INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

SISTEMA DE *CHECK-IN* PARA ESTUDANTES EM ATIVIDADES DE FORMAÇÃO

NATHAN PEREIRA SARMENTO

Cajazeiras - Paraíba, março de 2023

NATHAN PEREIRA SARMENTO

SISTEMA DE *CHECK-IN* PARA ESTUDANTES EM ATIVIDADES DE FORMAÇÃO

Trabalho de conclusão de curso apresentado junto ao **Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas** do **Instituto Federal de Educação, Ciência e Tecnologia da Paraíba**, como requisito parcial à obtenção do título de **Tecnólogo em Análise e Desenvolvimento de Sistemas**.

Orientador:

Prof. MSc. Diogo Dantas Moreira.

Coorientador:

Prof. MSc. Fábio Abrantes Diniz.

Cajazeiras - Paraíba, março de 2023

IFPB / Campus Cajazeiras
Coordenação de Biblioteca
Biblioteca Prof. Ribamar da Silva
Catalogação na fonte: Cícero Luciano Félix CRB-15/750

S246s Sarmento, Nathan Pereira.
Sistema de *check-in* para estudantes em atividades de formação / Nathan Pereira Sarmento.– 2024.

82f. : il.

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2024.

Orientador(a): Prof. Me. Diogo Dantas Moreira.
Coorientador(a): Prof. Me. Fábio Abrantes Diniz.

1. Desenvolvimento de sistemas. 2. Gestão hospitalar. 3. Controle de frequência. 4. Estágio supervisionado. I. Instituto Federal de Educação, Ciência e Tecnologia da Paraíba. II. Título.

IFPB/CZ CDU: 004.4(043.2)

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

NATHAN PEREIRA SARMENTO

SISTEMA DE CHECK-IN PARA ESTUDANTES EM ATIVIDADES DE FORMAÇÃO

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Me. Diogo Dantas Moreira

Aprovada em: **30 de Julho de 2024.**

Prof. Me. Diogo Dantas Moreira - Orientador

Prof. Dr. Fabio Gomes de Andrade - Avaliador

IFPB - Campus Cajazeiras

Prof. Me. Francisco Paulo de Freitas Neto - Avaliador

IFPB - Campus Cajazeiras

Documento assinado eletronicamente por:

- **Francisco Paulo de Freitas Neto**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/08/2024 07:46:20.
- **Fabio Gomes de Andrade**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/08/2024 14:08:14.
- **Diogo Dantas Moreira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 12/08/2024 14:44:58.

Este documento foi emitido pelo SUAP em 01/08/2024. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 586026
Código de Autenticação: a3c573601d

RESUMO

O presente trabalho apresenta uma proposta e implementação de uma solução de software para um problema real enfrentado pela comunidade em geral do Hospital Universitário Júlio Bandeira (HUJB), o de manejo das atividades de estágio no hospital. Para tanto, foi desenvolvido, em parceria ativa com os servidores inseridos no contexto, um sistema modularizado com componentes desktop e mobile, onde cada módulo é direcionado a um conjunto definido de stakeholders, priorizando aspectos de escalabilidade e reuso para outros hospitais universitários.

Palavras-chave: Sistema de Gestão para Hospitais, Sistema de Check-in, Informatização de Sistema

ABSTRACT

This present work deals with a system proposal to solve a real problem faced by the general community of HUJB, the management of internship activities at the hospital. For the solution, we thought of creating a modular system where each module presents a set of responsibilities within a specific scope, we also emphasized the creation of software components, components that would take care of a single responsibility at a time, favoring code cohesion , finally, with a well-designed software, we solved the stakeholder's main request, which is a better facility in the management of interns, and we fulfilled the 2nd main point of the software, which is scalability and reuse in other university hospitals.

Keywords: Management System for Hospitals, Check-in System, System Informatization

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A administração de empresas e repartições públicas tem sido cada vez mais difícil. Com o passar dos anos, as estruturas organizacionais foram ficando cada vez mais complexas de administrar por vários fatores, tais como aumento do número de funcionários, número maior de responsabilidades empregadas, legislação, dentre outros fatores. Tudo isso culminou em essas organizações serem obrigadas a ter uma melhor visão estratégica para conseguir cumprir seu propósito com cada vez mais eficiência.

Uma organização visando a melhoria geral no desempenho de suas atividades e melhoria na qualidade do produto deve ser capaz de confeccionar visões estratégicas, só que para elaborar estas ela deve também conhecer os seus processos de ponta a ponta. É somente conhecendo os processos dentro da empresa tanto a nível micro (olhando o processo individualmente) quanto macro (olhando todos os processos como um todo) que pode-se traçar as melhores estratégias.A TI se tornou um meio bastante conhecido para resolver esse problema, pois permite a organização, visualização e controle de informações, auxiliando na tomada de decisões administrativas.

Em um Hospital Universitário não poderia ser diferente assim como uma repartição pública cujo serviços são tanto a nível educacional realizando o treinamento prático de profissionais da saúde como também à comunidade no que tange às responsabilidades de um hospital comum.

O Hospital Júlio Bandeira fundado em 1970 no município de Cajazeiras, teve como objetivo inicial conter os altos índices de mortalidade infantil da época (Empresa Brasileira de Serviços Hospitalares, 2020). Com o tempo acabou se tornando um Hospital Universitário vinculado à Universidade Federal de Campina Grande, abrigando os períodos de estágios dos alunos matriculados no curso de Medicina da Universidade no campus de Cajazeiras de acordo com a resolução Nº 06/2015 (Colegiado Pleno do Conselho Universitário, 2015), tendo assim um compromisso na formação dos estudantes de Saúde pois trata-se do único hospital universitário do sertão Paraibano.

1.2 PROBLEMÁTICA

Como uma organização publica com objetos de servir a comunidade tanto de forma acadêmica quanto prestando serviços como um hospital comum e para atingir eficiência é necessário tomadas de decisões estratégicas usando recursos tecnológicos de TI.

Atualmente o HUJB já conta com estratégias de TI aliadas aos seus processos como um hospital.

Já a sua outra finalidade, a de auxílio na formação de novos profissionais, os processos são feitos de forma manual, demandando assim mais tempo e recursos humanos. A problemática deste trabalho se apresenta dentro desse contexto, explorando os problemas enfrentados pelos diversos setores, profissionais e estudantes envolvidos nos processos citados anteriormente.

1.3 OBJETIVOS

O **objetivo geral** deste trabalho é apresentar uma solução de **software para registro de frequência** dos estudantes que realizam atividades de formação no Hospital Universitário Júlio Bandeira. Os **objetivos específicos** desse trabalho são:

- Realizar entrevistas com os stakeholders, a fim de identificar problemas de processos e pontos de melhoria;
- Projetar uma proposta de solução de software para os problemas apresentados;
- Validar a proposta de software com os stakeholders;
- Definir e implementar um escopo da solução validada;

1.4 ORGANIZAÇÃO DO TRABALHO

O presente trabalho conta com a seguinte organização: No capitulo 1 é contextualizado o problema do trabalho. No capitulo 2 é apresentado a metodologia utilizada para o trabalho.No capitulo 3 elaboração da proposta do sistema com arquitetura e stakeholders.No capitulo 4 são apresentadas todas a funções que o sistema exerce. No capitulo 5 é falado sobre o objetivo final do trabalho e contribuições futuras.

2 METODOLOGIA

O presente trabalho apresenta-se como uma pesquisa aplicada, visando a apuração dos conhecimentos necessários dentro de diferentes áreas para a aplicação prática dentro do problema sugerido, e tal pesquisa classifica-se como exploratória, visto que possui como principal finalidade proporcionar uma maior aproximação com a temática levantada.

A principal fonte de coleta de dados sobre a problemática e os requisitos almejados foram os próprios stakeholders, principalmente os funcionários da administração do Hospital Júlio Bandeira, em Cajazeiras - PB. A coleta foi conduzida por meio de entrevistas informais para identificar os problemas e levantar possíveis melhorias.

Quanto à metodologia de desenvolvimento da aplicação, foram utilizadas práticas da metodologia *Scrum*, tais como reuniões de planejamento, revisão e grooming. Os detalhes do processo segundo essa metodologia serão detalhados na seção de processo de desenvolvimento de software.

2.1 FLUXO DO TRABALHO

Nesta seção serão detalhadas as atividades a serem desenvolvidas ao longo desse trabalho. O Quadro 2.1 demonstra as atividades alocadas dentro de seus meses de execução.

2.1.1 A0 - Acolhimento da Proposta

Nessa etapa foi definida a proposta de desenvolvimento de software proferida pelo HUJB para compor como tema deste presente trabalho.

2.1.2 A1 - Entrevista Técnica I

Nessa etapa foi realizada a primeira entrevista técnica com o cliente, realizada no próprio hospital com um dos stakeholders nessa entrevista foi falado sobre as dificuldades em manter a gestão dos estágios em folhas de pontos e também foi falado sobre fluxos que o software deve ter para resolver o problema.

2.1.3 A2 - Análise de Aplicativo Check-In SERPRO

Nessa etapa foi feita uma análise das funcionalidades do aplicativo *check-in* SERPRO, esse aplicativo indicado por um dos stakeholders,como a ferramenta mais próxima da proposta.

A2.1 Entrevista Técnica II. Nessa etapa com base na entrevista técnica, e a análise do Aplicativo foram elencados os primeiros Requisitos Funcionais de Software sendo eles as RFS 1,2,3,8,9,10 além da visualização de pontos inda esclarecidos.

A2.2 Entrevista 2 com os Stakeholders. Nessa etapa foi feito uma entrevista para um fechamento do escopo inicial da proposta nessa entrevista foi entendido melhor o funcionamento dos setores,preceptores a adição de permissões de acesso ao sistema, respectivamente as RFS 8,4,5

2.1.4 A3 - Análise e Projeto

A3.1 Arquitetura do Sistema. Etapa com a proposta arquitetural do sistema, nessa etapa foi pensado na criação de 2 módulos,realizando a separação de características dos usuários tornando-a modelagem mais simples,além da criação da RF 7.

A3.2 Processo de Desenvolvimento de Software. Etapa que define o processo de desenvolvimento de software utilizado sendo o escolhido o Scrum.

A3.3 Prototipação. Etapa que definiu a criação de protótipos utilizáveis, para mais uma vez validar os requisitos propostos em formato de telas navegáveis.

A3.4 Implementação. Etapa que definiu o desenvolvimento da aplicação

A3.5 Validação e Integração Contínua. Etapa de Análise para avaliar se o software está sendo desenvolvido da maneira que os stakeholders desejam.

A3.6 Documentação. Etapa que compreende a criação da documentação da proposta de software está será descrita neste Trabalho de Conclusão de Curso.

Quadro 2.1 – Cronograma de execução das atividades.

	TCC1						**TCC2**						
	2022				2023								
Atividades	**Setembro**	**Outubro**	**Novembro**	**Dezembro**	**Janeiro**	**Fevereiro**	**Março**	**Abril**	**Maio**	**Junho**	**Julho**	**Agosto**	**Setembro**
A0	X	X	X										
A1.1				X									
A2.1				X									
A2.2				X									
A3.1				X	X	X							
A3.2						X							
A3.3						X	X	X	X				
A3.4								X	X	X	X	X	X
A3.5						X	X	X	X	X	X	X	X
A4			X	X	X	X	X	X	X	X	X	X	X

3 O SISTEMA PROPOSTO

O aplicativo proposto por esse trabalho, é uma solução para controle de presença em setores e emissão de relatórios acerca dos horários de atividades dos estagiários dos cursos da área de saúde. Aqui, especificamente, foi usado os alunos da Universidade Federal de Campina Grande - Campus Cajazeiras como principais stakeholders, dentro do contexto do Hospital Júlio Bandeira, sediado na cidade de Cajazeiras - PB, como citado anteriormente neste trabalho.

Este capítulo vai detalhar os stakeholders e requisitos para a aplicação proposta, como obtido por meio de reuniões remotas. Também serão apresentadas as tecnologias utilizadas e como elas integram a arquitetura de software proposta para o desenvolvimento da ferramenta.

3.1 STAKEHOLDERS

Baseado nas reuniões com os stakeholders dessa proposta, foram identificados diferentes tipos de usuários: estagiários, preceptores e a administradores do hospital. Cada um deles será detalhado abaixo:

- **Estagiário** O estagiário é um dos principais interessados na solução aqui proposta, uma vez que ele atua em atividades diversas dentro do hospital ao qual foi designado a fim de completar a carga horária necessária para as disciplinas que está cursando na instituição de origem. Esse stakeholder busca facilidade na hora de registrar suas atividades e uma maneira prática de acompanhar o andamento das horas completadas.
- **Preceptor** O preceptor é um funcionário de plantão no hospital o mesmo além de desempenhar suas funções como funcionário ele também é responsável pelo monitoramento dos alunos em seus estágios como orientadores.
- **Administrador do Hospital** A administração do hospital é composta pelos funcionários responsáveis pela administração no contexto da proposta, sua principal função é a de monitoramento do processo de estágio geral de todos os alunos.

3.2 REQUISITOS

Essa seção irá detalhar os requisitos levantados para a aplicação proposta, assim como exibir os protótipos quando forem pertinentes. Os quadros exibidos nesta seção

apresentam o conceito de Prioridade. A prioridade irá variar de 1 a 5, sendo 5 os mais importantes ou seja foram os primeiros a ser implementados no software e 1 o que foram implementados em possíveis entregas futuras seja as mesmas dentro ou fora do escopo do projeto.

O ponto principal da proposta é resolver o problema da gestão de estágios, e todos os requisitos são permeados por essa problemática. Para tentar resolver o problema, foi implementado uma catalogação dos dados acerca do estágio de um estudante no hospital.

No Quadro 3.1, são expostos alguns requisitos diretamente ligados a esse contexto.

Só que surge um problema: quem deve validar as horas de estágios dos alunos? Para isso cada registro criado na **RF6** apresenta um status que é mudado quando o preceptor aprovar o registro de tempo do estágio do estagiário. Além do status também será adicionado um campo de assinatura para assim sabermos quem foi o responsável pela validação.

Os requisitos elencados no quadro 3.1 passaram por um refinamento, registrado em formato de *user stories* para adição de mais detalhes.

3.3 ARQUITETURA E TECNOLOGIAS

Definidos os requisitos de alto nível, foi projetada a arquitetura do software para a proposta a ser implementada nesse trabalho. A arquitetura de software permite aos stakeholders técnicos uma visualização de alto nível do projeto isso não só deixa o software mais claro quanto ao seu funcionamento mas também pode mostrar pontos de inconsistência que não foram esclarecidos em conversa com as partes interessadas.

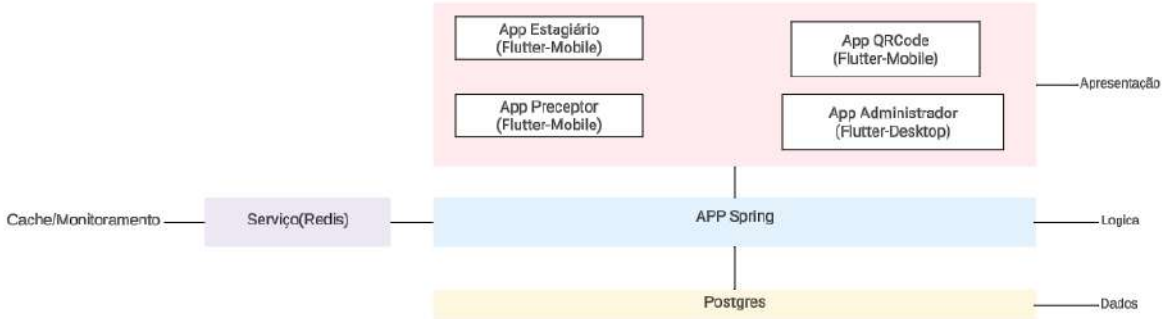
O software com uma boa arquitetura promove importantes indicadores de qualidade e da coesão de classes e/ou módulos um software coeso em suas classes e módulos evita problemas comuns da engenharia de software entre eles: repetição de códigos,aumento exponencial no uso de recursos para adição de novos requisitos, ao mesmo tempo que provê reuso, possibilidade maior de escala e facilidade de manutenção.

A Figura 3.1 demonstra de maneira visual a arquitetura para o sistema proposto aqui. Os detalhes das tecnologias juntamente com suas respectivas camadas serão descritos na sequência.

Quadro 3.1 – Requisitos levantados da aplicação.

RF	ID	Descrição	Prioridade
1	**Gestão de Usuários (Estagiário)**	Cadastramento, Remoção, Atualização e Leitura dos dados cadastrais dos Estagiários no HUBJ	5
2	**Gestão de Usuários (Acompanhamento de Estagiários)**	Acompanhamento de quantos estagiários estão no hospital e os seus respectivo setor	5
3	**Gestão de Usuários (Administradores)**	Cadastramento, Remoção, Atualização e Leitura dos dados cadastrais dos Administradores do HUJB	5
4	**Gestão de Usuários (Preceptores)**	Cadastramento, Remoção, Atualização e Leitura dos dados cadastrais dos Preceptores do HUJB	5
5	**Gestão de Usuários (Permissões)**	Níveis de acessos que os usuários terão ao sistema	5
6	**Gestão de Registro (Registro)**	Cadastramento, Atualização e Leitura dos dados dos Registros feitos pelos estagiários do HUJB	5
7	**Gestão de Registo (Assinatura)**	Assinatura dada pelo profissional preceptor para validação do Registro feito estagiário	5
8	**Gestão de Registro (Setores)**	Cadastramento, Atualização e Leitura dos dados dos Setores que recebem estagiários do HUJB	5
9	**Geração de QR Code**	QR code gerado para o aluno conseguir criar um registro	5
10	**Lista de Presença do Plantão**	Uma lista de Presença que mostre todos os estagiários que participaram em um plantão de um preceptor	3
11	**Gestão de Registo (Invalidação)**	Invalidação do registro do estagiário, feita pelo profissional preceptor no processo de avaliação do registro	5

Figura 3.1 – Representação gráfica da arquitetura de componentes



Fonte: Elaborado pelo autor (2024)

Na seguinte figura, é apresentado os sub-sistemas que juntos compõem o sistema geral do presente trabalho.

App Estagiário: Aplicativo Mobile para celulares (Android/Iphone) desenvolvido em flutter, é por meio deste aplicativo que os estagiários terão acesso a suas respectivas funcionalidades, são elas: Realização de todo o processo de estagiário (Check-In e Check-out), e visualização das informações de estágios feitos.

App Preceptor: Aplicativo Mobile para celulares (Android/Iphone) desenvolvido em flutter, por meio desse aplicativo os preceptores tem acesso aos estágios feitos que ainda não foram validados ou rejeitados, e por meio do mesmo eles realizam o processo de assinatura ou rejeição.

App QRCode: Aplicativo Mobile para tablets (Android) desenvolvido em flutter, este aplicativo é responsável por gerar os QRCodes nos tablets do hospital,QRCodes estes que contém a informação do setor do qual o tablet está localizado para o sistema.

App Spring: Aplicativo da lógica de negócio do sistema, o aplicativo foi desenvolvido no framework Spring e sua função é ser responsável por toda lógica de funcionamento do sistema, como: Autenticações,Lógica de CheckIn e CheckOuts, processos de assinatura e rejeição entre outros.

Serviço(Redis): O banco NoSQL Redis usado no sistema para adicionar estado ao sistema, assim os processos check-in que não foram concluídos são armazenados em uma instancia do redis e após conclusão total do processo de estágios são retirados e armazenados no banco de dados do sistema.

Postgres: Banco de Dados de código aberto usado no sistema para a persistência de todos os dados.

3.3.1 Camada de apresentação

Foi escolhido para a camada de apresentação, Flutter. O flutter permite a construção de aplicações web,desktop e mobile, não importando o tipo de Sistema Operacional do qual o dispositivo pertence o flutter contém porte para Android e IOS para mobile e Windows,Linux e Mac para desktop. A complexidade do projeto requere a construção de aplicações tipo mobile e desktop na camada de apresentação por isso o flutter foi escolhido mantendo assim uma base de código único.

Flutter: Flutter é um kit de desenvolvimento de interfaces feito em C C++ Dart desenvolvido pelo Google, lançado para o mercado em 4 de outubro de 2018, seu

funcionamento se dá na criação de widgets ou seja componentes que juntos irão compor as telas do aplicativo.Sendo um dos framewoks mais ultilizado no desenvolvimento de aplicativos cross-plataform, ele conta não só com o suporte de um grande como Google mas também de uma grande comunidade de desenvolvedores que se ajudam e criam continuamente novas bibliotecas para auxiliar no desenvolvimento de apps em Flutter. (FLUTTER, 2024).

3.3.2 Camada Lógica

A escolha pelo uso do Spring Boot se deu pelo fato dos desenvolvedores poderem contar com um melhor adaptabilidade a usar essa tecnologia, o fato de possuir boa documentação e suporte da comunidade, integrações e outras coisas pré-prontas auxiliando o desenvolvimento e por apresentar um ambiente favorável à escalabilidade do aplicativo.

Spring: O spring é um framework para o desenvolvimento de aplicativos corporativos em Java, seu surgimento se deu em junho de 2003 o motivo foi para substituir o JEE2 que contava com um problema de mal gerenciamento de Beans em Java, hoje em dia ele é um dos frameworks mais usados para desenvolvimento de aplicativos em Java, devido ao seu grande tempo já em mercado marcando-o como um framework robusto, além de possuir muitas coisas já pré-prontas como integrações com serviços de Autenticação, Banco de Dados, Cloud (TANZU, 2023).

3.3.3 Autenticação/Autorização

A segurança da informação é cada vez mais necessária na construção das aplicações atuais, com ela garantimos a Confidencialidade, Integridade, Disponibilidade, Autenticidade e Não-repúdio nos sistemas. Uma das formas de prover à aplicação uma maior segurança é a implementação de métodos de Autenticação e Autorização para garantir o acesso ao sistema, apenas pessoas autorizadas e conhecidas pelo mesmo. No presente trabalho o uso do token JWT, especificado na RFC 7519 permite a implantação de uma forma leve de autenticação entre 2 ou mais partes sejam quais forem elas, baseado em objetos JSON emitidos pelo servidor do qual será acessado o/os rercursos. A estrutura do JWT consiste em 3 partes: Header,Payload,Signature

Header: O Header se divide em 2 propriedades: typ,alg.

Typ: typ se refere ao tipo do token, no caso do JWT esse typ recebe a string 'JWT'.

Alg: alg se refere ao tipo de algoritmo de hash foi usado para encryptar o token.

Payload: A payload é a parte do token do qual carrega alguma informação de identificação (id no banco de dados) do usuário, usada pelo servidor para acesso aos recursos.

Signature: O signature é a parte do token necessária para garantir a integridade do mesmo, ela é formada pela seguinte estrutura: base64UrlEncode(header) + "."+ base64UrlEncode(payload) + secret o secret é uma sequencia de caracteres usado para garantir que apenas o servidor que emitiu o token, seja capaz de decripta-lo.

O resultado final do token é uma string formada pelas 3 partes encriptadas separadas por ".".

O token na web é trafegado através da propriedade Authorization contida no header de uma requisição HTTP/HTTPS.(AUTH0, 2024)

3.3.4 Camada de Dados

Não há muita diferença no que diz respeito aos SGBDS em banco de dados relacionais, então apenas por uma questão dos desenvolvedores estarem acostumados a utilizar essa ferramenta ele será utilizado.

Postgres: Postgres é um Sistema de Gerenciamento de Banco de Dados open source feito para gerência de banco de dados Relacionais, ele conta com uma boa comunidade, já esta a muito tempo em mercado sendo lançado em 8 de julho de 1996, além de uma boa ferramenta gráfica o PGADMIN do qual podemos se interfacear com as conexões e os bancos de dados (PostgreSQL Global Development Group, 2023).

Foi pensado a divisão do sistema em 2 módulos: um de gestão de usuários e outro de gestão de registros.

3.3.5 Gestão de usuários

O aplicativo conta com diferentes tipos de usuários: os estagiários, os preceptores e os administradores do hospital. Baseado nas suas funções dentro das atividades do hospital, cada usuário tem no sistema um papel e um nível de permissão.

Tendo essas informações em mente foi projetada a criação de um tipo genérico para abranger cada um dos papéis citados, ou seja um tipo genérico para usuários, papéis e permissões e criando um tipo complexo a partir da união dos 3 tipos. Assim, um estagiário que é uma representação complexa seria subdividido em representações simples, sua representação como pessoa na tabela de usuários, sua representação como papel na tabela de papel e seus níveis de acesso baseado no seu papel dentro do sistema.

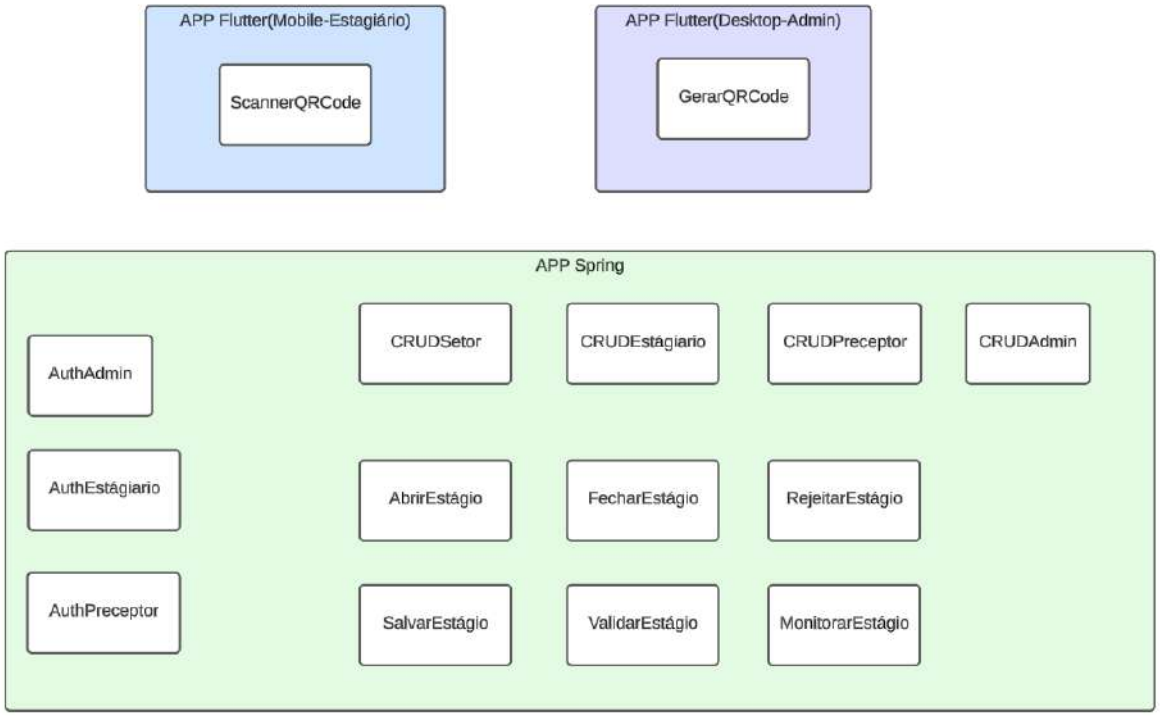
3.3.6 Gestão de registros

A gestão de registros é responsável por prover funcionalidade para que os estagiários possam registrar suas atividades dentro do sistema, os preceptores possam validar as atividades registradas e que ambos possam acompanhar as atividades desempenhadas em cada um dos setores do hospital.

Além disso, esse módulo é interessante para a administração do sistema, visto que seus dados podem prover informações suficientes para relatórios e outros trâmites administrativos.

3.3.7 Representação gráfica da arquitetura de serviços

Figura 3.2 – Arquitetura de Serviços



Fonte: Elaborado pelo autor (2024)

Figura que representa todas as funcionalidades providas pelo sistema.

3.4 PROCESSO DE DESENVOLVIMENTO

O scrum é um framework para desenvolver e entregar produtos complexos baseado no empirismo, a ferramenta promove uma melhoria contínua dos processos se baseando em resultados dos anteriores. O scrum se baseia em 3 pilares fundamentais:

Princípio da Transparência. O princípio da Transparência prega que todos os stakeholders envolvidos nos processos do scrum devem ser devidamente informados em uma linguagem comum entre eles acerca dos artefatos produzidos em cada um dos processos

Princípio da Inspeção. O princípio da Inspeção determina o cuprimento das regras do scrum, ou seja garantindo assim a qualidade dos artefatos produzidos ao fim de cada Sprint

Princípio da Adaptabilidade. O princípio da Adaptabilidade fala sobre a flexibilidade que o processo de desenvolvimento de software deve ter, afinal o scrum é pautado numa melhoria contínua tanto dos processos quanto dos artefatos desenvolvidos, por isso o mesmo, ele deve ser adaptável ao contexto do estado atual do desenvolvimento do produto.

Sobre as nomenclaturas do Scrum temos: **Backlog**, que se refere às tarefas que são necessárias para o desenvolvimento do produto, ou seja, podem ser, requisitos de softwares, refatorações, esclarecimentos de requisitos, entre outros. **Sprint**, que refere-se a um período de tempo pré-determinado que possui cerca de 1 mês de duração tendo como objetivo, o cumprimento de um conjunto de tarefas descritas no backlog. **Backlog-Sprint**, que é um conjunto de tarefas que são eleitas para serem cupridas no periodo de tempo da sprint em questão (SCHWABER; SUTHERLAND, 2013).

Esse é uma ideia geral do Scrum mas devido esta proposta de software possuir diferenças a projetos tradicionais de software, uma escolha mais assertiva é o Scrum Solo.

O scrum solo segue os mesmos princípios e funcionamento do scrum tradicional com o adendo que ele é projetado para projetos de um único desenvolvedor, ele mescla os princípios do scrum com os princípios do PSP (Personal Software Process) um conjunto de práticas para desenvolvedores individuais de software (PAGOTTO et al., 2016).

4 FUNCIONALIDADES

As funcionalidades de um software definem a capacidade de através de algoritmos,procedimentos entre outros recursos computacionais, resolver problemas reais de quaisquer natureza, sejam eles complexos,repetitivos,críticos entre outros aspectos. O presente capitulo conta com a implementações das regras de negócio do sistema (os RFs) e conta também com soluções técnicas de apoio (Requisitos Não Funcionais) para facilitar e diminuir a complexidade dos requisitos funcionais.

4.1 GESTÃO DE USUÁRIOS

A implementação da gestão de Usuários foi feita através da criação de uma aplicação para desktop feita em flutter. A aplicação conta com de modo geral,o acesso ao banco de dados do sistema, e ao acompanhamento em tempo real sobre os processos de check-in abertos no sistema. Com isso o aplicativo abrange a implementação técnica das RFs 1,2,3,4,5.

4.2 REQUISITOS NÃO FUNCIONAIS DE CONTROLE

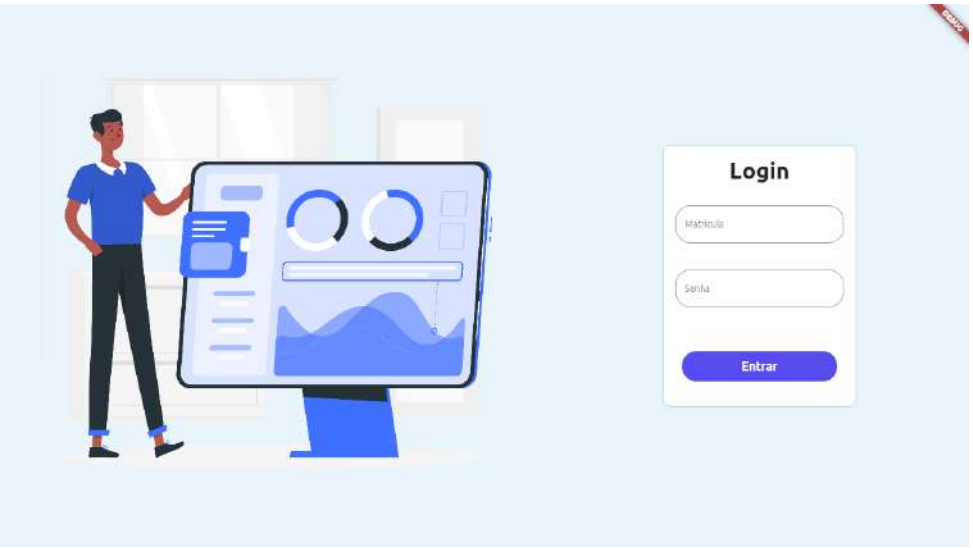
4.2.1 Autenticação e Autorização

A funcionalidade de Autenticação e Autorização é pré-requisito para todas os requisitos funcionais da aplicação. O usuário deve estar devidamente autenticado para ter acesso ao escopo de suas funcionalidades, tal objetivo e atingindo de diferentes modos de acordo com o tipo do usuário. Para os estagiários e preceptores, o sistema prove a autenticação por meio de login no aplicativo.

Eles irão fornecer como credenciais, matricula e senha para ter acesso ao sistema e suas funcionalidades.

O sistema prove aos administradores a autenticação por meio do aplicativo desktop. Forcendo a matricula e senha irão ser devidamente autenticados e terão acesso a suas funcionalidades. Em ambos os casos será gerado um token JWT que será guardado no dispositivo pelo qual o usuário se interfaceia com o sistema, o token é utilizado para autenticações posteriores enquanto o token JWT estiver valido. Para informações mais detalhadas consulte o tópico Autenticação e Autorização.

Figura 4.1 – Tela de Login APP Administração, APP QRCode



Fonte: Elaborado pelo autor (2024)

Figura 4.2 – Tela de Login APP Estagiário, APP Preceptor

Fonte: Elaborado pelo autor (2024)

4.3 REQUISITOS NÃO FUNCIONAIS DE MONITORAMENTO

4.3.1 Serve Side Events

4.3.1.1 Funcionamento

Os chamados serve side events são um dos tipos de comunicação entre servidor e cliente. O cliente inicia a conexão por meio de um cliente SSE enviando uma requisição HTTP ou HTTPS padrão na resposta do servidor o Media-Type irá conter text/event-stream indicando ao cliente a reposta em modo streaming, ou seja indicando ao cliente que irá manter a conexão ativa e passará a enviar eventos. O tempo de vida dessa resposta é configurável neste trabalho ela permanece ativa até que o cliente encerre a conexão.(ROZA, 2024).

4.3.1.2 Eventos

Os eventos são objetos responsáveis pelo tráfego da informação desde o emissor até o cliente. Ele foi implementado no servidor através da combinação de funcionalidades providas pelo Spring e pelo Jedis, uma biblioteca de conexão ao Redis por meio de código Java.

4.3.1.3 KeySpace Notifications

O redis conta com uma funcionalidade chamada keyspace notifications um canal de pub/sub. keyspace notifications permite que um cliente tenha acesso a eventos em tempo real que afetam os dados do redis, operações de inserção,deleção podem ser capturadas e tratadas pela aplicação. Caso o consumer deseje eventos específicos o mesmo, deve fazer um tratamento de strings nos eventos pois como é um canal de pub/sub qualquer evento que altere os dados, é enviado para o consumer.(REDIS, 2024).

4.3.1.4 SseEmitter

O framework Spring possui uma classe especial chamada SseEmitter, esta classe é utilizada para transmissão de dados em tempo real. Esta classe é uma especialização da classe `ResponseBodyEmitter` usada para enviar objetos de forma assíncrona na response, no caso a `SSeEmitter` lida especificamente com Serve-Side Events. (STOYANCHEV et al., 2024).

4.4 REQUISITOS NÃO FUNCIONAIS DE INTEGRIDADE

4.4.1 Soft Delete

O soft delete é uma técnica de gerenciamento de dados. Essa técnica permite a remoção de entidades do sistema sem que haja a deleção do seus dados, essa técnica é utilizada quando é necessário manter o histórico de dados e/ou manter a integridade de dados, no caso deste trabalho o segundo.

A implementação desta técnica consiste na adição de uma nova coluna chamada `deleted` com valores booleanos. A entidade é inicialmente criada com o valor false na coluna `deleted`, caso a entidade seja removida pelo sistema, esse valor é alterado para true. O sistema só irá realizar outras operações de manipulação de dados leitura,alteração ou qualquer outra, somente se o valor de da coluna `deleted` estiver false. (BAELDUNG, 2024).

4.5 REQUISITOS NÃO FUNCIONAIS DE UI

4.5.1 HUB de Entidades (Administradores)

Após à autenticação no sistema, será dado a opção de escolha das entidades. Será mostrado na tela do aplicativo desktop, as opções estagiário,preceptor,administração e setores, ao clicar em uma das opções o administrador acessa as funcionalidades relativas aquela entidade.

Figura 4.3 – Tela de Escolha de Entidade APP Administração

Fonte: Elaborado pelo autor (2024)

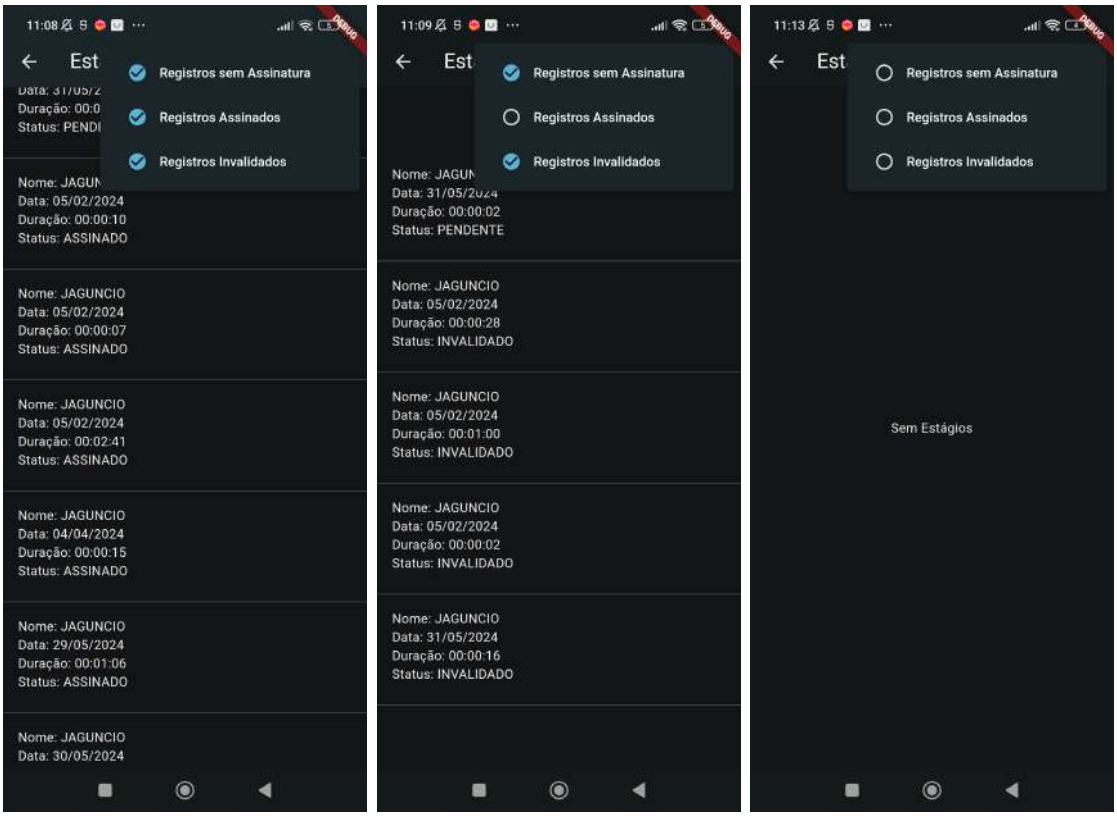
4.5.2 Riverpod

O riverpod é uma biblioteca de gerenciamento de estado para aplicativos flutter. A principal funcionalidade do riverpod é podermos atualizar o contexto de dados em qualquer parte da aplicação graças ao escopo global fornecido pela classe `ProvideScope`.Por ser global, cada mudança de valores ocorrida nas variáveis persistidas pelo riverpod, são atualizadas tanto em valores como também UIs que as utilizam.(ROUSSELET, 2023).

4.5.3 Filtragem Visual dos Registros nos Aplicativos

Para melhorar a visualização dos registros foi implementado no aplicativo do estagiário um mecanismo de filtragem, esse mecanismo também poderia posteriormente ser implementado da mesma forma no aplicativo do preceptor. O primeiro passo foi elaborar um meio de sinalizar quais registro devem ou não ser mostrados, para isso foi usado uma lista de maps, o map é uma estrutura de dados de chave e valor, cada tipo de status gera uma estrutura map onde na chave temos o próprio status em forma de enum, e no valor temos um booleano indicando se ele deve ou não estar visível na tela e depois é criado uma lista vazia, essa lista vazia será a lista que será mostrada na UI então os elementos são adicionados nessa lista baseados no valor booleano dos elementos maps, se um determinado status no map estiver true, todos os registros desse status são adicionados a lista geral.

Figura 4.4 – Listagem dos Estágios do Estagiário



Fonte: Elaborado pelo autor (2024)

4.5.4 Operações de Manipulação Básica de Dados (CRUD)

A manipulação de dados das entidades do sistema, segue a mesma lógica no que tange as operações básicas. A entidade Administrador, possui acesso a todas as operações CRUD de todas entidades, enquanto as outras entidades (Estagiario,Preceptor,Setor) possuem apenas acesso a operação de leitura do seu respectivo tipo. Além disso para qualquer operação, o individuo deve estar devidamente autenticado, em posse do token JWT cada requisição que fizer deve conter o respectivo token na header Authorization do cabeçalho, nas requisições HTTP ou HTTPS.

4.5.4.1 Criação:

Na criação das entidades, o backend irá receber uma requisição do tipo POST em http(s)://domínio/entidade/create, no body dessa requisição deve conter os dados necessários para criação de cada entidade, caso os dados sejam válidos é criado um objeto da entidade e o mesmo é salvo no banco de dados, e então retorna uma resposta com status 201.

4.5.4.2 Consulta:

Para a consulta das entidades, o client deve enviar uma requisição do tipo POST com o atributo identificador da entidade (ID) no body da requisição para a rota http(s)://domínio/entidade, o backend faz a consulta da entidade e caso exista, retorna a resposta com status 200 e os dados no body em formato JSON.

4.5.4.3 Atualização:

Para alteração edição total de dados de uma entidade, o cliente deve envia uma requisição do tipo PUT para a rota http(s)://domínio/entidade, em requisições do tipo PUT todos os dados da entidade são alterados, então o cliente deve enviar no body da requisição, todos os dados da entidade, sejam eles alterados ou não. Caso a entidade exista o backend altera os dados e retorna status 200 na resposta.

4.5.4.4 Deleção:

Na remoção o cliente deve enviar uma requisição do tipo DELETE, para a rota http(s)://domínio/entidade, no body da requisição deve conter o atributo identificador da requisição (ID), o backend então em posse do ID realiza a operação Soft Delete, caso não encontre problemas retorna uma resposta com status 200 ao cliente.

4.6 RFS - 1(ESTÁGIARIO),3(ADMINISTRADOR),4(PRECEPTOR),8(SETOR)

As RFs 1,3,4,8 tratam das operações de Operações de Manipulação Básica de Dados (CRUD), das entidades Estagiário,Administrador,Preceptor e Setor respectivamente.Cada entidade conta com os seguintes dados:

Estagiário:matrícula(String)(ID),senha(String),nome(String)

Preceptor:matrícula(String)(ID),senha(String),nome(String),setor(String)

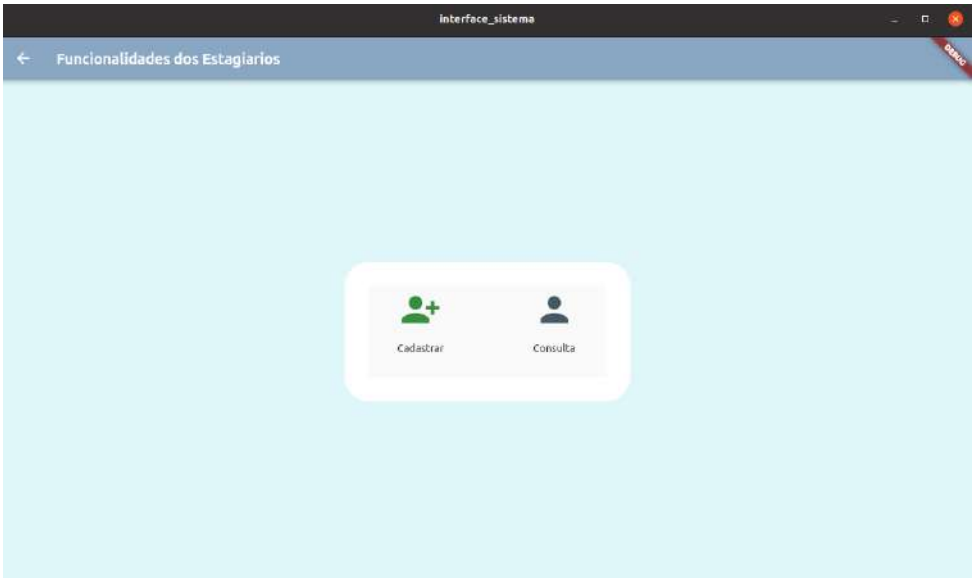
Administrador:matrícula(String)(ID),senha(String),nome(String)

Setor:id(int)(ID),nome(String)

Exemplo de telas:

Tela exemplo onde é listado todas as funcionalidades das entidades.Para acessar a funcionalidade basta clicar na mesma.

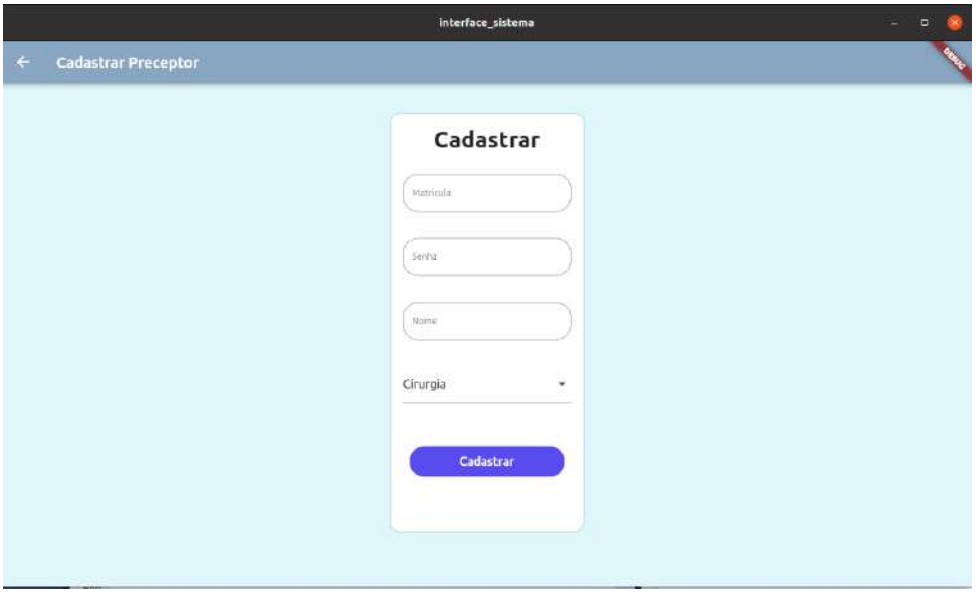
Figura 4.5 – Tela de Funcionalidades de Estagiário



Fonte: Elaborado pelo autor (2024)

Tela exemplo onde é feito o cadastramento das entidades. Conforme mostra no formulário o administrador deve preencher todas as informações requiridas no formulário e então aperta no botão cadastrar.

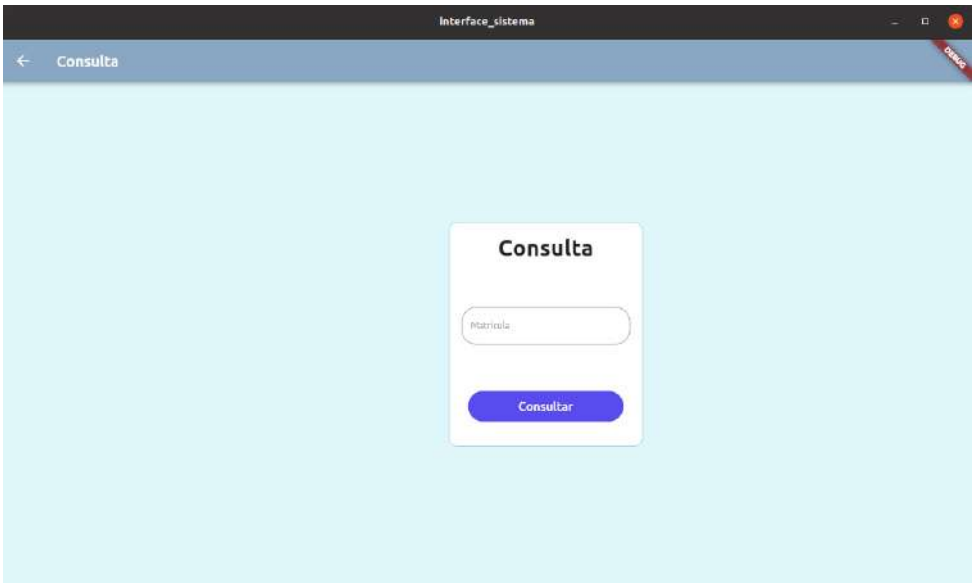
Figura 4.6 – Tela de Cadastramento de Preceptor



Fonte: Elaborado pelo autor (2024)

Tela de exemplo onde é realizado a consulta. O administrador irá preencher o formulário com o atributo identificador da entidade e então clicar em consultar.

Figura 4.7 – Tela de Consulta do Estagiário

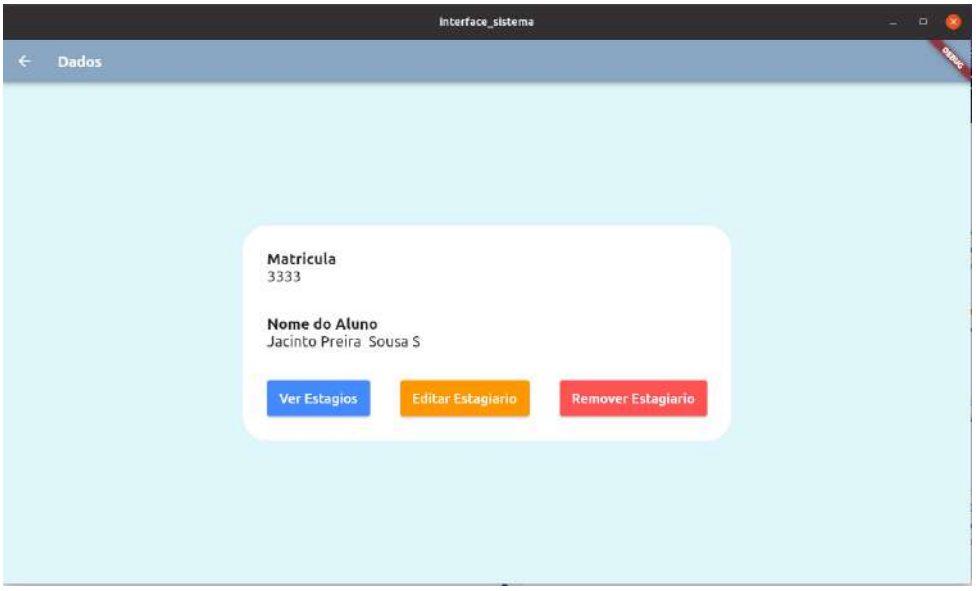


Fonte: Elaborado pelo autor (2024)

Ao fim de consulta, a tela mostrada será a tela de dados da entidade.

Exemplo de tela de dados da entidade.nesta tela reside o acesso as funcionalidades de deleção e alteração das entidades.

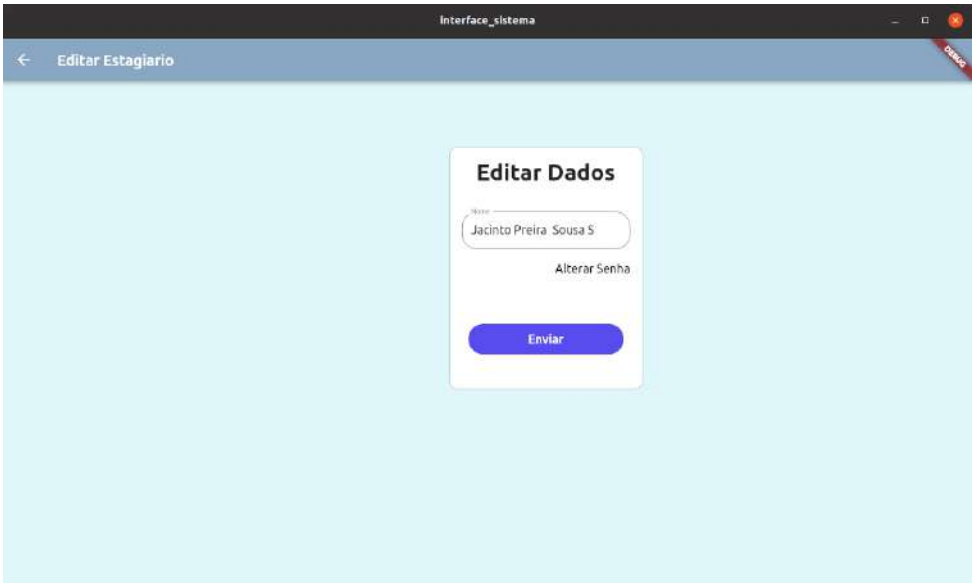
Figura 4.8 – Tela de Dados com Operações do Estagiário



Fonte: Elaborado pelo autor (2024)

Tela Exemplo da edição de dados das entidades,o formulário já estará pré preenchido com os dados atuais da entidade então o administrador altera os dados que deseja e envia-os novamente.

Figura 4.9 – Tela de Edição de Dados do Estagiário



Fonte: Elaborado pelo autor (2024)

Para visualizar todas as telas do Operações de Manipulação Básica de Dados (CRUD), consulte o apêndice.

4.7 RF2 - GESTÃO DE USUÁRIOS (ACOMPANHAMENTO DE ESTAGIÁRIOS)

4.7.1 Acompanhamento de Estagiários - Visão Usuário

Para o acompanhamento de estagiários o administrador após escolher a entidade setor no hub de entidades, após isso ele irá fazer uma consulta do setor que ele quer monitorar utilizando o código do setor, na tela das informações do setor possui um botão com ícone de pessoa, ao clicar no botão abre um painel lateral contendo uma lista de todos os estagiários com processo de check-in aberto naquele setor.

4.7.2 Acompanhamento de Estagiários - Visão Técnica

Na implementação técnica do acompanhamento, foi dividido em 2 partes que ocorrem de forma síncrona. Primeiro termos o contexto atual do redis para saber quais estágios estão abertos no momento, e segundo ativar o listener que irá notificar caso um novos estágios abertos.

4.7.2.1 Get-Context

Para a primeira parte, do lado do frontend, envia-se uma requisição GET ao backend, o backend então faz um scan de todas as chaves do redis, após isso ele filtra quais são instancias da classe `CheckInOpen` (a filtragem é necessária a chance de no scan ser retornado objetos `CheckInClosed`, objetos esse que indicam o fim do estágio então eles seriam um problema no contexto da RF) após a filtragem, ele envia uma lista de objetos da classe `EstagiarioEvent`. Essa classe contém 3 atributos, `opPeformed`,`nome`,`setor`.

A classe `EstagiarioEvent` é a classe por meio do qual o backend envia as informações de novas aberturas de estágios. O atributo `opPeformed` indica o tipo de operação feita no estágio, indicando ao frontend se ele deve ser disponibilizado na tela ("open") ou retirado da tela ("close"), a propriedade `nome` refere-se ao nome do estagiário do estágio,a `matricula` que refere-se a matricula do estagiário, o `setor` e `setorId` sendo respectivamente o nome do setor do qual o estágio pertence e o id.

Ainda na primeira parte, o backend então envia uma lista de objetos `EstagiarioEvent` que é armazenada no controller riverpod no frontend que é responsável por manter o estado dos dados atualizado com isso a primeira parte que é pegar o estado dos estágios é finalizada.

4.7.2.2 Stream:

A segunda parte refere-se a atualização em tempo real de novos estágios abertos, após a primeira parte ser concluída com sucesso, o frontend automaticamente ou seja ainda no mesmo fluxo, envia uma nova requisição GET ao backend, requisição essa que inicia o processo de SSE.

No backend ao iniciar a aplicação é instanciado um bean da classe `RedisMessageListener`, esse bean contem uma função chamada `onMessage`, tal função recebe como parâmetros um objeto da classe `Message` e um array de byte este não utilizado.A classe contém 2 atributos, são eles o `body` e o `channel`, o `body` é uma string que contém a operação de alteração de dados feita no evento, já o `channel` é uma string que contém 2 strings concatenadas, a primeira refere-se ao canal pub/sub nesse contexto o keyspace na segunda string contém a chave do objeto map redis na qual foi realizado a operação.

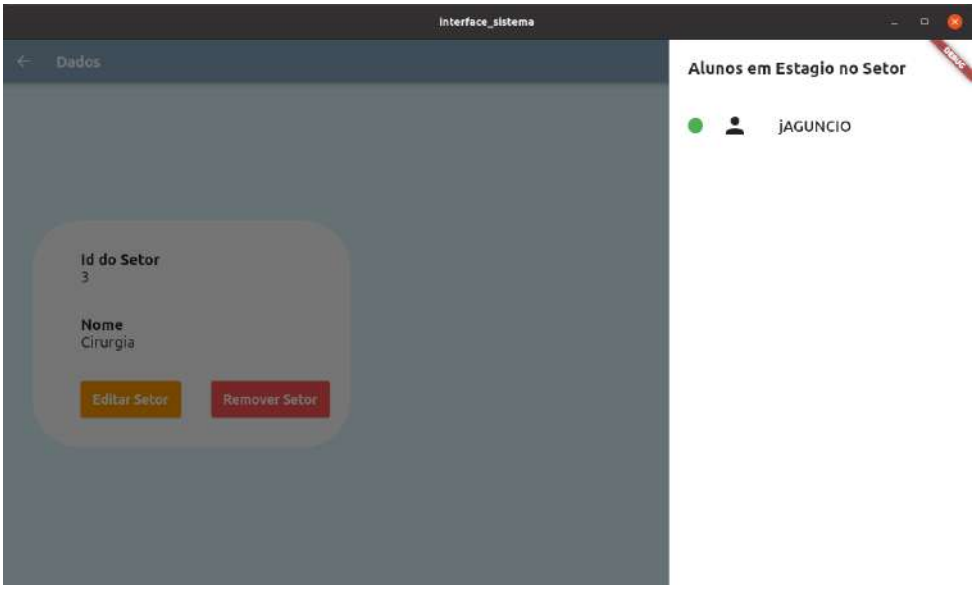
Explicado a estrutura do objeto `message`, a próxima parte é a filtragem de eventos pois como já foi falado no parágrafo sobre keyspace, a aplicação captura todos os eventos então a aplicação retém apenas os objetos `message` do qual o atributo `body` contém a string "HSET"simbolizando as operações de abertura e fechamento do check-in, após a filtragem utilizamos o outro atributo da `message` o `channel` para recuperar dados do estagiário tendo em vista que a chave do objeto do redis contida no `channel` é o mesmo ID do estagiário possibilitando a consulta de dados do mesmo.

O backend então recupera da base de dados o nome do estagiário e parte para uma 2 filtragem.

Na segunda filtragem o backend verifica se o estágio está fechado ou aberto, verificação necessária pois a operação HSET do redis e performada para ambos, essa filtragem é feita apenas usando o operador do java "instanceoff"com as classes de `ChecKInOpen` e `CheckInClosed`.

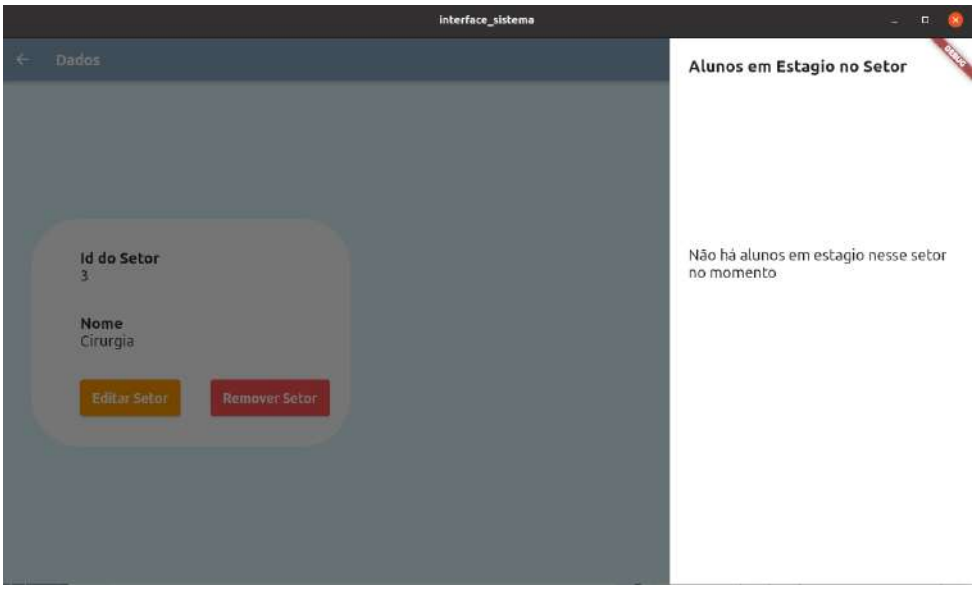
Por fim após sabermos se evento desse estágio foi uma abertura ou fechamento e feito um objeto da classe `EstagiarioEvent`, a string de `opPerformed` depende do tipo do estágio sendo "open"para o tipo `CheckInOpen` e "closed"para `CheckInkClosed`, no segundo atributo temos o `nome` do estagiário já recuperado via `channel`, a `matricula` também obtida através do `channel`,o restante dos atributos `setor` e `setorId` obtidos através do valor do objeto armazenado no redis sendo esse obtido através da chave contida no `channel` o objeto então e enviado para o frontend. O frontend em posse da lista inicial de eventos (a obtida na operação de Get-Context), realiza operação de inserção do objeto `EstagiarioEvent` caso a string `opPerformed` seja igual a "open"e remoção de objeto caso seja "closed".

Figura 4.10 – Tela de Acompanhamento com Estagiário em Estágio



Fonte: Elaborado pelo autor (2024)

Figura 4.11 – Tela de Acompanhamento sem Estagiário em Estágio



Fonte: Elaborado pelo autor (2024)

4.8 RF6 - GESTÃO DE REGISTRO (REGISTRO)

A RF 6 trata das operações básicas de manipulação de dados dos registros de estágio.

4.8.1 Registro - Criação

Para criação de um processo de estágio, o estagiário deve realizar os processos de check-in e check-out no seu aplicativo.

4.8.1.1 Check-In - Visão Usuário

O processo de Check-In é feito na tela principal do aplicativo do estagiário, o estagiário aperta no botão Check-In localizado no centro da tela.

Figura 4.12 – Tela de Inicial APP do Estagiário

Fonte: Elaborado pelo autor (2024)

4.8.1.2 Scaneamento de QR-Code

Em seguida o mesmo e direcionado para a tela de leitura de QR-Code, os QR-Codes que estão espalhados pelo hospital fornecem a informação de qual Setor o estagiário iniciou o processo de Check-In.

Figura 4.13 – QrCode disponibilizado pelo toten no setor de Cirurgia

Fonte: Elaborado pelo autor (2024)

Figura 4.14 – Tela do Scanner de QRCode do APP do Estagiário

Fonte: Elaborado pelo autor (2024)

4.8.1.3 Finalização de Check-In

Após o scaneamento do QR-Code ele retorna a tela principal, a tela principal passa por uma atualização, o botão de Check-In é substituido pelo botão de Check-Out usado para fechar o processo de estágio e o Check-In está finalizado.

4.8.1.4 Check-Out - Visão Usuário

O processo de Check-Out inicia-se após a conclusão do período de estágio, o estagiário aperta o botão de Check-Out que está posicionado no centro da tela e inicia o processo de fechamento do registro.

Figura 4.15 – Tela do Checkout APP do Estagiário

Fonte: Elaborado pelo autor (2024)

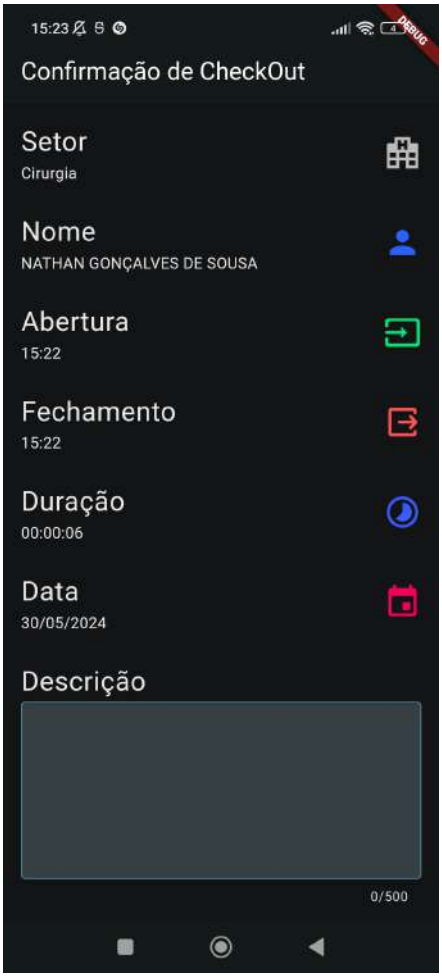
4.8.1.5 Conferencia de dados e finalização de Registro

Logo em seguida o estagiário é enviado para a tela de confirmação de Check-Out do qual o estagiário irá conferir dados relevantes a cerca do estágio como: horários de

abertura e fechamento,duração setor, o nome completo e a data, além disso o estagiário tem a opção de escrever como foi o seu processo de estagio do dia no campo descrição.

Após conferir os dados e colocar a descrição, ele aperta em no botão confirmar e é redirecionado para a tela inicial, com a mensagem de Check-In concluído com sucesso finalizando assim o processo de fechamento do registro Check-Out.

Figura 4.16 – Tela de Conferencia de Dados do Estágio APP do Estagiário



Fonte: Elaborado pelo autor (2024)

Figura 4.17 – Tela de Estágio Finalizado Com Sucesso APP do Estagiário

Fonte: Elaborado pelo autor (2024)

4.8.1.6 Check-In - Visão Técnica

O aplicativo envia uma requisição POST para o backend com os dados do setor no body e um token de autenticação JWT do estagiário, após recebida a requisição, é criado um novo objeto chamado `CheckInOpen`. O `CheckInOpen` é uma classe que contém como atributos a `matricula` do estagiário, dados do setor do qual ele leu o qr code (`id`,`nome`) e um `timestamp`, `timestamp` esse que é usado para o cálculo do tempo de estágio daquele registro, após a criação do objeto ele é armazenado no redis, como o redis e um banco de dados NoSQL de chave valor a chave será a `matricula` do estagiário e o valor um JSON da classe `CheckInOpen`.

4.8.1.7 Check-Out - Visão Técnica

O aplicativo envia uma requisição POST para o backend com um token de autenticação JWT , o backend então utiliza a `matricula` contida no token para recuperar o objeto `CheckInOpen` colocado anteriormente no redis. Após isso a aplicação faz um novo

objeto chamado `CheckInClosed`, esse objeto contém todas as informações do primeiro com um `timestamp` adicional representando o tempo de fechamento do registro, o mesmo e colocado no redis utilizando a mesma chave do `CheckInOpen`.

Após isso backend prepara um novo objeto chamado `InfoCheckIn`, esse objeto contém o `nome` do estagiário,o `nome` do setor do qual foi feito o registro,os `horários de abertura` e `fechamento do registro` e a `data do registro`, essas informações são enviadas para a aplicação do estagiário para o mesmo conferir as informações do registro antes de finaliza-lo.

Após a conferencia dos dados do registro o estagiário caso queira coloca a `descrição` do estágio e envia ao backend. O backend recebe essa `descrição`, retira o objeto `CheckInClosed` colocado no redis anteriormente utilizando a `matricula` do estagiário contida no token JWT, junta a `descrição` mais as informações contidas em `CheckInClosed` criando um novo objeto de `Registro` e o salva no banco de dados.

4.8.1.8 Status Inicial de Registro

Os registros contém status, os status definem a situação atual daquele registro podendo ser dos tipos: `sem assinatura`, `assinados` pelo preceptor ou `rejeitados` pelo preceptor.

Quando um registro é criado ele recebe o status inicial de sem assinatura, os `registros sem assinatura` são enviados para os preceptores responsáveis pelo setor do qual o registro foi criado por meio do aplicativo do preceptor.

4.8.2 Leitura - Visão Usuário

A leitura dos registros é feita por todas as entidades. Os estagiários tem acesso aos seus registros através do aplicativo, na página inicial do aplicativo existe um botão com o nome "Meus Estágios"ao clicar o estagiário visualiza uma lista com todos os seus registros, cada item da lista mostra o nome do estagiário,a data,o tempo e o status do registro, os itens são botões ao ser apertado o estagiário vai para uma tela onde mostra todas as informações do registro.

Os preceptores contam com um estilo de visualização parecida, eles também possuem a mesma implementação da lista e da visualização de todos os dados do registro, a diferença consiste em eles não possuem a filtragem por status, já que os registros visualizados pelos preceptores possuem apenas o status de "em análise"e a lista é mostrada já na tela inicial do aplicativo dos preceptores.

Os administradores podem visualizar os registros através do aplicativo desktop. Primeiro o administrador deve realizar a consulta do estagiário do qual ele deseja visualizar os registros,após o resultado na tela de informações há um botão com o nome "Ver Estágios"ao clicar nesse botão o administrador vai para uma tela onde é mostrado os tipos de registros de acordo com o status ao clicar o administrador vai para uma tela onde mostrado uma lista com todos os registros de um status específico.Cada item da lista mostra a data do registro,o setor e caso ele esteja assinado ou invalidado também mostra o nome do preceptor, ao clicar na lupa é aberto uma caixa de dialogo contendo todas as informações do registro.

4.8.3 Leitura - Visão Técnica

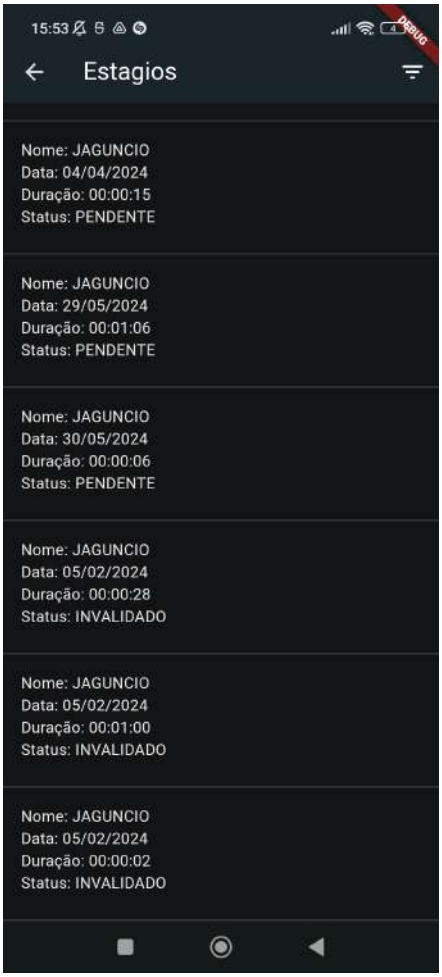
Ao clicar em "Meus Estágios"o aplicativo faz uma requisição ao backend, o backend decodifica o token JWT para extrair a matricula do JWT, com a `matricula` o backend faz uma consulta SQL com todos os registros do estagiário, depois aglutina o resultado em uma classe que contém 3 propriedades, `registrosSemAssinaturas`, `registrosAssinados`, `registrosRejeitados`, cada propriedade é uma lista de registros com o mesmo status. Após fazer o objeto, ele e enviado ao aplicativo em formato JSON.

Após receber o JSON com todos os registros, ele aplica a técnica de filtragem dos registros com o JSON.

O aplicativo dos preceptores faz uma requisição POST automaticamente após o processo de login ao backend, o backend utiliza a `matricula` contida no token JWT para realizar uma consulta, o resultado da consulta é todos os `registros sem assinatura` pertencentes ao setor do qual o preceptor está alocado, os dados são enviados de volta ao aplicativo em formato JSON.

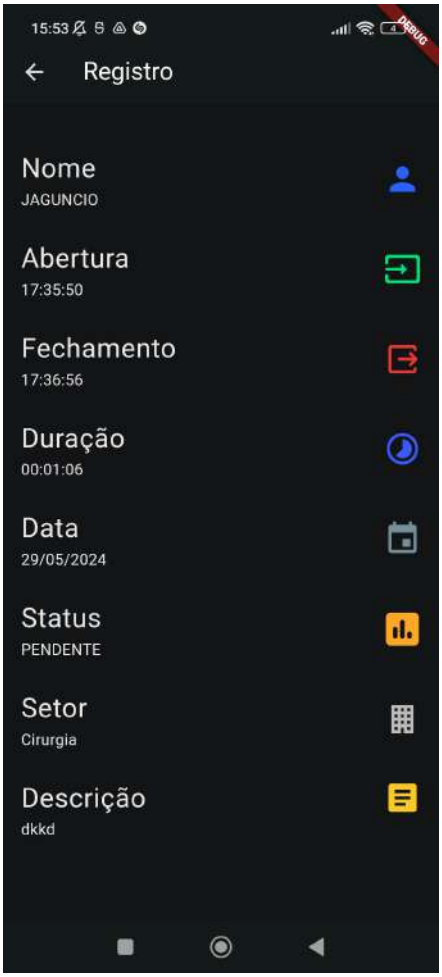
Após o administrador escolher o tipo de registro, o aplicativo desktop envia uma requisição POST ao backend, o administrador acessa apenas um tipo especifico de registros por vez, o backend possui para cada tipo de registro, o acesso por meio de uma rota especifica. Com a `matricula` do estagiário o backend faz uma consulta SQL de todos os registros específicos da rota, e envia de volta ao frontend com o formato JSON. O frotend então faz a listagem dos registros em uma nova tela.

Figura 4.18 – Listagem de Estágios APP Estagiário



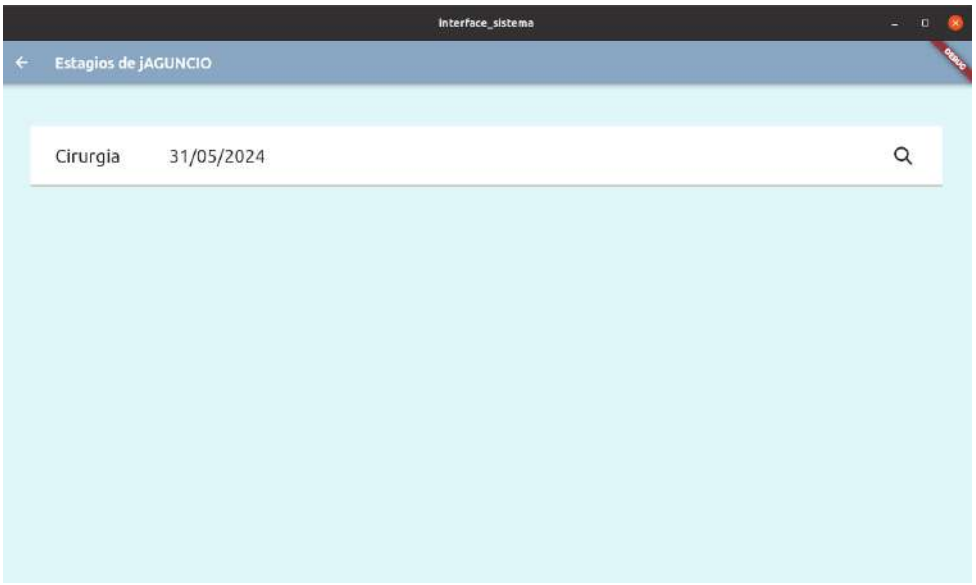
Fonte: Elaborado pelo autor (2024)

Figura 4.19 – Detalhes do Estágio APP Estagiário



Fonte: Elaborado pelo autor (2024)

Figura 4.20 – Listagem de Estágios em Análise APP Administração



Fonte: Elaborado pelo autor (2024)

Figura 4.21 – Detalhe de Estágio em Análise APP Administração

Fonte: Elaborado pelo autor (2024)

Figura 4.22 – Listagem de Estágios Assinados e Rejeitados APP Administração

Fonte: Elaborado pelo autor (2024)

Figura 4.23 – Detalhes de Estágio Assinado APP Administração

Fonte: Elaborado pelo autor (2024)

Figura 4.24 – Detalhes de Estágio Rejeitado APP Administração

Fonte: Elaborado pelo autor (2024)

4.8.4 Deleção e Atualização

As operações de deleção e atualização não foi implementada para manter a integridade dos registros pois os mesmos podem posteriormente serem usados para processos de auditoria.

4.9 RF7 - GESTÃO DE REGISTO (ASSINATURA)

4.9.1 Visão Usuário

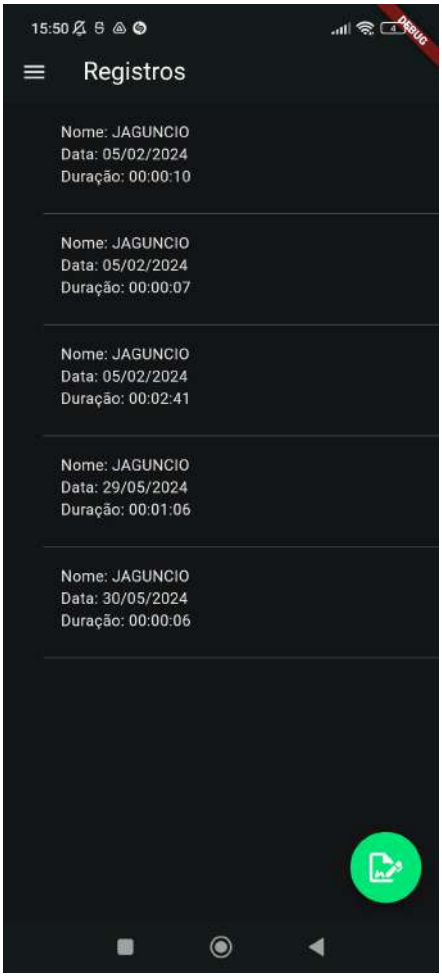
A validação dos registros é realizada pelos preceptores responsáveis do setor do qual o registro foi feito. O preceptor acessa seu aplicativo e na primeira tela pós autenticação, irá aparecer uma lista com todos os registros sem assinatura do setor em que o mesmo trabalha. Para assinar o preceptor irá apertar no botão verde flutuante com o simbolo de papel assinado, ao fazer isso irá aparecer uma caixa de dialogo para confirmação, ao confirmar o preceptor assina todos os registros da lista.

4.9.2 Visão Técnica

Ao passar da tela de autenticação, o aplicativo do preceptor irá automaticamente enviar uma requisição com o verbo GET para o backend, o backend irá usa a `matricula` do preceptor que está dentro do token JWT, para fazer a consulta no banco de dados sobre quais registros pertencem ao setor dele, e não estão nem `assinados` nem `rejeitados`.Após isso é feito um novo objeto com os dados resultante da consulta, o objeto foi chamado de `RegistryUI` ele contém o `nome` do estagiário, os horários de `entrada` e `saída`, a `duração` e a `descrição` do registro. Como mais de 1 registro pode ser recuperado, os objetos `RegistryUI` são colocados em uma lista e enviados para o aplicativo do preceptor.

Após o processo de checagem dos registros e a confirmação de assinar todos os registros restantes pela caixa de dialogo do aplicativo do preceptor, a aplicação envia uma requisição POST com autenticação JWT, o backend recupera todos `registros sem assinaturas` daquele preceptor utilizando a `matricula` contida no JWT, após a recuperação os registros é feito uma lista de `RegistrosAssinados`, essa entidade contém todos os dados do registro mais a `matricula` do preceptor indicando que o registro foi validado, a lista com a nova entidade e salva no banco de dados.

Figura 4.25 – Lista de Estágios em Análise APP do Preceptor



Fonte: Elaborado pelo autor (2024)

Figura 4.26 – Assinatura de Estágios APP Preceptor

Fonte: Elaborado pelo autor (2024)

4.10 RF9 - GERAÇÃO DE QR CODE

4.10.1 Visão Usuário

Após a tela de autenticação no toten, o usuário escolhe qual dos setores cadastrados que ele deseja representar no QRCode, ao apertar no botão, o aplicativo gera um QRCode que contém o nome do setor e o id dele no banco de dados. Caso o usuário deseje trocar o qrcode basta apertar no botão localizado no canto superior direito, fazer o processo de autenticação novamente e ele irá ser redirecionado para a ela de escolha de setor novamente.

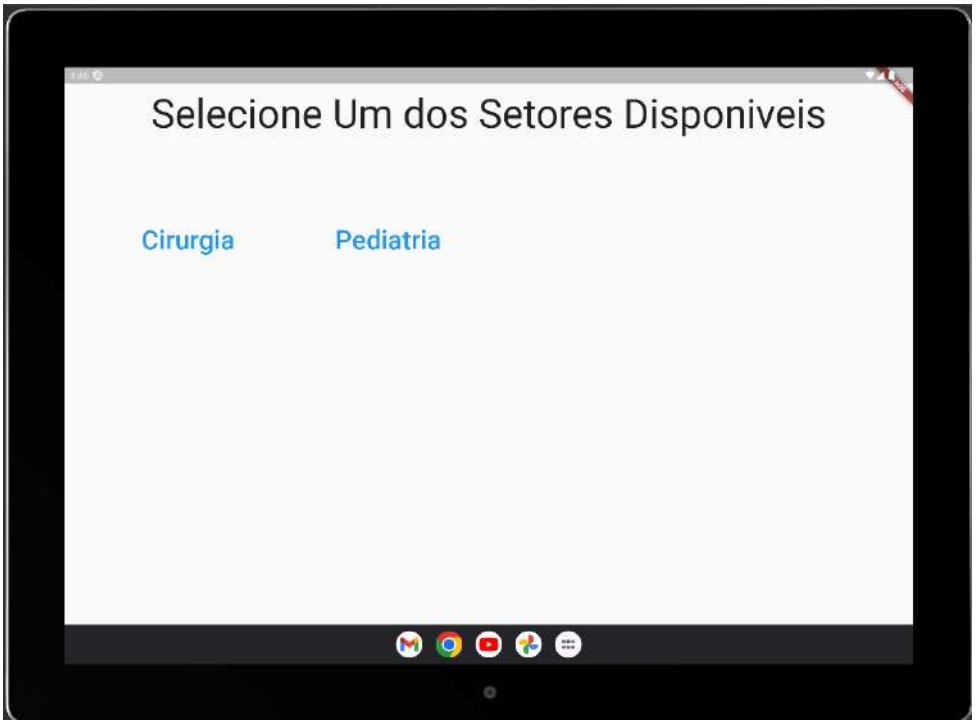
4.10.2 Visão Técnica

O aplicativo do toten envia uma requisição GET ao backend, o backend envia os dados do `id` e `nome` de todos os setores cadastrados no banco de dados, o toten faz um botão para cada setor, ao clicar as informações do setor são enviadas para uma nova tela,

na nova tela, a biblioteca qrflutter irá utilizar esses dados para fazer o qrcode, após feito ele é mostrado no centro da tela.

Caso o usuário deseje troca de QrCode o aplicativo toten irá disponibilizar formulário de autenticação padrão com `matricula` e `senha` e caso ele seja autorizado, ele irá ser redirecionado para a tela de escolha de setor.

Figura 4.27 – Tela de Seleção de Setores para disponibilização de QRCode



Fonte: Elaborado pelo autor (2024)

Figura 4.28 – Autenticação para Mudança de QRCode

Fonte: Elaborado pelo autor (2024)

4.11 RF11 - GESTÃO DE REGISTO (INVALIDAÇÃO)

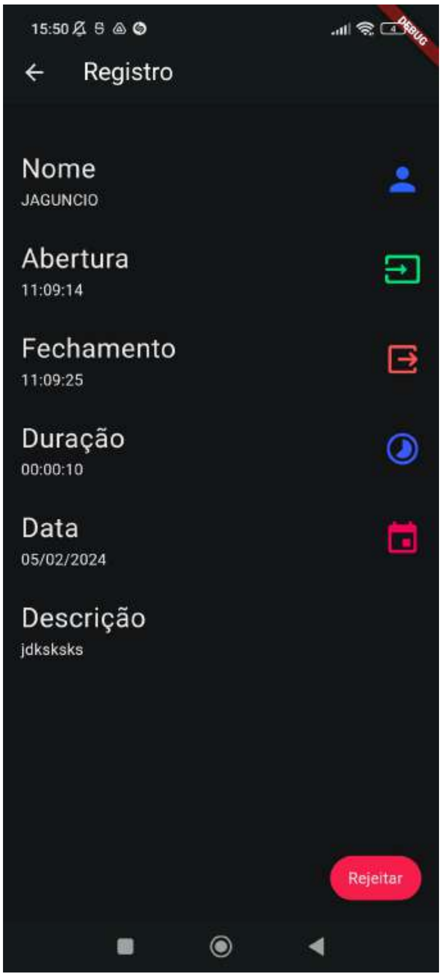
4.11.1 Visão Usuário

A invalidação dos registros é feita pelos preceptores através do aplicativo dos mesmos, após a tela de login, aparecerá uma lista dos registros sem assinatura do preceptor, para rejeitar o preceptor irá apertar no registro, após apertar ele verá as informações mais importantes do registro, ao final há o botão Rejeitar após apertar o preceptor escolhe ou "O aluno não compareceu", uma das opções já pré definidas ou ele pode escolher a opção "Outro"e deixar uma mensagem explicando melhor o motivo.

4.11.2 Visão Técnica

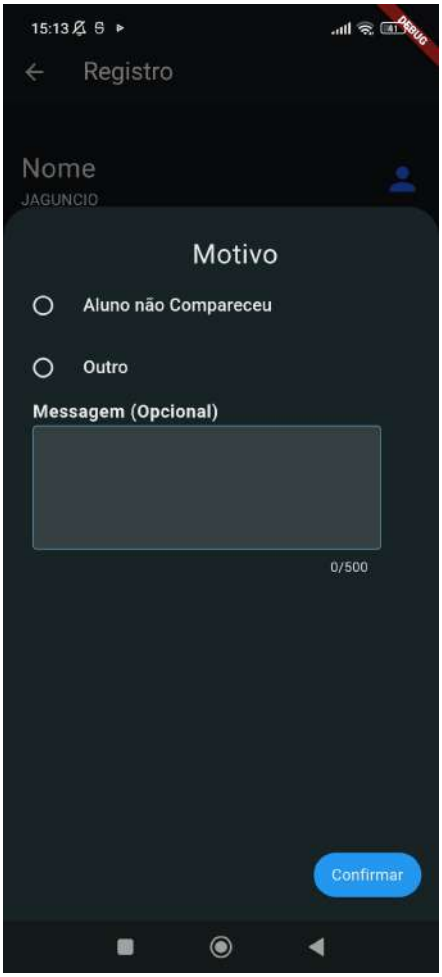
Após o preceptor invalidar o registro pelo aplicativo, o aplicativo envia uma requisição POST para o backend, no body da requisição contém o `id` do registro invalidado, o `motivo` e a `mensagem`. O backend então recupera os dados do preceptor com a `matricula` contida no JWT, o registro com o `id` do registro no body da requisição e cria o objeto `RegistroInvalidado`, que contém os dados do preceptor que invalidou , os dados básico do registro, o `motivo` e a `mensagem`, após ser criado objeto, ele é salvo no banco de dados.

Figura 4.29 – Detalhes de Estágio APP Preceptor



Fonte: Elaborado pelo autor (2024)

Figura 4.30 – Rejeição de Estágio APP Preceptor



Fonte: Elaborado pelo autor (2024)

As funcionalidades apresentadas neste capítulo tem como objetivo promover o suporte e um controle maior aos processos de estágios no HUJB.

4.12 FUNCIONALIDADES FUTURAS

Infelizmente para o presente trabalho as funcionalidades das RFs 5 e 10 não foram implementadas.

4.12.1 RF5

O principal fator motivador da criação da RF5, foi promover mecanismos de Autenticação e Autorização, mecanismos esses que foram implementados nessa versão atual do software a nível de entidade ou seja acada administrador terá acesso a todas as funcionalidades disponíveis para os administradores.Já a RF5 prove o mesmo mecanismo porém com acesso sendo ainda mais especifico, a nível de usuário, portanto a RF5 não

foi implementada nessa versão podendo posteriormente ser implementada ao expandir os mecanismos de Autenticação e Autorização a usuários, e não somente a entidades como é atualmente.

4.12.2 RF10

O foco do presente trabalho consiste nos processos de estágios do HUJB, a RF10 não está diretamente relacionada com o foco do trabalho por isso apesar de ser importante, ela não foi implementada nesta versão do projeto.

4.12.3 Proposta - RF12 Geração de Relatórios

O software deve contar com um serviço de geração de relatórios para visualização gráfica dos dados de estágios, como por exemplo a quantidade de horas de estágio que determinado aluno fez em determinado setor. Com esse relatório a gestão, teria um controle maior sobre o desempenho dos alunos no geral.

5 CONSIDERAÇÕES FINAIS

Espera-se que ao final deste projeto, o artefato de software desenvolvido ajude na automatização do Hospital Universitário Júlio Bandeira, melhorando não só a qualidade do hospital como instituição educadora, mas também o aprendizado dos estudantes que dela compõem.

5.1 CONTRIBUIÇÕES

- Automatização parcial das etapas do processo de estágio no HUJB.
- Coleta de dados dos estágios, podendo estes através de BI serem utilizados para melhoria do processo.
- Visualização dos dados dos estágios pelos stakeholders a qualquer momento.
- Melhor controle dos processos de estágios.
- Melhor segurança de acesso aos dados dos estágios.

5.2 TRABALHOS FUTUROS

O ponto de partida para continuação deste presente trabalho pode ser a implementação das FUNCIONALIDADES FUTURAS na seção 4.12, estas que conferem funções de apoio aos processos de estagio.

Após essas implementações, podemos pensar também sobre a 2 versão do aplicativo, esta contendo integrações (caso possível) com os sistemas acadêmicos da UFCG, para uma maior automatização de todos os tramites de estágios no HUJB.

REFERÊNCIAS

AUTH0. **JWT - Introduction to JSON Web Tokens**. 2024. <https://jwt.io/introduction>. [Accessed 20-Mai-2024].

BAELDUNG. **Soft Delete - How to Implement a Soft Delete with Spring JPA**. 2024. <https://www.baeldung.com/spring-jpa-soft-delete>. [Reviewed by Bruno Fontana, Accessed 24-Mar-2024].

Colegiado Pleno do Conselho Universitário. **RESOLUÇÃO Nº 06/2015**. 2015. Available at <https://www.gov.br/ebserh/pt-br/hospitais-universitarios/regiao-nordeste/hujb-ufcg/acesso-a-informacao/institucional/RegimentointernodoHUJBantigo.pdf> (2023/02/12).

Empresa Brasileira de Serviços Hospitalares. **Institucional - Empresa Brasileira de Serviços Hospitalares**. 2020. Available at <https://www.gov.br/ebserh/pt-br/hospitais-universitarios/regiao-nordeste/hujb-ufcg/acesso-a-informacao/institucional> (2023/02/12).

FLUTTER. **Flutter - What is Flutter?** 2024. <https://docs.flutter.dev/resources/faq/>. [Accessed 20-Mar-2024].

PAGOTTO, T.; FABRI, J. A.; LERARIO, A.; GONÇALVES, J. A. Scrum solo: Software process for individual development. In: IEEE. **2016 11th Iberian Conference on Information Systems and Technologies (CISTI)**. [S.l.], 2016. p. 1–6.

PostgreSQL Global Development Group. **PostgreSQL — postgresql.org**. 2023. <https://www.postgresql.org/>. [Accessed 12-Feb-2023].

REDIS. **Redis - Redis keyspace notifications**. 2024. <https://redis.io/docs/latest/develop/use/keyspace-notifications/>. [Accessed 15-Mar-2024].

ROUSSELET, R. **Riverpod – A Reactive Caching and Data-binding Framework**. 2023. <https://riverpod.dev/>. [Accessed 25-Nov-2023].

ROZA, G. **SSE - Server-Sent Events in Spring**. 2024. <https://www.baeldung.com/spring-server-sent-events>. [Reviewed by Michal Aibin, Accessed 16-Mar-2024].

SCHWABER, K.; SUTHERLAND, J. O guia do scrum. **Scrumguides. Org**, v. 1, p. 21, 2013.

STOYANCHEV, R.; HOELLER, J.; BRANNEN, S.; CLOZEL, B. **SseEmitter - Class SseEmitter**. 2024. <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/servlet/mvc/method/annotation/SseEmitter.html>. [Accessed 16-Mar-2024].

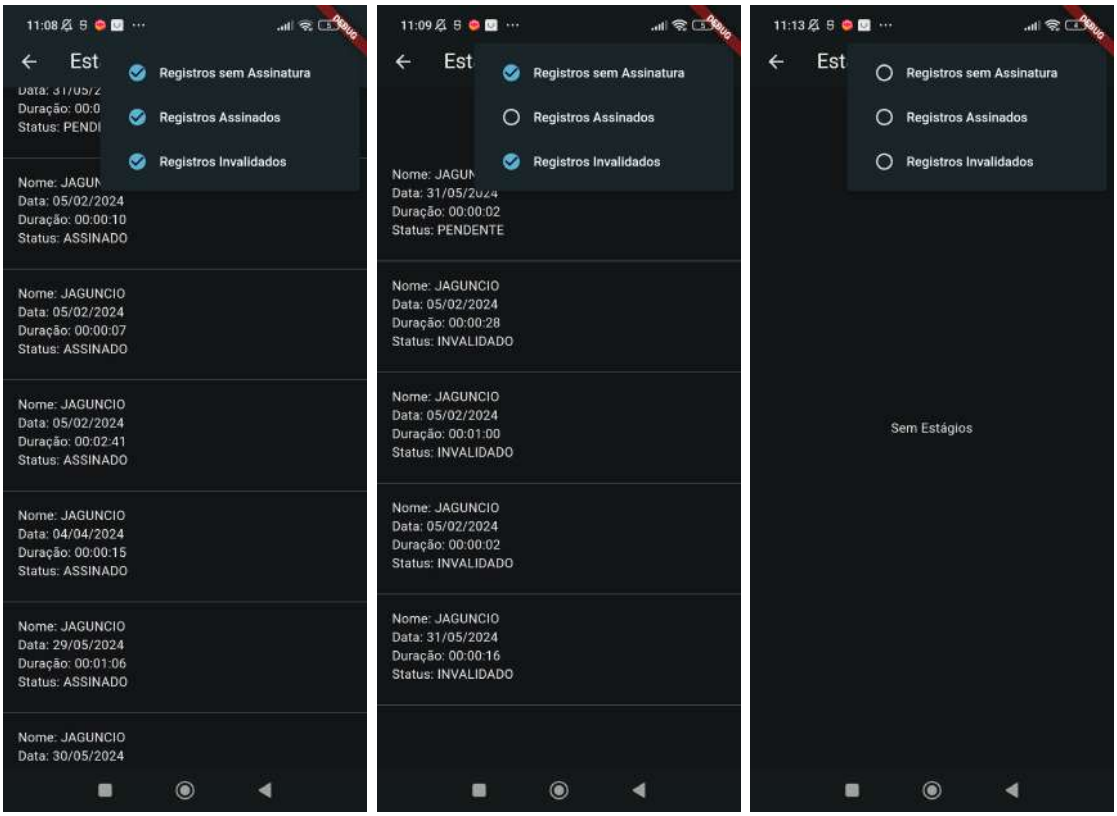
TANZU, V. **Spring.io**. 2023. <https://spring.io/>. [Accessed 12-Feb-2023].

APÊNDICE A – TELAS DO APLICATIVO

Figura A.1 – Tela de Escolha de Entidade APP Administração

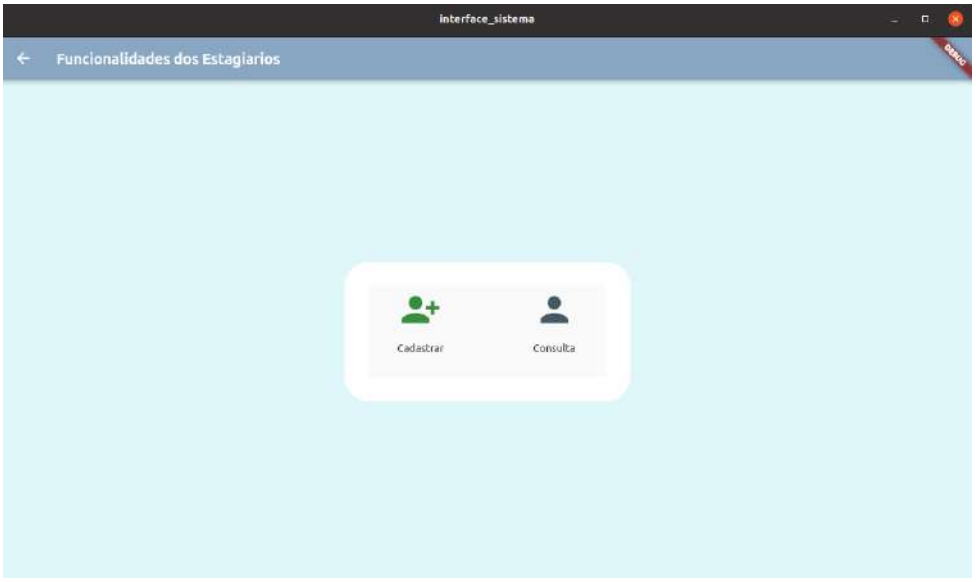
Fonte: Elaborado pelo autor (2024)

Figura A.2 – Listagem dos Estágios do Estagiário



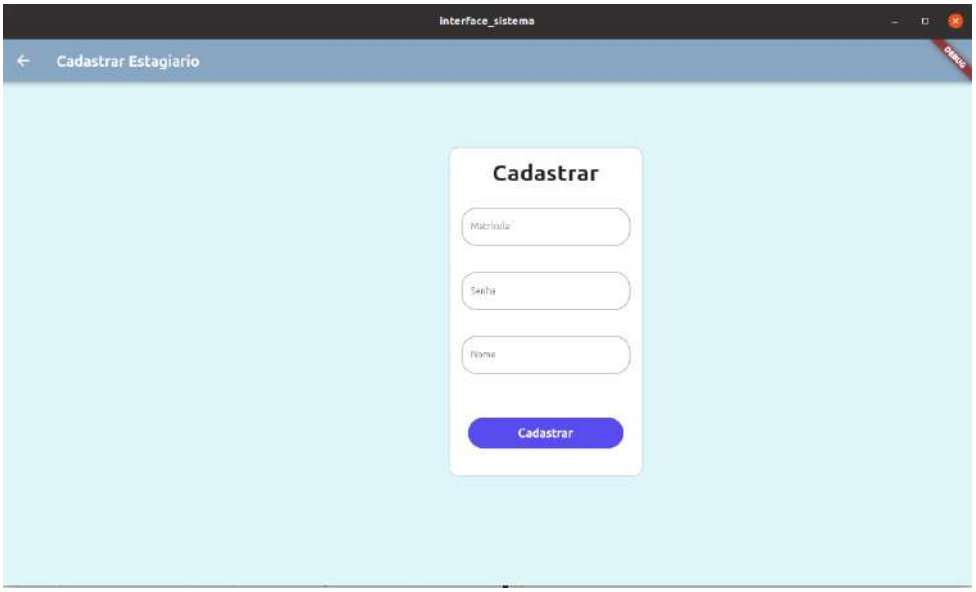
Fonte: Elaborado pelo autor (2024)

Figura A.3 – Tela de Funcionalidades de Estagiário



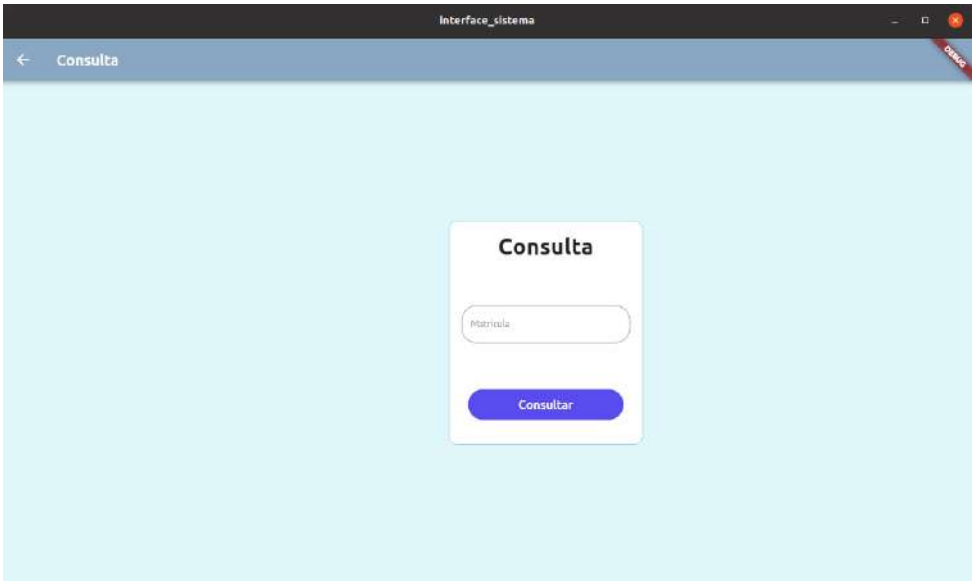
Fonte: Elaborado pelo autor (2024)

Figura A.4 – Tela de Cadastramento de Estagiário



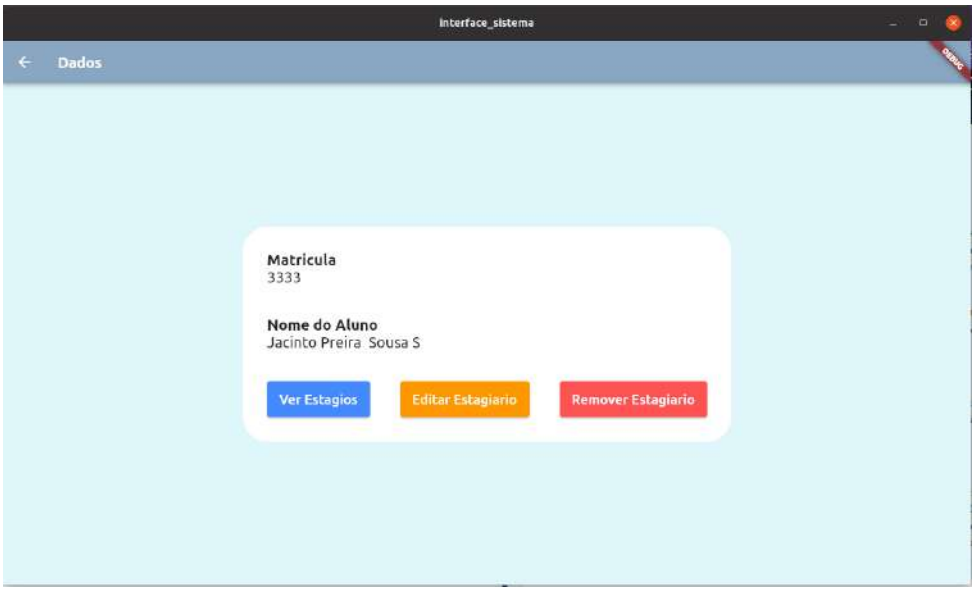
Fonte: Elaborado pelo autor (2024)

Figura A.5 – Tela de Consulta do Estagiário



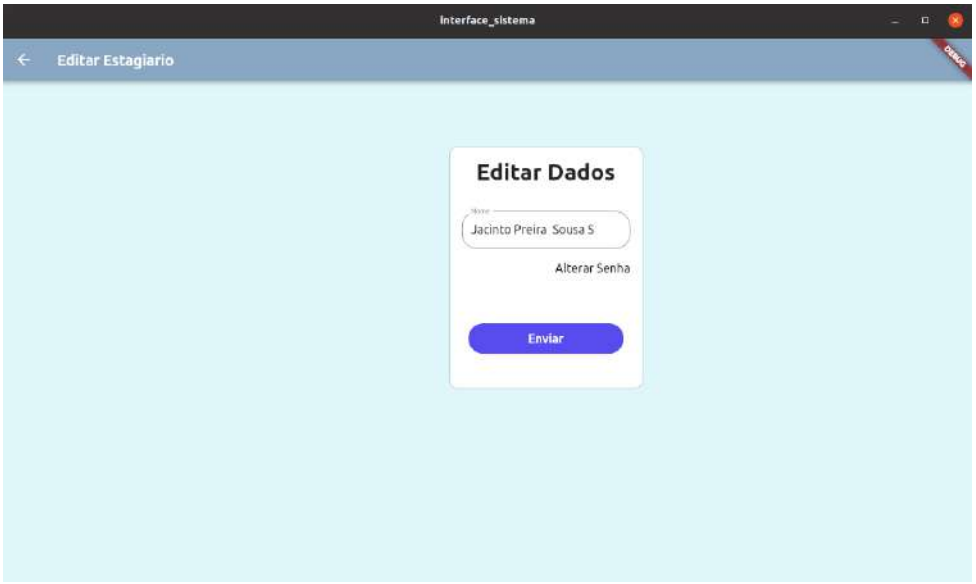
Fonte: Elaborado pelo autor (2024)

Figura A.6 – Tela de Dados com Operações do Estagiário



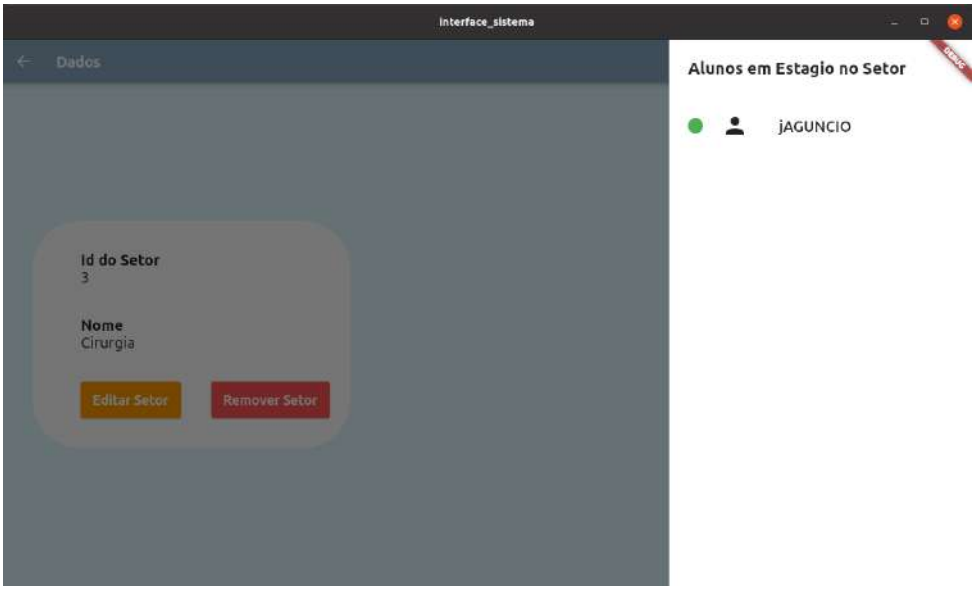
Fonte: Elaborado pelo autor (2024)

Figura A.7 – Tela de Edição de Dados do Estagiário



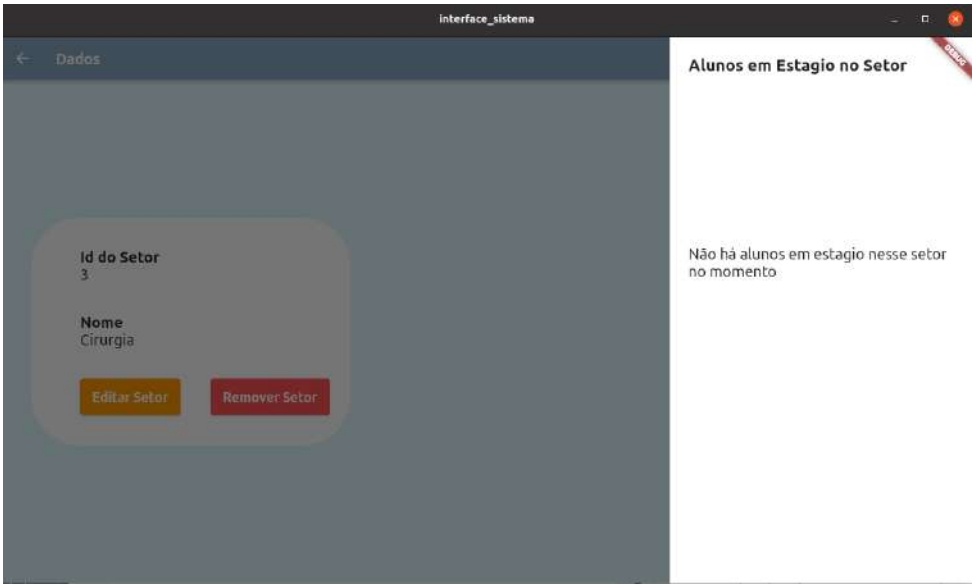
Fonte: Elaborado pelo autor (2024)

Figura A.8 – Tela de Acompanhamento com Estagiário em Estágio



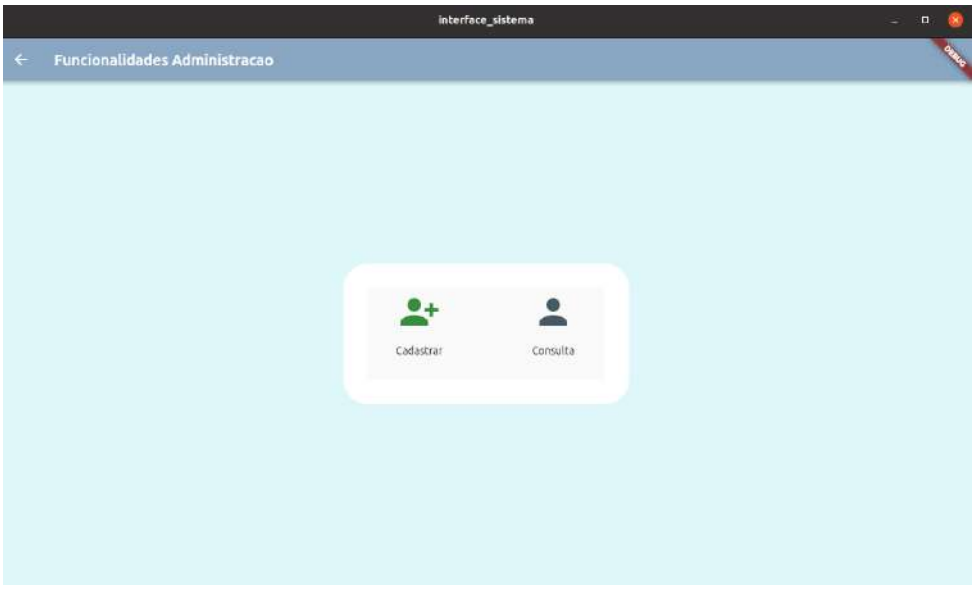
Fonte: Elaborado pelo autor (2024)

Figura A.9 – Tela de Acompanhamento sem Estagiário em Estágio



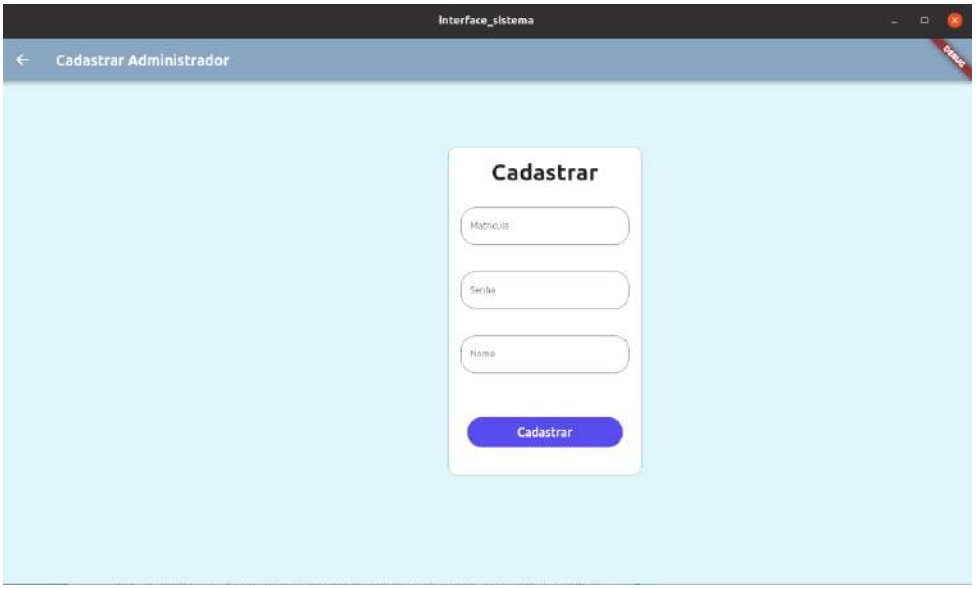
Fonte: Elaborado pelo autor (2024)

Figura A.10 – Tela de Funcionalidades do Administrador



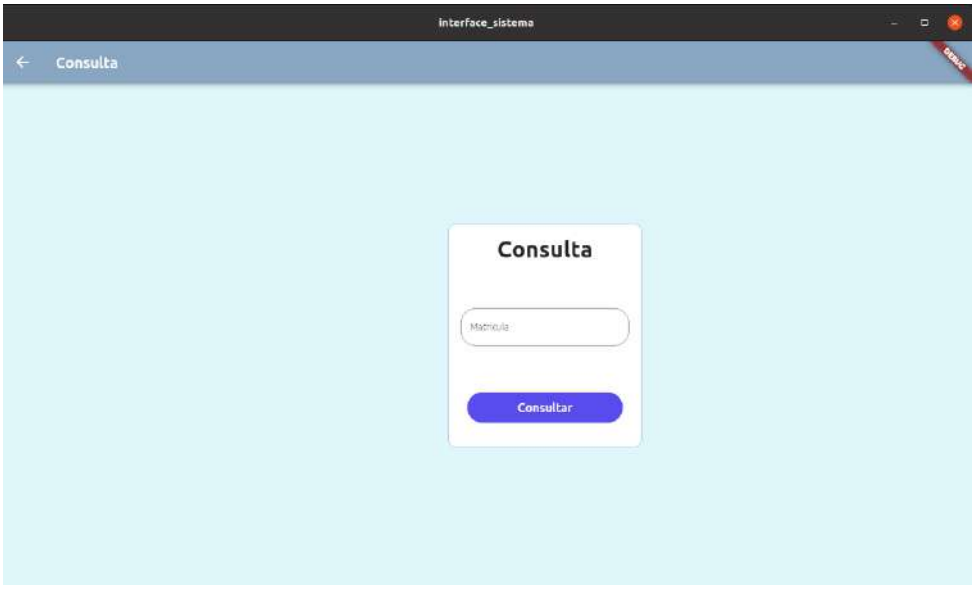
Fonte: Elaborado pelo autor (2024)

Figura A.11 – Tela de Cadastramento de Administrador



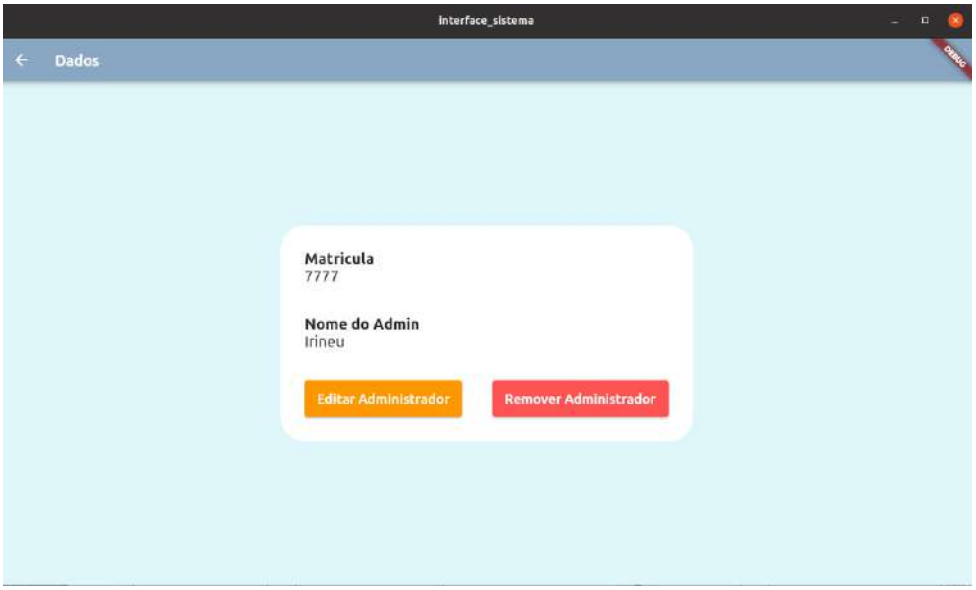
Fonte: Elaborado pelo autor (2024)

Figura A.12 – Tela de Consulta do Administrador



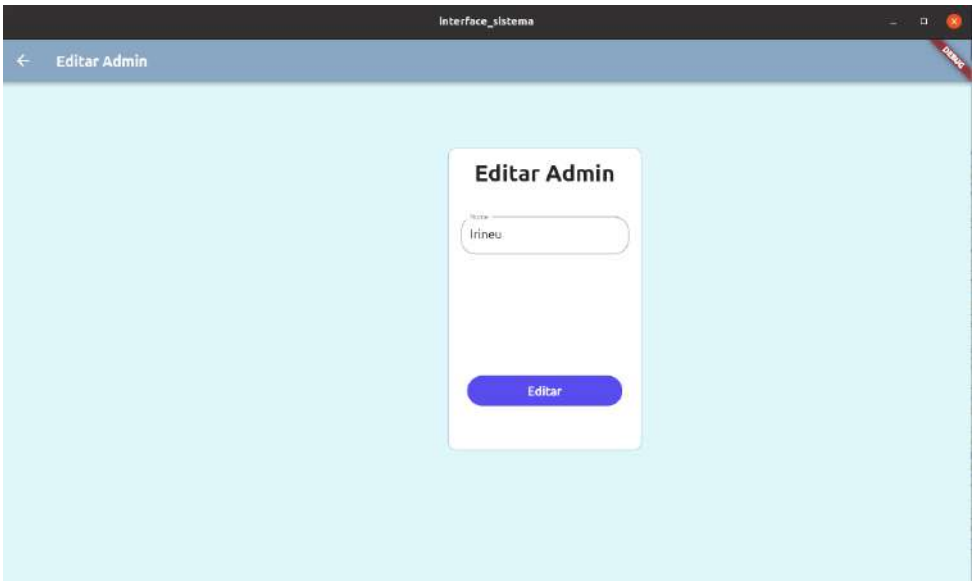
Fonte: Elaborado pelo autor (2024)

Figura A.13 – Tela de Dados de Administrador



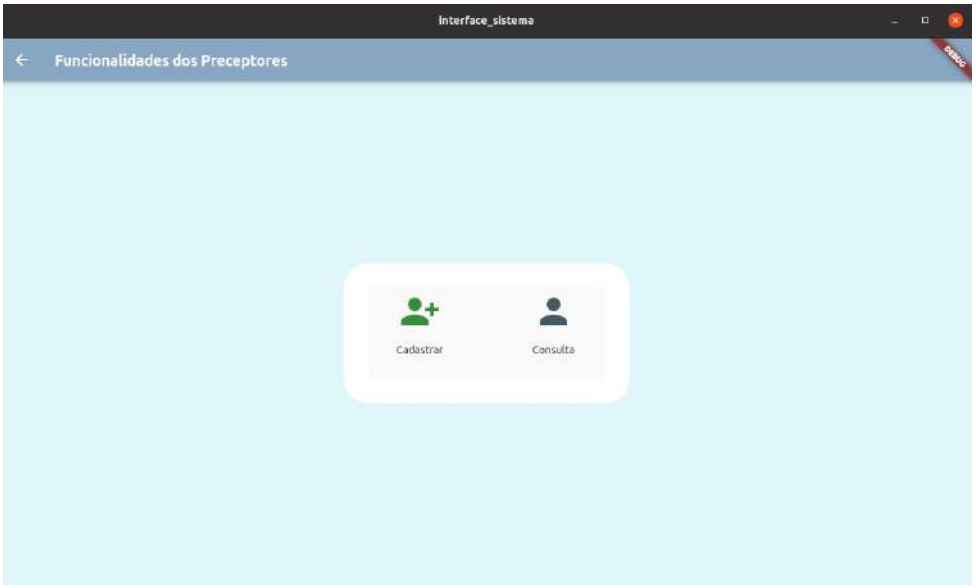
Fonte: Elaborado pelo autor (2024)

Figura A.14 – Tela de Edição do Administrador



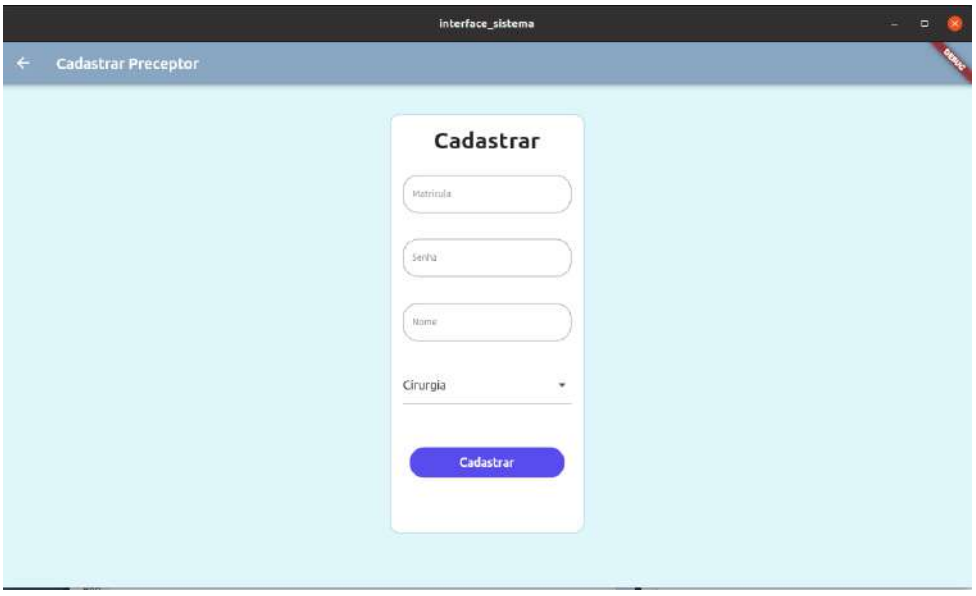
Fonte: Elaborado pelo autor (2024)

Figura A.15 – Tela de Funcionalidades do Preceptor



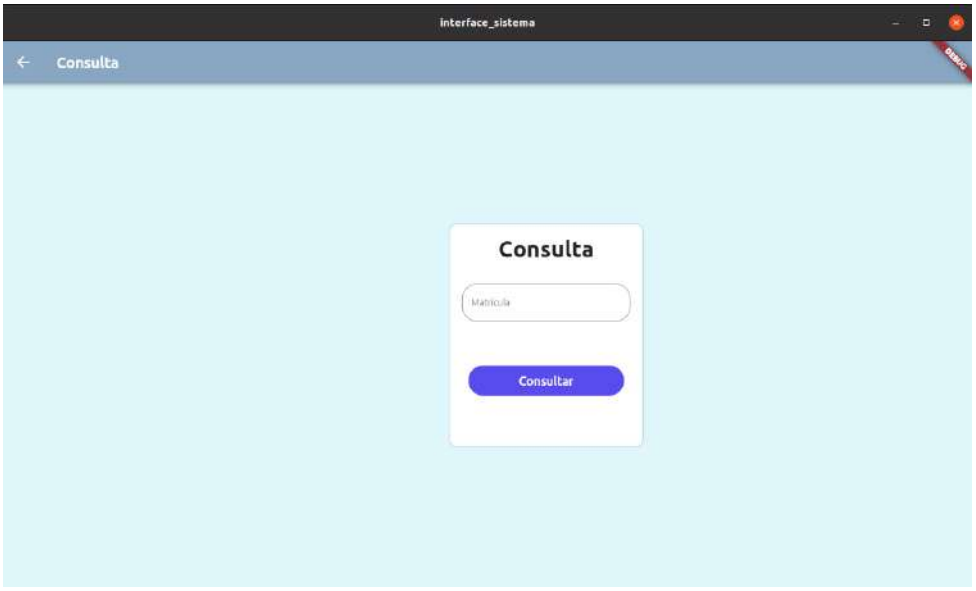
Fonte: Elaborado pelo autor (2024)

Figura A.16 – Tela de Cadastramento de Preceptor



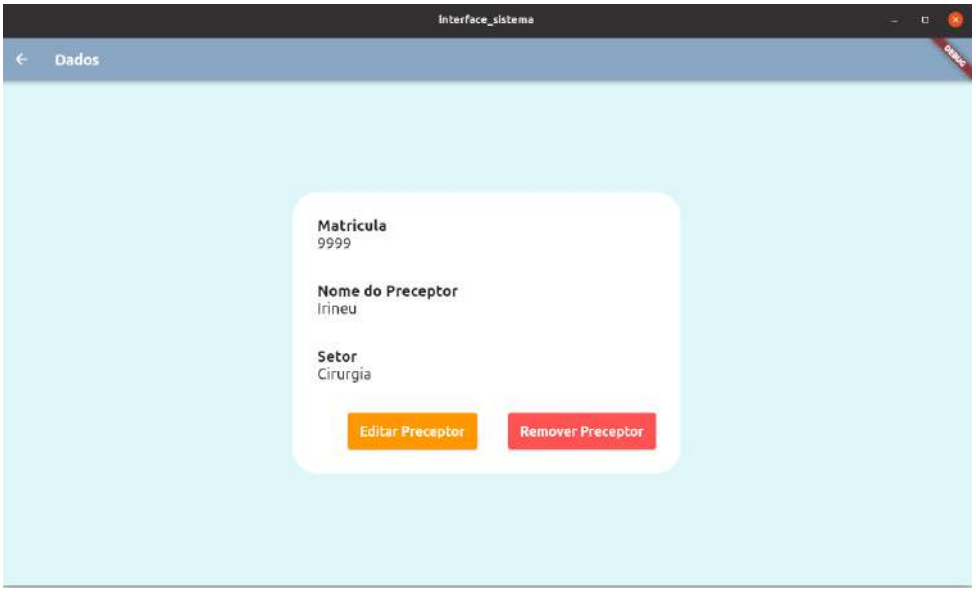
Fonte: Elaborado pelo autor (2024)

Figura A.17 – Tela de Consulta do Preceptor



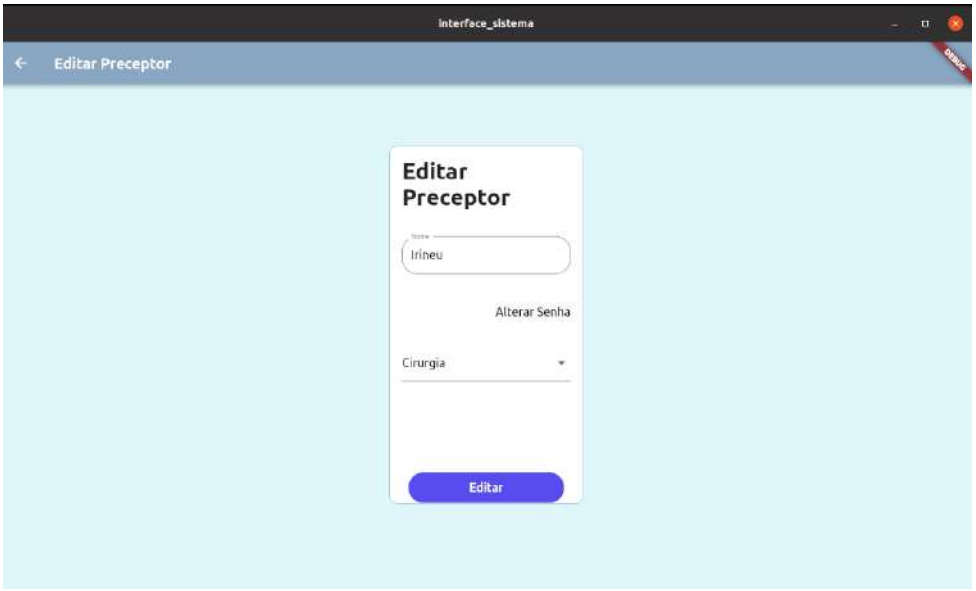
Fonte: Elaborado pelo autor (2024)

Figura A.18 – Tela de Dados de Preceptor



Fonte: Elaborado pelo autor (2024)

Figura A.19 – Tela de Edição do Preceptor



Fonte: Elaborado pelo autor (2024)

Figura A.20 – Tela de Inicial APP do Estagiário

Fonte: Elaborado pelo autor (2024)

Figura A.21 – QrCode disponibilizado pelo toten no setor de Cirurgia

Fonte: Elaborado pelo autor (2024)

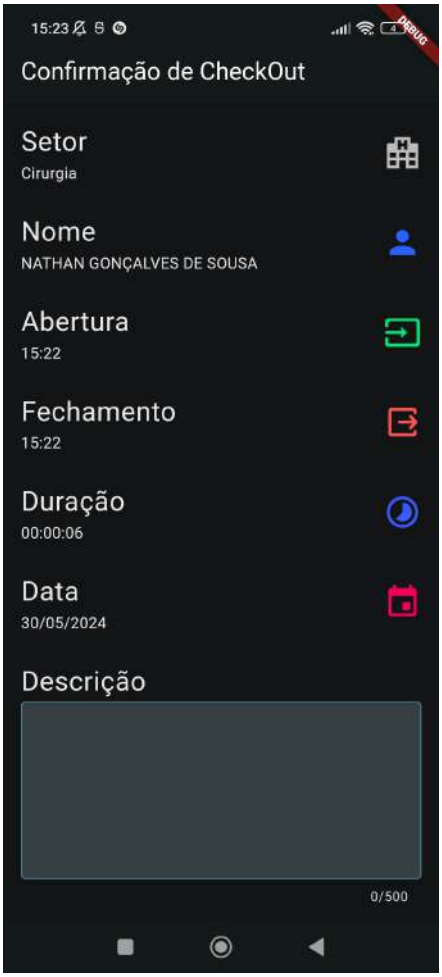
Figura A.22 – Tela do Scanner de QRCode do APP do Estagiário

Fonte: Elaborado pelo autor (2024)

Figura A.23 – Tela do Checkout APP do Estagiário

Fonte: Elaborado pelo autor (2024)

Figura A.24 – Tela de Conferencia de Dados do Estágio APP do Estagiário

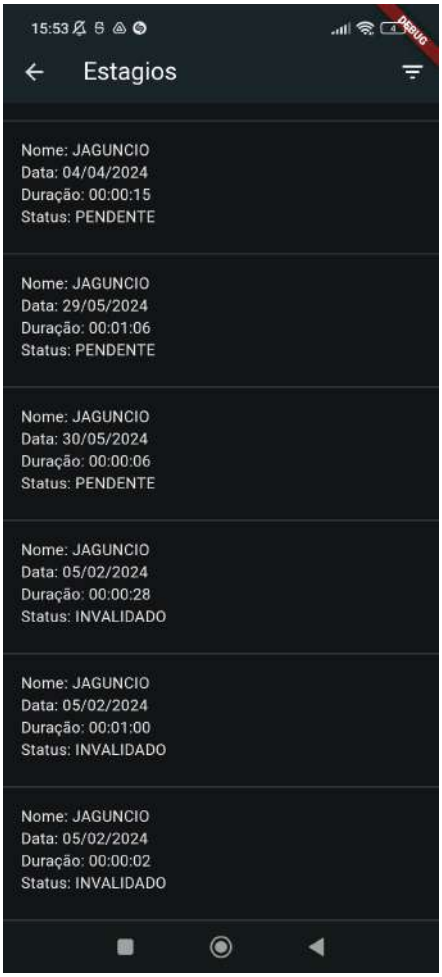


Fonte: Elaborado pelo autor (2024)

Figura A.25 – Tela de Estágio Finalizado Com Sucesso APP do Estagiário

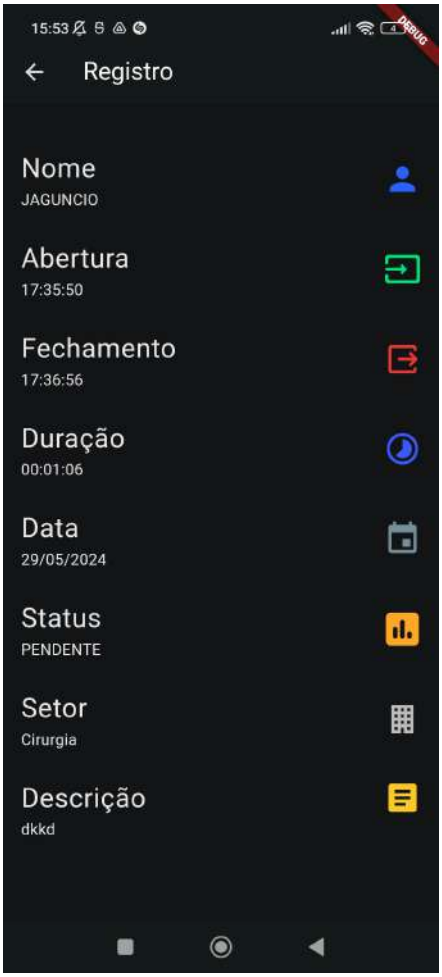
Fonte: Elaborado pelo autor (2024)

Figura A.26 – Listagem de Estágios APP Estagiário



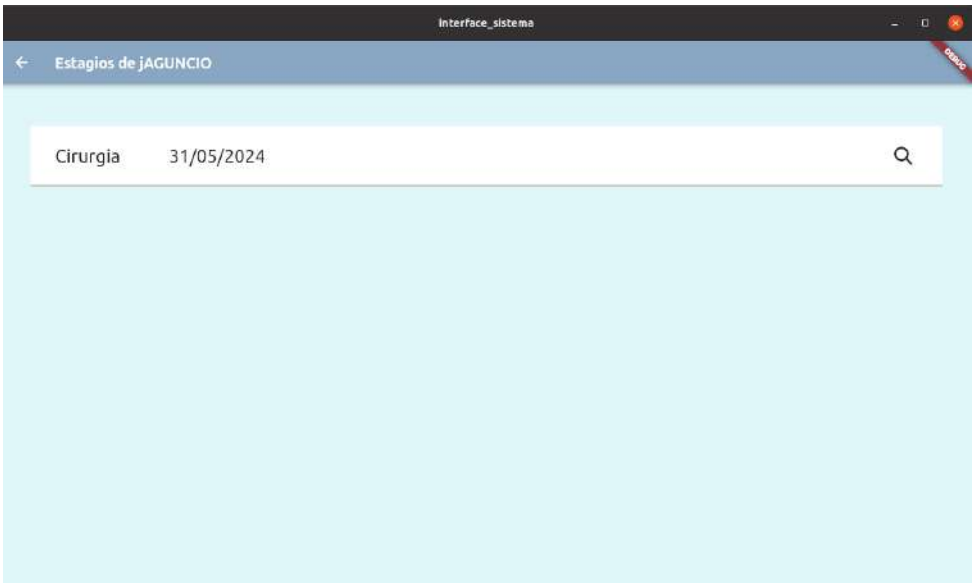
Fonte: Elaborado pelo autor (2024)

Figura A.27 – Detalhes do Estágio APP Estagiário



Fonte: Elaborado pelo autor (2024)

Figura A.28 – Listagem de Estágios em Análise APP Administração



Fonte: Elaborado pelo autor (2024)

Figura A.29 – Detalhe de Estágio em Análise APP Administração

Fonte: Elaborado pelo autor (2024)

Figura A.30 – Listagem de Estágios Assinados e Rejeitados APP Administração

Fonte: Elaborado pelo autor (2024)

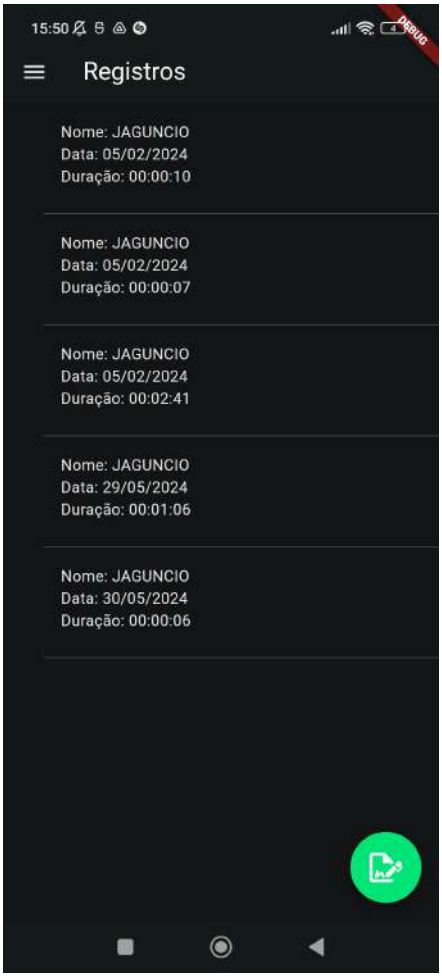
Figura A.31 – Detalhes de Estágio Assinado APP Administração

Fonte: Elaborado pelo autor (2024)

Figura A.32 – Detalhes de Estágio Rejeitado APP Administração

Fonte: Elaborado pelo autor (2024)

Figura A.33 – Lista de Estágios em Análise APP do Preceptor

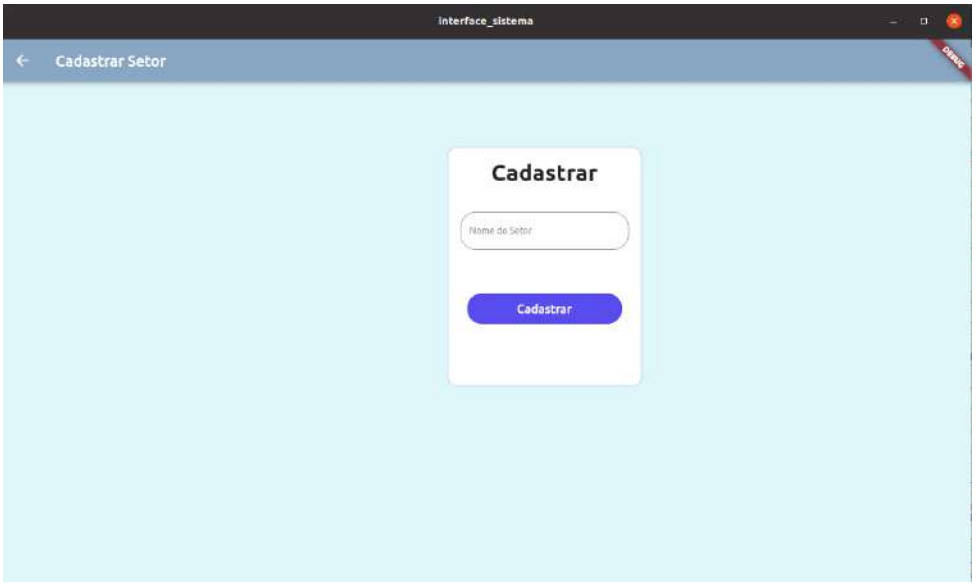


Fonte: Elaborado pelo autor (2024)

Figura A.34 – Assinatura de Estágios APP Preceptor

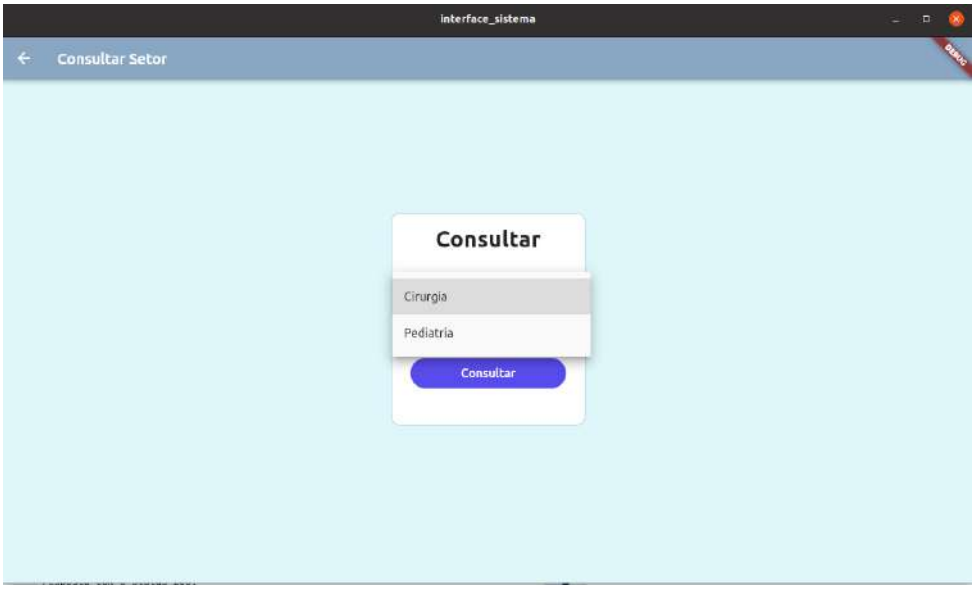
Fonte: Elaborado pelo autor (2024)

Figura A.35 – Tela de Cadastro de Setores APP Administração



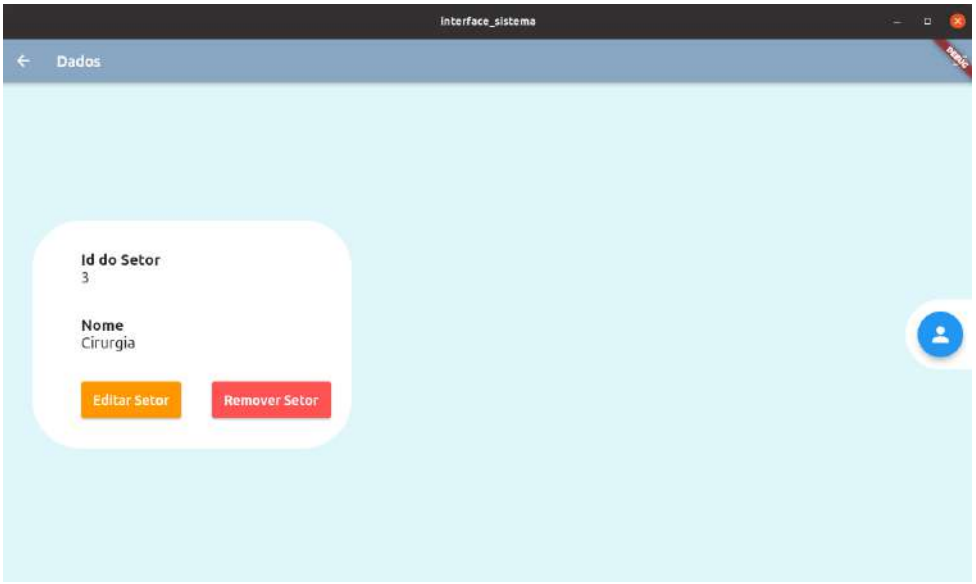
Fonte: Elaborado pelo autor (2024)

Figura A.36 – Tela de Consulta dos Setores APP Administração



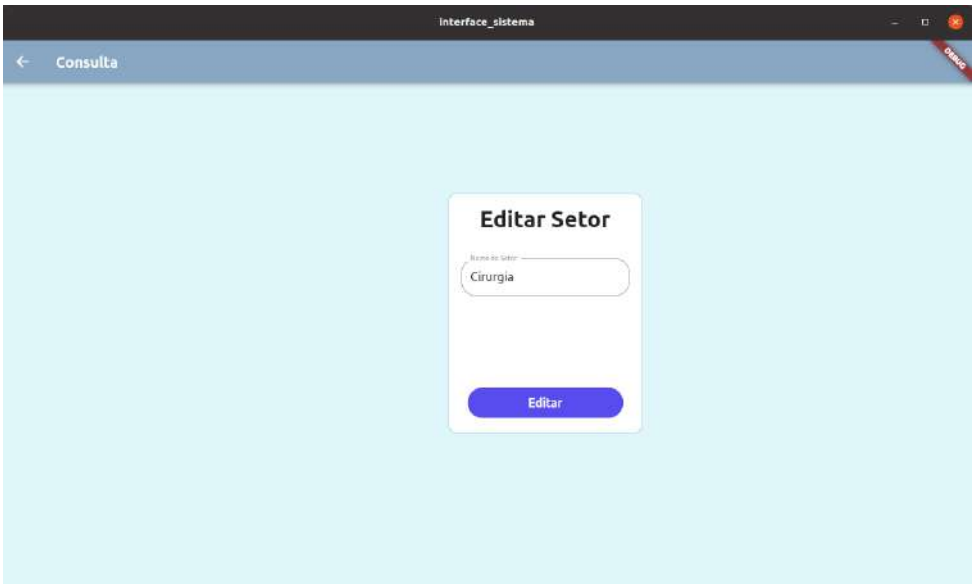
Fonte: Elaborado pelo autor (2024)

Figura A.37 – Tela de Dados dos Setores APP Administração



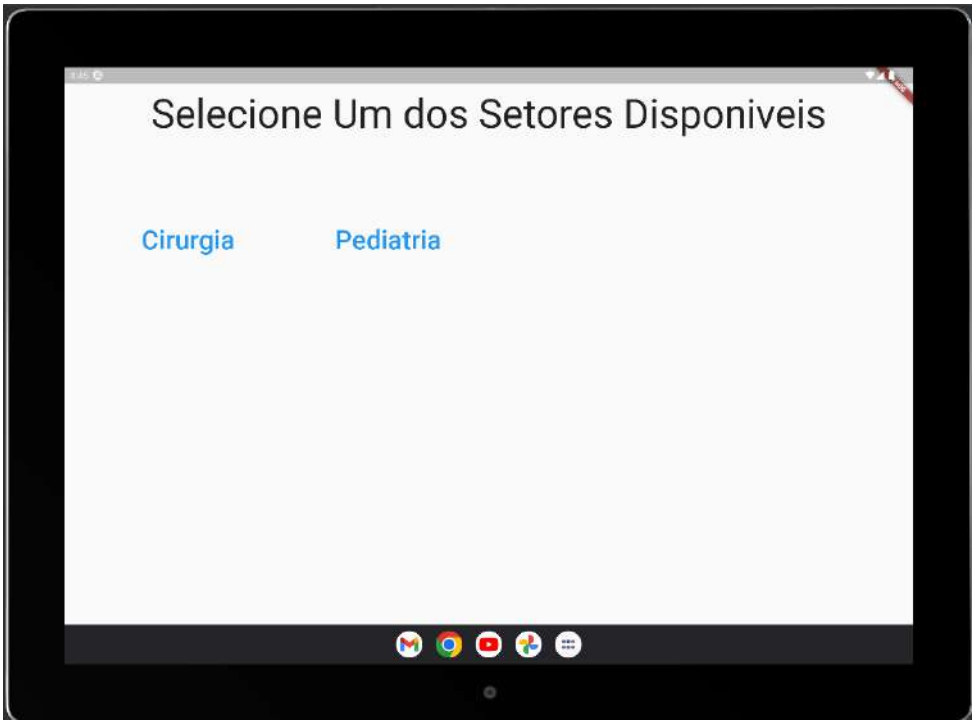
Fonte: Elaborado pelo autor (2024)

Figura A.38 – Tela de Edição de Setores APP Administração



Fonte: Elaborado pelo autor (2024)

Figura A.39 – Tela de Seleção de Setores para disponibilização de QRCode

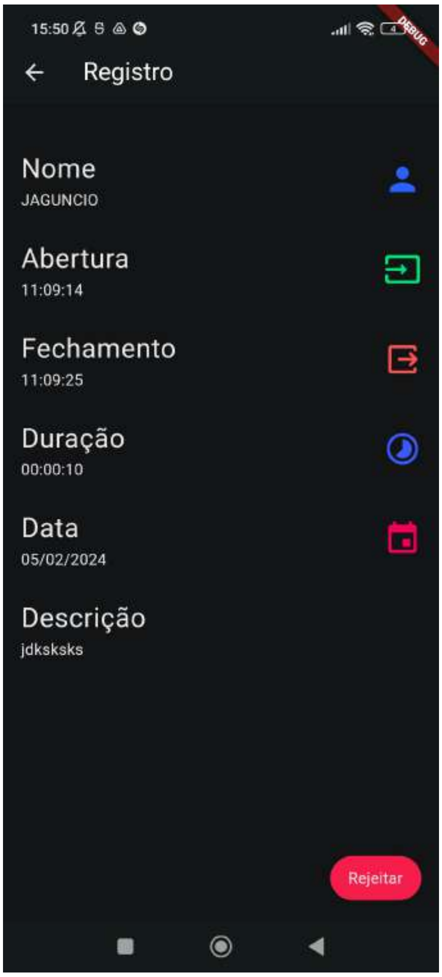


Fonte: Elaborado pelo autor (2024)

Figura A.40 – Autenticação para Mudança de QRCode

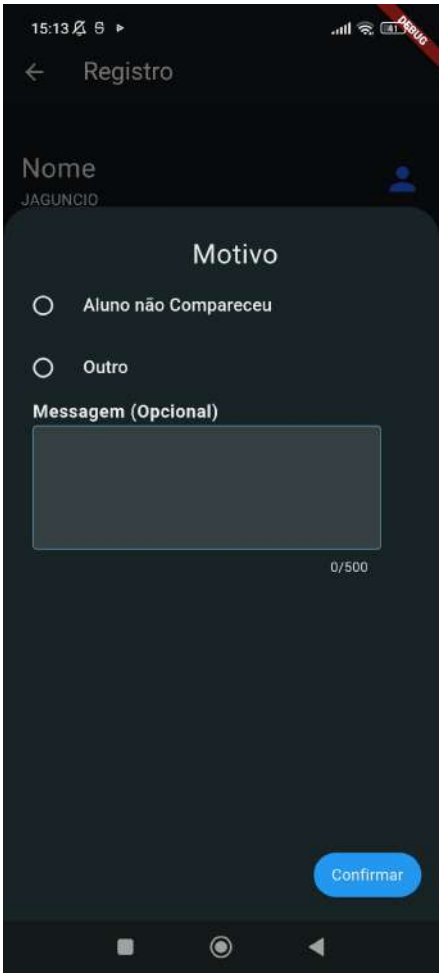
Fonte: Elaborado pelo autor (2024)

Figura A.41 – Detalhes de Estágio APP Preceptor



Fonte: Elaborado pelo autor (2024)

Figura A.42 – Rejeição de Estágio APP Preceptor



Fonte: Elaborado pelo autor (2024)

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Cajazeiras - Código INEP: 25008978
Rua José Antônio da Silva, 300, Jardim Oásis, CEP 58.900-000, Cajazeiras (PB)
CNPJ: 10.783.898/0005-07 - Telefone: (83) 3532-4100

Documento Digitalizado Ostensivo (Público)

TCC

Assunto:	TCC
Assinado por:	Nathan Sarmento
Tipo do Documento:	Projeto
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Nathan Pereira Sarmento**, **ALUNO (202012010013) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS**, em 30/08/2024 09:42:45.

Este documento foi armazenado no SUAP em 30/08/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1234473
Código de Autenticação: a1333a8fcc