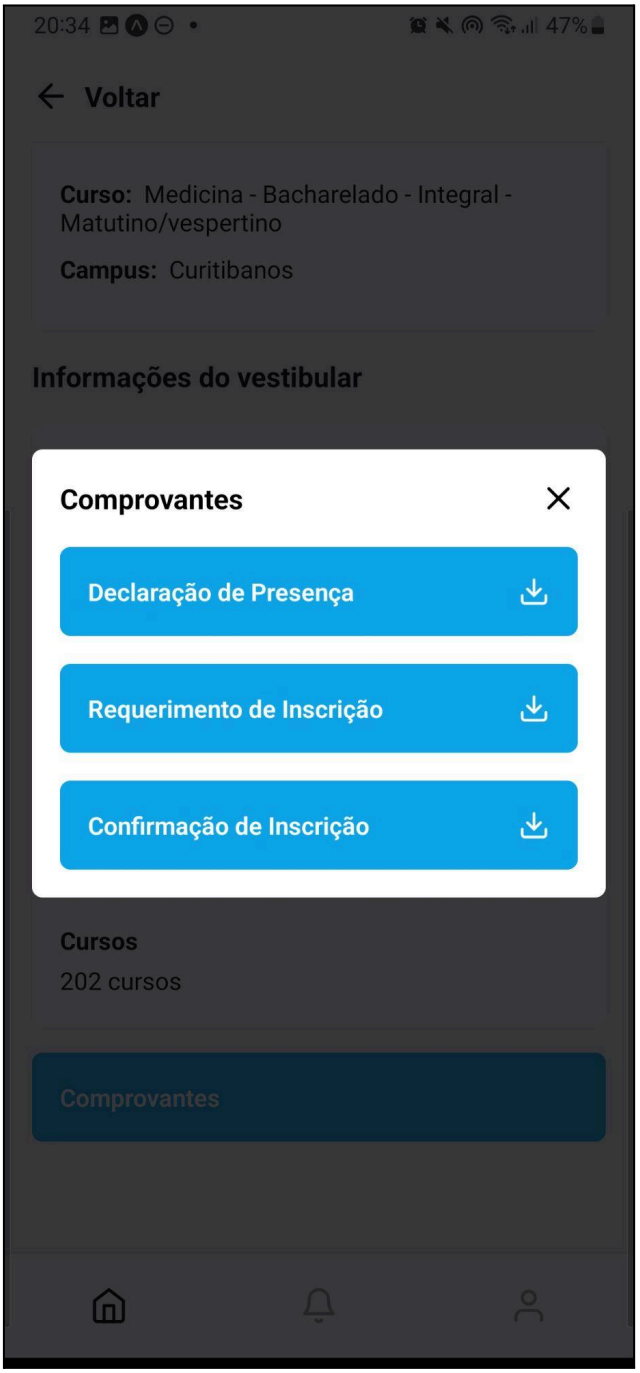
Tela criada a partir do *wireframe* da tela de boletim de desempenho individual - 1 (Figura 15).

Figura 35: Tela de boletim de desempenho individual - 2.

Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela de boletim de desempenho individual - 2 (Figura 16).

Figura 36: Tela de download de comprovantes.



Fonte: Elaborado pelo autor.

Tela criada a partir do *wireframe* da tela de download de comprovantes (Figura 17).

4.5. INTEGRAÇÃO

A integração entre o front-end e o back-end foi realizada com sucesso, garantindo que as informações do servidor fossem corretamente exibidas no aplicativo. O processo de login foi implementado para autenticar o usuário, e a partir disso, os dados do candidato foram carregados e exibidos nas telas apropriadas.

A autenticação foi gerenciada por meio do **JWT**, garantindo que apenas usuários autenticados acessassem informações sensíveis. A biblioteca **React Hook Form** foi utilizada para a implementação dos formulários, oferecendo validação automatizada e exibição de erros.

5. TESTES DE USABILIDADE

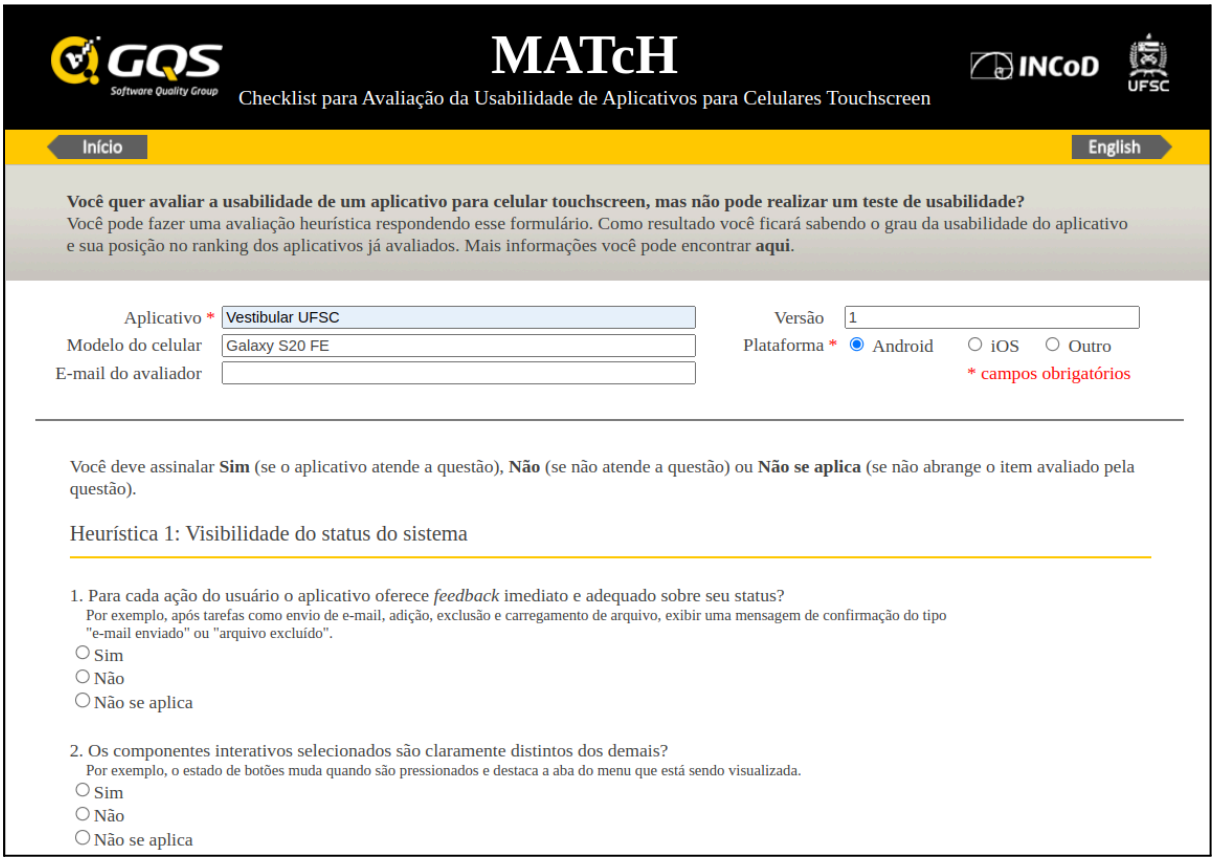
Nesta seção, serão apresentados os testes de usabilidade realizados para avaliar a experiência dos usuários com o aplicativo desenvolvido. Esses testes foram elaborados para identificar a eficiência, a eficácia e a satisfação dos usuários ao interagir com o sistema, proporcionando dados fundamentais para eventuais melhorias na interface e na navegação. Para tal, foram utilizados dois instrumentos complementares: o MATcH Checklist, desenvolvido pelo Grupo de Qualidade de Software (GQS) da UFSC, e o questionário System Usability Scale (SUS), uma ferramenta amplamente utilizada para avaliação de usabilidade em projetos de software.

O uso combinado desses métodos proporciona uma análise abrangente da usabilidade, permitindo que tanto aspectos específicos quanto uma visão geral da experiência do usuário sejam capturados e avaliados quantitativamente e qualitativamente.

5.1 MATCH CHECKLIST

O MATcH Checklist é um instrumento de avaliação de usabilidade desenvolvido pela UFSC, utilizado para identificar possíveis barreiras na interação entre o usuário e o sistema. O checklist contém critérios baseados em diretrizes ergonômicas, abordando diversos aspectos da usabilidade, como a facilidade de uso, a consistência das interfaces e a clareza das informações apresentadas.

Figura 37: Tela inicial do MATcH Checklist.



Fonte: MATcH Checklist, GQS UFSC, 2024.

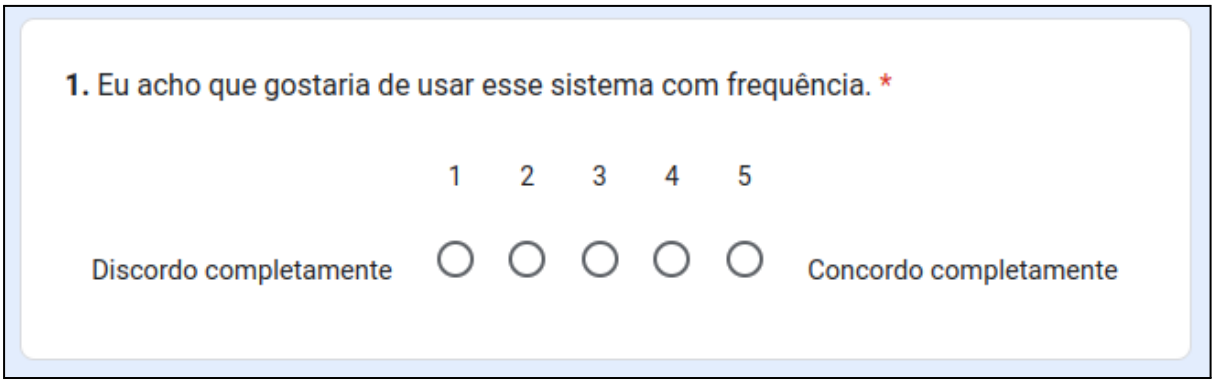
O processo de aplicação do MATcH envolve a análise de critérios específicos, em que cada item é avaliado quanto à sua presença e adequação no aplicativo. O checklist permite uma verificação sistemática, assegurando que elementos críticos de usabilidade sejam observados e registrados. A aplicação do MATcH neste projeto foi realizada para oferecer um diagnóstico detalhado das áreas que necessitam de ajustes, contribuindo diretamente para a melhoria da experiência do usuário.

5.2. QUESTIONÁRIO SUS

O SUS é um questionário de usabilidade criado para medir a percepção dos usuários em relação à facilidade de uso de um sistema. Composto por 10 perguntas, o SUS avalia a experiência do usuário de forma simples e rápida, permitindo obter um escore de usabilidade que varia de 0 a 100. Cada item é

pontuado em uma escala Likert de cinco pontos, que vai de "discordo totalmente" a "concordo totalmente".

Figura 38: Primeira pergunta do questionário SUS.



Fonte: Elaborado pelo autor.

O SUS é amplamente utilizado por sua eficiência e simplicidade, fornecendo uma métrica quantitativa da usabilidade que pode ser comparada a padrões de referência. Neste projeto, o SUS foi aplicado após os participantes utilizarem o aplicativo, para registrar a percepção imediata de usabilidade e satisfação, possibilitando a análise de possíveis melhorias.

5.2.1. CÁLCULO DA PONTUAÇÃO DO SUS

Para interpretar os resultados do questionário SUS, foi utilizada a seguinte metodologia de cálculo:

1. **Conversão das Respostas para Pontuações SUS:**
    - O SUS é composto por 10 questões alternadas entre positivas (ímpar) e negativas (par).
    - Para as perguntas ímpares (1, 3, 5, 7, 9), que representam afirmações positivas, subtraímos 1 do valor da resposta dada pelo usuário. Por exemplo, se um participante respondeu 5, a pontuação para essa pergunta será 5 - 1 = 4.
    - Para as perguntas pares (2, 4, 6, 8, 10), que são afirmações negativas, subtraímos de 5 o valor da resposta dada pelo usuário. Por exemplo,

se o participante respondeu 1, a pontuação para essa pergunta será 5 - 1 = 4.

2. **Soma das Pontuações:**
    - As pontuações das 10 questões de cada participante são somadas, resultando em um total que varia entre 0 e 40.
3. **Cálculo da Pontuação Final do SUS:**
    - Para obter a pontuação final do SUS, multiplicamos a soma de cada participante por 2,5, convertendo-a para uma escala de 0 a 100. Este resultado reflete a percepção geral de usabilidade do sistema.

O valor final do SUS, resultante da média das pontuações de todos os participantes, permite uma interpretação qualitativa da usabilidade do sistema. De acordo com a literatura, pontuações abaixo de 50 são consideradas insatisfatórias, entre 50 e 70 indicam uma usabilidade aceitável, enquanto valores superiores a 70 são classificados como "bons", com pontuações acima de 85 sendo consideradas "excelentes".

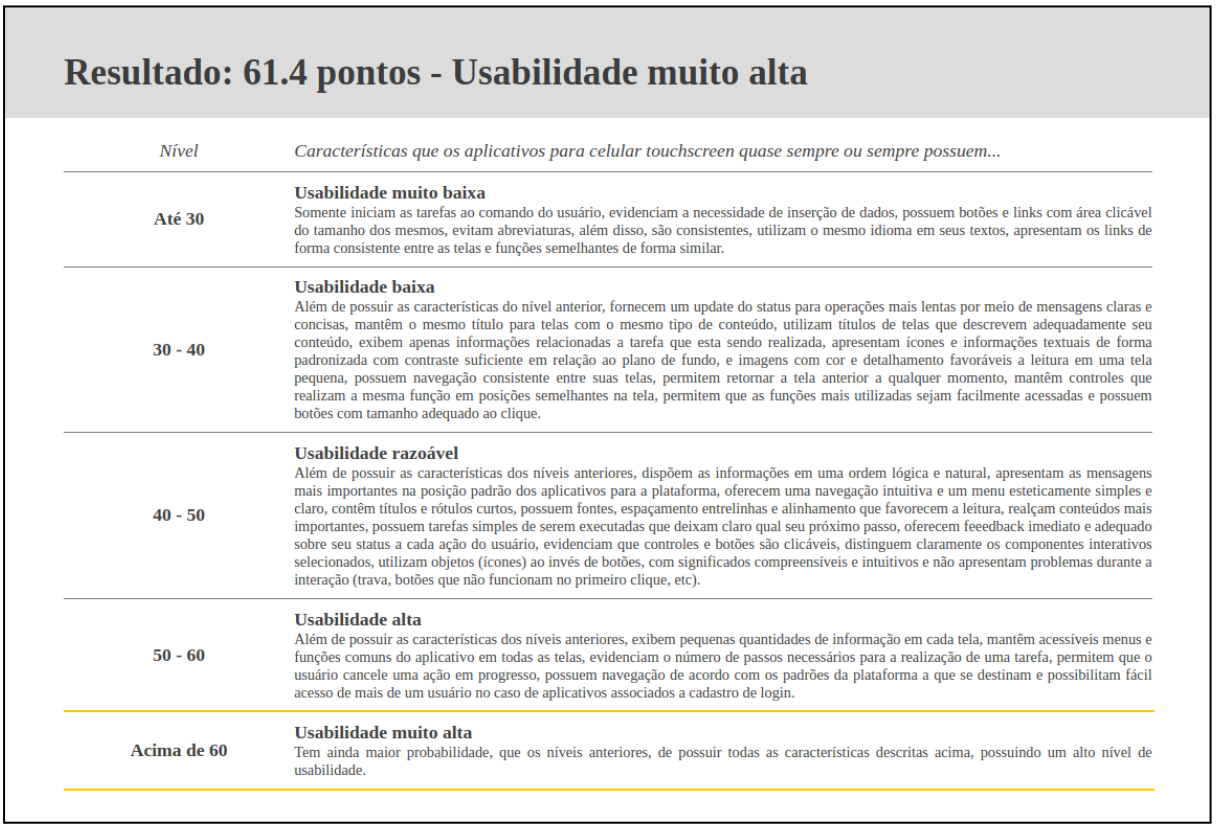
5.3. RESULTADOS

Nesta subseção, serão apresentados os resultados obtidos a partir da aplicação do MATcH Checklist e do questionário SUS. Os dados coletados refletem a percepção dos usuários em relação aos principais aspectos de usabilidade do aplicativo, destacando pontos fortes e oportunidades de aprimoramento.

5.3.1. RESULTADO MATCH

A avaliação de usabilidade do aplicativo foi realizada utilizando o MATcH Checklist. Este checklist analisa diversos aspectos relacionados à usabilidade de aplicativos para dispositivos touchscreen, atribuindo uma pontuação final que classifica o nível de usabilidade do aplicativo em diferentes categorias, de "Usabilidade muito baixa" a "Usabilidade muito alta".

Figura 39: Resultado questionário MATcH Checklist.



Resultado: 61.4 pontos - Usabilidade muito alta

Nível	*Características que os aplicativos para celular touchscreen quase sempre ou sempre possuem...*
Até 30	**Usabilidade muito baixa** Somente iniciam as tarefas ao comando do usuário, evidenciam a necessidade de inserção de dados, possuem botões e links com área clicável do tamanho dos mesmos, evitam abreviaturas, além disso, são consistentes, utilizam o mesmo idioma em seus textos, apresentam os links de forma consistente entre as telas e funções semelhantes de forma similar.
30 - 40	**Usabilidade baixa** Além de possuir as características do nível anterior, fornecem um update do status para operações mais lentas por meio de mensagens claras e concisas, mantêm o mesmo título para telas com o mesmo tipo de conteúdo, utilizam títulos de telas que descrevem adequadamente seu conteúdo, exibem apenas informações relacionadas a tarefa que esta sendo realizada, apresentam ícones e informações textuais de forma padronizada com contraste suficiente em relação ao plano de fundo, e imagens com cor e detalhamento favoráveis a leitura em uma tela pequena, possuem navegação consistente entre suas telas, permitem retornar a tela anterior a qualquer momento, mantêm controles que realizam a mesma função em posições semelhantes na tela, permitem que as funções mais utilizadas sejam facilmente acessadas e possuem botões com tamanho adequado ao clique.
40 - 50	**Usabilidade razoável** Além de possuir as características dos níveis anteriores, dispõem as informações em uma ordem lógica e natural, apresentam as mensagens mais importantes na posição padrão dos aplicativos para a plataforma, oferecem uma navegação intuitiva e um menu esteticamente simples e claro, contêm títulos e rótulos curtos, possuem fontes, espaçamento entrelinhas e alinhamento que favorecem a leitura, realçam conteúdos mais importantes, possuem tarefas simples de serem executadas que deixam claro qual seu próximo passo, oferecem feeedback imediato e adequado sobre seu status a cada ação do usuário, evidenciam que controles e botões são clicáveis, distinguem claramente os componentes interativos selecionados, utilizam objetos (ícones) ao invés de botões, com significados compreensíveis e intuitivos e não apresentam problemas durante a interação (trava, botões que não funcionam no primeiro clique, etc).
50 - 60	**Usabilidade alta** Além de possuir as características dos níveis anteriores, exibem pequenas quantidades de informação em cada tela, mantêm acessíveis menus e funções comuns do aplicativo em todas as telas, evidenciam o número de passos necessários para a realização de uma tarefa, permitem que o usuário cancele uma ação em progresso, possuem navegação de acordo com os padrões da plataforma a que se destinam e possibilitam fácil acesso de mais de um usuário no caso de aplicativos associados a cadastro de login.
Acima de 60	**Usabilidade muito alta** Tem ainda maior probabilidade, que os níveis anteriores, de possuir todas as características descritas acima, possuindo um alto nível de usabilidade.

Fonte: MATcH Checklist, GQS UFSC, 2024.

Conforme ilustrado na imagem de resultados (Figura 39), o aplicativo obteve uma pontuação de 61.4 pontos, o que corresponde ao nível de usabilidade muito alto. Esse nível de usabilidade indica que o aplicativo cumpre ou excede todas as características desejáveis dos níveis anteriores. Em termos práticos, essa pontuação sugere que o aplicativo possui um alto nível de usabilidade, com um design intuitivo, uma navegação bem estruturada e elementos interativos que facilitam o uso.

5.3.2. RESULTADO SUS

O questionário SUS foi aplicado para avaliar a usabilidade do aplicativo, sendo respondido por um total de 11 participantes. A média final do SUS obtida foi de 89,58 pontos em uma escala de 0 a 100, indicando um alto nível de usabilidade. Esta pontuação sugere uma aceitação elevada entre os usuários, com o sistema

sendo avaliado como "excelente" de acordo com os padrões de interpretação do SUS.

De acordo com a escala SUS, pontuações acima de 85 são geralmente associadas a uma excelente experiência de usuário, situando o aplicativo em um patamar de alta satisfação e usabilidade. A partir desses resultados, é possível concluir que o aplicativo cumpre com sucesso seu propósito de oferecer uma interface intuitiva e funcional. No entanto, os dados também podem ser utilizados para identificar áreas específicas para melhorias futuras, garantindo uma experiência ainda mais satisfatória aos usuários.

Figura 40: Resultado do questionário SUS calculado através do Google Sheets.

Questionário SUS

Endereç	1.	2.	3.	4.	5.	6.	7.	8.	9.	10	Média Score	Score * 2,5	Nota Final
Pessoa 1	4	1	5	1	5	1	5	1	5	1	39	97,5	**89,58**
Pessoa 2	5	1	5	1	5	1	5	1	5	1	40	100	
Pessoa 3	5	1	5	1	5	1	5	1	5	1	40	100	
Pessoa 4	4	1	5	2	5	1	4	1	5	1	37	92,5	
Pessoa 5	5	1	5	1	5	1	5	1	5	1	40	100	
Pessoa 6	5	1	5	1	5	1	5	1	5	1	40	100	
Pessoa 7	4	1	5	1	5	1	5	1	5	1	39	97,5	
Pessoa 8	5	1	5	1	5	1	5	1	5	1	40	100	
Pessoa 9	3	1	5	2	5	1	4	1	4	1	35	87,5	
Pessoa 10	5	1	5	1	5	1	5	1	5	1	40	100	
Pessoa 11	5	1	5	1	5	1	5	1	5	1	40	100	

Fonte: Google Sheets elaborado pelo autor.

A Figura 40 apresenta o resultado detalhado das respostas de cada participante, incluindo as pontuações individuais e o cálculo final da média, conforme elaborado no Google Sheets.

6. CONCLUSÃO

Este trabalho apresentou o desenvolvimento de um aplicativo móvel voltado para auxiliar candidatos inscritos no vestibular da Universidade Federal de Santa Catarina (UFSC). Com o objetivo de aprimorar a experiência desses candidatos, o aplicativo foi projetado para notificar o candidato e fornecer informações detalhadas sobre os processos seletivos, incluindo datas e locais de prova, boletins de desempenho, classificação obtida, e convocação para matrícula. Durante o processo de desenvolvimento, foram utilizadas técnicas de design centrado no usuário e ferramentas de avaliação de usabilidade, como o checklist MATcH e o questionário SUS, para assegurar que o aplicativo atendesse às necessidades dos candidatos de forma eficiente e intuitiva.

Os resultados dos testes de usabilidade indicaram um alto nível de usabilidade, com o aplicativo recebendo uma avaliação positiva dos usuários quanto à sua facilidade de uso e organização das informações. Este projeto alcançou seus objetivos ao criar uma solução digital que melhora o acesso a informações essenciais, facilitando o processo de acompanhamento do vestibular. Espera-se que o aplicativo contribua para uma experiência mais positiva para os candidatos, eliminando barreiras e promovendo o acesso à educação.

6.1. TRABALHOS FUTUROS

Para dar continuidade a este projeto e expandir suas funcionalidades, há diversas possibilidades de desenvolvimento futuro. Um dos próximos passos será realizar pequenos ajustes na interface e na estrutura do aplicativo, com o intuito de adaptá-lo para a plataforma iOS, permitindo que ele esteja disponível para um público ainda maior. Após essa adaptação, poderá ser realizada a publicação do aplicativo nas lojas de aplicativos, como Google Play e App Store, o que permitirá o download do aplicativo pelos usuários finais.

Outro aprimoramento planejado envolve a implementação de um sistema externo integrado de envio de notificações, que permitirá a comunicação em tempo real com os candidatos, informando-os sobre atualizações importantes, como novas convocações, mudanças de data, e lembretes de prova. Além disso, uma futura funcionalidade a ser explorada é a possibilidade de realizar a inscrição para o

vestibular diretamente pelo aplicativo, oferecendo uma experiência mais completa e prática para os candidatos, que poderão gerenciar todas as etapas do processo seletivo em um único local.

Também pode ser interessante fazer uma integração com a autenticação do GovBr, que também foi implementada na versão web do sistema.

Essas melhorias visam tornar o aplicativo uma ferramenta mais abrangente e acessível, contribuindo ainda mais para a facilidade e eficiência no acompanhamento do vestibular e para a melhoria da experiência dos candidatos.

REFERÊNCIAS

The 21 Benefits of Technology in Education, abr. 2023. Disponível em: <https://sgs.upm.edu.my/article/the_21_benefits_of_technology_in_education-72593>. Acesso em: 12 nov. 2023.

DESHDEEP, Nitin. Mobile App Or Website? 10 Reasons Why Apps Are Better, set. 2023. Disponível em: <https://vwo.com/blog/10-reasons-mobile-apps-are-better/>. Acesso em: 12 nov. 2023.

WURMSER, Yoram. The Majority of Americans' Mobile Time Spent Takes Place in Apps, jul. 2020. Disponível em: <https://www.insiderintelligence.com/content/the-majority-of-americans-mobile-time-spent-takes-place-in-apps>. Acesso em: 12 nov. 2023.

Pesquisa do Uso da TI - Tecnologia de Informação nas Empresas, 2023. Disponível em: <https://static.poder360.com.br/2023/05/pesti-fgvcia-2023.pdf>. Acesso em: 12 nov. 2023.

BERGÁRIA, Helil Barbosa; REIS, Pedro Lima; ROLAND, Carlos Eduardo de França. EUVESTIBULANDO: Aplicativo para Gerenciamento de Datas de Processos Seletivos. Disponível em: <http://periodicos.unifacef.com.br/reca/article/view/2291>. Acesso em: 04 abr. 2024.

VIEIRA, Matheus Antunes; OLIVEIRA, Daniel Cardoso Moraes de. ENEM GO: Um Aplicativo de Estudo Gamificado para o Exame Nacional do Ensino Médio. 2022-07-18. Disponível em: <https://app.uff.br/riuff/handle/1/26309>. Acesso em: 04 abr. 2024.

SILVA, Denys Alves; SOUSA, Caio Frias de. Construção de App com React Native. 2019-08-19. Disponível em: <https://revista.projecao.br/index.php/Projecao4/article/view/1316>. Acesso em: 04 abr. 2024.

Engenharia de Software . Disponível em: <http://www.governancamunicipal.sp.gov.br/conteudo/arquivos/Analise%20de%20requisitos.pdf >. Acesso em: 03 jun. 2024.

Git e GitHub para iniciantes – Tutorial completo. Disponível em: <https://fullcycle.com.br/git-e-github/>. Acesso em: 15 jun. 2024.

UFSC - MATCH Checklist. Disponível em: http://match.inf.ufsc.br:90/. Acesso em: 11 nov. 2024.

TEIXEIRA, Fabricio. O que é o SUS (System Usability Scale) e como usá-lo em seu site. Disponível em: <https://brasil.uxdesign.cc/o-que-%C3%A9-o-sus-system-usability-scale-e-como-us%C3%A1-lo-em-seu-site-6d63224481c8>. Acesso em: 11 nov. 2024.

Contribuidores da Wikipedia. SQLite. Wikipedia, The Free Encyclopedia. Disponível em: <https://en.wikipedia.org/w/index.php?title=SQLite&oldid=1221930289>. Acesso em: 14 nov. 2024.

APÊNDICE A - ARTIGO

Desenvolvimento de um Aplicativo Móvel para Auxílio aos Candidatos dos Vestibulares da UFSC

Bruno Barreto

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

bbarreto18@gmail.com

Resumo: *Ingressar em uma instituição de ensino superior é um marco significativo na vida acadêmica e exige organização e acesso facilitado a informações sobre os processos seletivos. Este estudo apresenta o desenvolvimento de um aplicativo móvel voltado para otimizar a experiência dos candidatos nos vestibulares da Universidade Federal de Santa Catarina (UFSC). A proposta inclui funcionalidades como acesso às datas, locais de prova, desempenho individual e envio de notificações personalizadas em tempo real. Além disso, o aplicativo foi projetado para superar limitações comuns dos sistemas web, garantindo acesso rápido e eficiente, mesmo em situações de conectividade reduzida. O processo de validação foi realizado por meio do questionário SUS (System Usability Scale), que resultou em um índice de usabilidade de 89,58 pontos, uma classificação excelente. Isso demonstrou que a interface do aplicativo atende de forma eficaz às necessidades dos usuários, oferecendo uma experiência intuitiva e satisfatória.*

Abstract: *Applying to a higher education institution is a significant milestone in academic life, requiring organization and easy access to information regarding the selection processes. This study presents the development of a mobile application aimed at optimizing the experience of candidates for the entrance exams of the Federal University of Santa Catarina (UFSC). The proposed solution includes features such as access to exam dates and locations, individual performance tracking, and real-time personalized notifications. Moreover, the application was designed to overcome common limitations of web systems, ensuring fast and efficient access even in scenarios of reduced connectivity. The validation process*

was conducted using the System Usability Scale (SUS) questionnaire, which yielded a usability score of 89.58, considered excellent. This demonstrated that the application's interface effectively meets users' needs, offering an intuitive and satisfying experience.

1. INTRODUÇÃO

O vestibular da UFSC é um processo essencial para o ingresso na universidade, exigindo organização e atenção por parte dos candidatos. Atualmente, as informações estão concentradas em um sistema web. No entanto, considerando o aumento do uso de dispositivos móveis no Brasil, a criação de um aplicativo móvel se torna uma oportunidade para otimizar a experiência desses usuários.

Com cerca de 464 milhões de dispositivos móveis em uso no Brasil, seguidos por estudos que indicam uma preferência crescente por aplicações móveis em comparação aos sites, os candidatos frequentemente optam por meios mais rápidos e acessíveis para gerenciar suas necessidades. Esse cenário reforça a relevância de desenvolver soluções que integrem tecnologia e mobilidade.

Este trabalho propõe um aplicativo móvel para notificar candidatos e fornecer informações sobre os vestibulares, melhorando o acesso e a organização. A interface intuitiva e as notificações oportunas são diferenciais que visam atender às necessidades dos candidatos. Além disso, o aplicativo foi estruturado para funcionar em sincronia com sistemas existentes, garantindo consistência e confiabilidade nas informações.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. DESENVOLVIMENTO

O uso de frameworks como React Native permite a criação de aplicativos para Android e iOS com uma base de código única, aumentando a eficiência no desenvolvimento. A tecnologia Expo foi empregada para simplificar o processo de desenvolvimento e compilação, reduzindo o tempo entre as etapas de codificação e teste. Essa abordagem é particularmente vantajosa para projetos de curto prazo ou com recursos limitados.

React Native também oferece flexibilidade na customização de componentes, permitindo a criação de interfaces otimizadas para diferentes tamanhos de tela e resoluções. A integração de bibliotecas como NativeWind acelera a estilização visual e aumenta a consistência entre as telas do aplicativo.

2.2. USABILIDADE E EXPERIÊNCIA DO USUÁRIO

A experiência do usuário (UX) e a usabilidade são componentes críticos no sucesso de aplicativos móveis. O questionário SUS foi utilizado para medir a usabilidade do aplicativo, sendo amplamente reconhecido como um método eficiente e confiável de avaliação.

Ademais, foi utilizado o checklist MATcH para complementar a avaliação, identificando barreiras específicas na interação do usuário com o sistema. Juntas, essas ferramentas forneceram uma análise abrangente da usabilidade, abordando tanto aspectos quantitativos quanto qualitativos.

3. METODOLOGIA

3.1. ANÁLISE DE REQUISITOS

Os requisitos funcionais incluem a visualização de informações sobre vestibulares, notificações e acompanhamento de desempenho. Esses requisitos foram definidos considerando cenários reais enfrentados pelos candidatos, como dificuldade em acessar boletins de desempenho ou obter informações de última hora.

Requisitos não funcionais destacam a segurança dos dados e alta disponibilidade. Por exemplo, a utilização de SQLite e Prisma ORM foi essencial para assegurar a persistência de dados.

3.2. IMPLEMENTAÇÃO

O aplicativo foi estruturado em front-end, desenvolvido em React Native, e back-end, utilizando Node.js com Fastify e SQLite. Notificações push foram integradas através do Expo Server SDK. O back-end foi projetado para suportar um alto número de requisições simultâneas, utilizando técnicas de cache e otimização de chamadas à API central.

3.3. VALIDAÇÃO

Foram realizados testes de usabilidade com 11 participantes, utilizando o questionário SUS e o checklist MATcH. Além da coleta de feedbacks qualitativos, os participantes foram incentivados a realizar simulações de tarefas reais, como verificar a localização de uma prova ou acessar notificações.

4. RESULTADOS

4.1. AVALIAÇÃO DE USABILIDADE

Os resultados do SUS indicaram um índice de usabilidade de 89,58, classificado como excelente. Os participantes destacaram a clareza das informações e a facilidade de navegação. Os principais pontos positivos incluíram a responsividade do aplicativo e a organização das telas, enquanto sugestões de melhoria envolveram ajustes na exibição de notificações em dispositivos com telas menores.

4.2. IMPACTO NA EXPERIÊNCIA DO USUÁRIO

O aplicativo melhorou o acesso às informações e reduziu o tempo para obter atualizações relevantes, como datas de prova e boletins de desempenho. Além disso, os participantes relataram maior confiança no uso do aplicativo em relação ao sistema web, citando a portabilidade como uma vantagem significativa.

Os resultados também indicaram que o aplicativo incentivou os usuários a acompanhar suas inscrições de forma mais frequente, eliminando barreiras técnicas ou logísticas que poderiam impedir esse processo.

5. CONCLUSÃO

O desenvolvimento do aplicativo móvel para os vestibulares da UFSC atendeu aos objetivos de melhorar a experiência dos candidatos, fornecendo uma ferramenta intuitiva e eficiente. Os resultados de usabilidade validaram a abordagem adotada, e o impacto positivo no acesso às informações foi significativo.

O sucesso deste projeto demonstra o potencial de soluções móveis em cenários educacionais e administrativos, abrindo caminho para aplicações semelhantes em outros contextos.

5.1. TRABALHOS FUTUROS

Planeja-se expandir o suporte para iOS e implementar a funcionalidade de inscrição direta no aplicativo, além de aprimorar o sistema de notificações em tempo real. Também é importante efetuar a submissão do aplicativo nas lojas de aplicativos como App Store e Play Store. Outro aprimoramento seria a integração do aplicativo com o GovBr, proporcionando uma nova forma de autenticação.

APÊNDICE B - CÓDIGO FONTE

O respectivo código fonte do projeto pode ser encontrado no seguinte link: https://github.com/BrunoBross/vestibular-ufsc