UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE | CCENS

LEONARDO ALPORGES MARTINS

MEU LAB – APLICATIVO MÓVEL PARA GERENCIAMENTO E CONTROLE DOS LABORATÓRIOS DE INFORMÁTICA DA UFES CAMPUS DE ALEGRE

Alegre
2023

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE | CCENS

MEU LAB – APLICATIVO MÓVEL PARA GERENCIAMENTO E CONTROLE DOS LABORATÓRIOS DE INFORMÁTICA DA UFES CAMPUS DE ALEGRE

Trabalho de conclusão de curso apresentado à Universidade Federal do Espírito Santo, como parte das exigências para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Giuliano Prado de Morais Giglio

Alegre

2023

LEONARDO ALPORGES MARTINS

MEU LAB – APLICATIVO MÓVEL PARA GERENCIAMENTO E CONTROLE DOS LABORATÓRIOS DE INFORMÁTICA DA UFES CAMPUS DE ALEGRE

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovado em 17 de julho de 2023.

COMISSÃO EXAMINADORA

Prof. M.Sc. Giuliano Prado de Morais Giglio
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Helder de Amorim Mendes
Universidade Federal do Espírito Santo

Prof. Dr. Marcelo Otone Aguiar
Universidade Federal do Espírito Santo

AGRADECIMENTOS

Em primeiro lugar, a Deus, que fez com que meus objetivos fossem alcançados, durante todos os meus anos de estudos. Aos meus pais que sempre lutaram para me dar a melhor educação, sempre apoiaram em minhas escolhas e nunca me deixaram faltar nada para ter condições de ter chegado até aqui, e hoje poder retribuir isso para eles com orgulho. Aos meus irmão que além de irmão são os melhores amigos que eu poderia ter, e por deixar sempre ativo em mim o sentimento de irmão mais velho e exemplo para eles, isso fez com que não desistisse em momentos difíceis da graduação. A Joana por ter aberto meus olhos para entrar na faculdade e estar sempre me ajudando com meus projetos e meus sonhos e apoiando durante toda graduação. Ao meu orientado, professor Giuliano por ter me alocado e orientado em projetos de iniciação cientifica onde esse foram fundamentais para minha progressão na carreira, além de ter me orientado em todos os aspectos para a conclusão deste trabalho e aos meus amigos de turma por sempre estarem disponíveis nos fins de semana para comer churrasco e jogar conversa fora.

RESUMO

Ao adentrar em uma universidade, um dos desafios dos ingressantes é saber se localizar, sendo a instituição composta de inúmeros setores, departamentos, unidades, dentre outros. Além disso, o ingressante ainda precisa utilizar os serviços que o campus disponibiliza e, mesmo após a conclusão do curso, ainda pode persistir o desconhecimento de alguns setores e o funcionamento de alguns serviços. Isso se dá pela falta de um sistema centralizado com todas as informações e a forma como essas informações são disponibilizadas e controladas. Em muitos casos são ainda utilizados meios manuais de anotação para controles de acesso a espaços do campus. Desta forma, um dos objetivos desse projeto é realizar a modernização em um desses setores que é a Gestão dos Laboratórios de Informática do Departamento de Computação, possibilitando agilizar a coordenação das diversas salas para este fim, e disponibilizar todas as informações sobre a utilização e funcionamento destes. Com a utilização desse sistema, espera-se que alunos, professores e coordenadores dos laboratórios possam ter um controle mais ágil e centralizado do acesso às informações essenciais e em tempo real, de maneira simplificada e segura. Pretende-se, com a implantação de um sistema móvel como proposto nesse projeto, ampliar as demais iniciativas do campus visando implantar a visão de Campus Inteligente no campus de Alegre, ES da UFES, melhorando o acesso à informação de toda comunidade acadêmica.

Palavra-chave: campus inteligente; cidade inteligente; sistema móvel; internet das coisas.

LISTA DE FIGURAS

Figura 1 - Ilustração de uma cidade inteligente. 17
Figura 2 - Imagem da câmera de segurança da porta do RU da UNICAMP. 20
Figura 3 - Sistema de administração do U²Pass 21
Figura 4 - Ranking de tecnologias mais populares em 2022 24
Figura 5 - Fluxo de versionamento de código usando a estrutura git workflow. 25
Figura 6 - Arquitetura MVC 27
Figura 7 - Teste de caixa preta. 28
Figura 8 - Estrutura do desenvolvimento incremental. 38
Figura 9 - Diagrama de caso do sistema para usuários. 44
Figura 10 - Diagrama de caso de uso do sistema para coordenador. 45
Figura 11 - Diagrama de pacotes do sistema. 48
Figura 12 - Diagrama de componente do sistema. 48
Figura 13 - Modelo físico do banco de dados. 50
Figura 14 - Esquema de pasta do projeto. 53
Figura 15 - Estrutura da routes para monitor 53
Figura 16 - Estrutura da função getMonitor 54
Figura 17 - Estrutura do arquivo de configuração knex 55
Figura 18 - Estrutura da função bussinesExceptionValidation 55
Figura 19 - Estrutura de pasta do front end. 56
Figura 20 - Estrutura da pasta assets. 57
Figura 21 - Estrutura da pasta Components; 58
Figura 22 - Função de configuração do acesso à ferramenta Firebase. 58
Figura 23 - Funções de autenticação no aplicativo 59
Figura 24 - Arquivo de variáveis globais do aplicativo relacionada a estilos 60
Figura 25 - Hooks de ocorrência onde são definidas as funções usadas em tela. 61
Figura 26 - Estrutura de pasta dos hooks. 62
Figura 27 - Estrutura da tela Home separada em componentes. 62
Figura 28 - Estrutura das Routes para usuário autenticados. 63
Figura 29 - Definições das api’s para a services de reserva do aplicativo. 64
Figura 30 - Estrutura da pasta service. 65
Figura 31 - Variável utilitária para definição de itens em um selectBox. 65
Figura 32 - Tela inicial do aplicativo após usuário logado. 67

Figura 33 - Tela de visualização dos horários dos monitores.....67
Figura 34 - Registro de teste do sistema como coordenador.....71
Figura 35 - Formulário sobre usabilidade do sistema.....72
Figura 36 - Gráfico de respostas de alunos de computação sobre o login.....73
Figura 37 - Gráfico de respostas de alunos de computação sobre acesso às informações de laboratório. 73
Figura 38 - Gráfico de respostas de alunos de outros curso sobre o login 74
Figura 39 - Gráfico de respostas de alunos de outros cursos sobre acesso a informações de laboratório. 74
Figura 40 - Gráfico representando respostas de alunos que são ou já foram monitores dos laboratórios, sobre o login.....75
Figura 41 - Gráfico representando respostas de alunos que são ou já foram monitores dos laboratórios, sobre acesso a informações de laboratório.....75
Figura 42 - Codigo que mostra como é feita atribuição de número de matrícula a um usuário.....76

LISTA DE TABELAS

Tabela 1 - Definição de campus inteligente segundo Ferreira e Araújo. 18
Tabela 2 - Tabela de occurrence.....51
Tabela 3 - Descrição dos módulos da estrutura do projeto 53
Tabela 4 - Backlog de tarefas iniciais do sistema.....68
Tabela 5 - Públicos alvos dos testes de usabilidade. 69
Tabela 6 - Roteiro de teste funcional do sistema 70

SUMÁRIO

1.Introdução 12
1.1 O problema e sua importância 14
2. Objetivos 15
2.1 Objetivo geral 15
2.2 Objetivos específicos 15
3. Revisão da literatura 16
3.1 Sistema de informação integrada aos campus universitários 16
3.1.1 Cidade e campus inteligente 17
3.2 Tecnologias empregadas 22
3.2.1 Linguagem de programação 22
3.2.2 Frameworks 22
3.2.2.1 React-Native e Node JS 23
3.2.3 SGBD 24
3.2.4 Controle de versão 24
3.2.5 Ferramentas CASE 25
3.2.6 Arquitetura do sistema 26
3.2.7 Teste de sistema 27
3.2.8 Integração entre as tecnologias 29
3.3 Engenharia de Software 29
3.3.1 Desenvolvimento incremental 30
3.3.2 Levantamento preliminar de requisitos 30
3.3.3 Diagrama de caso de uso 31
3.3.4 Narrativa caso de uso 31
3.3.5 Modelagem de domínio 31
3.3.6 Diagrama de pacotes e componentes 32
3.4 Banco de dados 32
3.4.1 Modelagem de dados 32
3.4.2 Modelo lógico 33
3.4.3 Modelo físico 33
3.5 Backend 33
3.5.1 Routes 34
3.5.2 Controller 34
3.5.3 Útil 34

3.6 Frontend 34
3.6.1 Components 35
3.6.2 Config 35
3.6.3 Context 35
3.6.4 Global 36
3.6.5 Hooks 36
3.6.6 Routes 36
3.6.7 Utils 36
4. Metodologia 37
4.1 Desenvolvimento incremental 38
4.2 Levantamento preliminar de requisitos 39
4.2.1 Requisitos funcionais 39
4.2.2 Requisitos não funcionais 43
4.3 Diagrama de caso de uso 43
4.4 Narrativa caso de uso 45
4.5 Modelagem de domínio 47
4.5.1 Diagrama de pacotes 47
4.5.2 Diagrama de componente 48
4.6 Modelo de banco de dados 49
4.6.1 Modelo Lógico 49
4.6.2 Modelo Físico 49
4.7 Descrição geral do banco de dados 50
4.8 Garantia da qualidade e testes de validação do sistema 51
5. Desenvolvimento da aplicação 52
5.1 Backend 52
5.1.1 Estrutura do projeto 52
5.1.2 Routes 53
5.1.3 Controller 54
5.1.4 Conexão com banco de dados 54
5.1.5 Útil 55
5.2 Frontend 56
5.2.1 Estrutura do frontend 56
5.2.2 Assets 57
5.2.3 Components 57
5.2.4 Config 58
5.2.5 Context 59

5.2.6 Global 59
5.2.7 Hooks 60
5.2.8 Pages 62
5.2.9 Routes 63
5.2.10 Services 64
5.2.11 Utils 65
5.2 Git hub 65
5.3 Banco de dados 66
5.4 Interface 66
5.5 Ciclo de trabalho - Implementações 67
6. Garantia da qualidade e processo de teste 68
6.1 Testes funcionais 69
6.2 Testes de sistema 71
6.3 Testes de usabilidade 72
6.4 Análise dos testes 75
7. Considerações finais 76
7.1 Melhorias e trabalhos futuros 77
7.1.1 Informativos gerais do campus 78
7.1.2 Integrações com outros sistemas 78
7.1.3 Disponibilização para sistemas operacionais IOS 78
REFERÊNCIAS BIBLIOGRÁFICAS 79
APÊNDICE A – Narrativas de caso de uso 83
APÊNDICE B – Descrição das tabelas do modelo conceitual do banco de dados 105
APÊNDICE C – Modelo físico de banco de dados em português 110
APÊNDICE D - Tarefas elaboradas para o desenvolvimento do sistema. 111
APÊNDICE E - Perguntas para teste de usabilidade. 113
APÊNDICE F - Interfaces do sistema 130

1. Introdução

Atualmente é possível perceber que a tecnologia está presente em muitos lugares, o simples ato de consultar o horário do ônibus no celular ou fazer um café na cafeteira é um exemplo da tecnologia no dia a dia das pessoas. Para aproveitar o máximo da tecnologia, são criados a todo instante *softwares* e *hardwares* que auxiliam a sociedade, seja no trabalho, nos estudos ou no lazer. Todos esses instrumentos possuem o intuito de proporcionar uma melhor qualidade de vida para quem os utiliza. Segundo os dados da International Data Corporation (IDC), em 2021 foram investidos cerca de 2,79 trilhões de dólares em tecnologia da informação a nível global. Os computadores, *smartphones, notebook e tablets*, quando somados, já ultrapassam a marca de 447 milhões de unidades, sendo mais de duas unidades por habitantes no Brasil (MEIRELLES, 2020).

No meio acadêmico e científico, sobretudo nas Instituições de Ensino Superior (IES), é comum observar que a grande maioria das pessoas possuem aparelhos celulares que utilizam para se comunicar, navegar na internet ou até mesmo auxiliar em aulas. Segundo pesquisa realizada pela Passei Direto, cerca de 82% dos alunos utilizam os aparelhos celulares para armazenar os materiais de aula (SODRÉ, 2014). O celular é uma ferramenta rápida e eficiente, além de ser fundamental para trocas de informações ou dúvidas entre alunos e professores, seja por correio eletrônico (e-mail) ou por aplicativos de mensagens (SODRÉ, 2014).

A pandemia, anunciada pela Organização Mundial da Saúde em março de 2020, foi um problema global que causou impactos em vários setores por todo o mundo. Dentro da área da tecnologia, intensificou o uso dos sistemas de informação em geral, sobretudo no Brasil, passando de 71% das casas com internet em 2019 para 83% no ano de 2021 (NITAHARA, 2021). Esse aumento se fez necessário para que houvesse a continuidade das atividades do comércio com a venda online, do setor empresarial com o modelo de serviço *home office*, do ensino remoto e da prestação de serviços públicos (NITAHARA,2021). Segundo Oliveira e Chalub (2021), o número

de brasileiros que utilizam os serviços digitais do governo federal subiu de 1,7 milhão, em janeiro de 2019, para 113 milhões neste ano.

Para acompanhar esse grande avanço tecnológico na sociedade e no meio acadêmico, o termo “campus inteligente” vem sendo bastante estudado e implementado nas universidades, que integra um estudo ainda maior chamado de “Cidades Inteligentes”. Segundo Neves et al. (2017), ainda não existe uma definição geral para esses termos, porém a maioria das definições existentes se baseiam no uso de Tecnologias da Informação e Comunicação (TICs) para prover serviços básicos à sociedade e uma melhor qualidade de vida.

A característica principal de campus inteligente é a rápida adaptação do ambiente em relação às demandas, que podem ter diversas origens e contextos diferentes (JACOSKI, LISSANDRO, 2019, p. 4, apud FERREIRA, ARAUJO 2018*)*. Trazer esse conceito para dentro das universidades promove o aprendizado e o incentivo para serem criados mais projetos utilizando-o e assim promovendo um desenvolvimento inteligente e sustentável do campus (JACOSKI, et al., 2019).

Esse termo também pode ser encontrado como Universidade Inteligente, porém como citado anteriormente, “a falta desse consenso resulta em diferentes abordagens para o mesmo termo.” (FERREIRA, ARAUJO, 2018, p. 5). “Uma Cidade Inteligente vai além do simples uso de tecnologias. É uma visão sistêmica e holística que inclui a sociedade no processo de tomada de decisão e promove a integração de diversos setores.” (NEVES, et al., 2017).

Para acompanhar o atual cenário dos avanços da tecnologia no meio acadêmico é preciso que sejam feitas alterações em como são gerenciados os processos internos e nas distribuições de informações destes, trazendo uma informatização para esses processos de tal forma que a instituição possa crescer e se tornar referência em uso da tecnologia para gerenciar o campus e promover uma interação mais ativa com seu corpo acadêmico.

1.1 O problema e sua importância

Atualmente na Universidade Federal do Espírito Santo (UFES) são disponibilizados recursos que facilitam os estudos dos alunos, como por exemplo os laboratórios de informática com computadores, entretanto, esse recurso pode ficar inacessível para alguns alunos devido às dificuldades de acesso a informações como, horários de funcionamento; disponibilidade de monitor para auxiliar no uso do laboratório, lotação, entre outros. Essas dificuldades refletem também em outros setores dentro da universidade e cada um deles lida com o problema de uma forma diferente. Geralmente, essas informações são disponibilizadas em sites ou por meio de impressões, afixadas em murais da instituição.

Os alunos que dependem dos laboratórios do departamento de computação, precisam buscar no site da Coordenação de Laboratórios de Informática do campus Alegre a disponibilidade do laboratório, porém, essa informação muitas vezes não está de acordo com a atual realidade do laboratório. Podem, por exemplo, ocorrer imprevistos, até mesmo pela falta de manutenção da informação dessa plataforma. Além disso assim, o gerenciamento desses laboratórios não conta com um sistema de notificação em tempo real, avisando, por exemplo, se o laboratório estará disponível para uso. Desta forma, o aluno precisará procurar o coordenador do laboratório para validar as informações de funcionamento.

Para utilizar os laboratórios disponíveis do departamento de computação, é necessário que o aluno tenha conhecimento do horário de monitoria e que, no local, tenha um monitor responsável no horário da utilização. Mesmo sendo disponibilizados os horários das chamadas monitorias, pode-se ocorrer algum problema ou imprevisto, dessa forma, impossibilitando a utilização do laboratório. Quando isso ocorre, não é possível comunicar a todos os alunos interessados, de forma ágil e sistemática.

Avaliando o gerenciamento atual dos laboratórios da UFES, encontra-se uma dificuldade na hora de organizar e designar os monitores para horários de maior fluxo de alunos nos laboratórios, ou para quando ocorre a alteração de horários de um dos monitores, uma vez que tudo isso é feito por planilha. Quando algum horário é alterado, não há notificação em tempo real, somente aviso por e-mail ou avisos no local afixados pelo Coordenador.

Quando é preciso agendar um dos laboratórios, há todo um processo a ser executado, podendo ser um processo demorado, pois a solicitação é feita via formulário e, para que seja efetivada de fato a reserva, é necessário que o coordenador esteja atento ao seu e-mail institucional, o que pode não acontecer devido a sua disponibilidade ou há um alto fluxo de e-mails recebidos.

2. Objetivos

Nas seções abaixo serão apresentados o objetivo geral e os objetivos específicos.

2.1 Objetivo geral

Objetivou-se desenvolver um sistema mobile para auxiliar pessoas ligadas ao campus acadêmico a obter as informações referentes aos laboratórios em único lugar, facilitando assim o acesso. Além disso, disponibilizar recursos para facilitar o gerenciamento dos laboratórios que é feito pelo coordenador, permitindo que, alterações e suportes necessários possam ser rapidamente executados sem a necessidade de um *desktop*.

2.2 Objetivos específicos

Ter uma aplicação móvel compatível com aparelhos com sistema operacional Android que disponibilize informações sobre os laboratórios de informática de maneira simplificada; Notificar o usuário em tempo real sobre o status de solicitação de suporte, andamento de ocorrência, alteração ou adição de horários de monitoria e notícias gerais sobre os laboratórios; Disponibilizar de maneira centralizada, ferramentas para que o coordenador do laboratório possa ter maior controle sobre eles, podendo emitir notificações em tempo real sobre o funcionamento dos laboratórios; Elaborar um serviço de API onde serão tratadas todas as informações referentes a ferramentas

disponibilizadas no aplicativo; Desenvolver a interface do sistema utilizando react-native; Disponibilizar a aplicação para usuários através da google play; Hospedar o serviço de API nos servidores da universidade; Elaborar documentações para o usuário final sobre como utilizar o sistema de ponta a ponta e Elaborar documentação sobre autenticação com credenciais da universidade.

3. Revisão da literatura

Para este projeto será apresentado nas seções abaixo o conceito teórico de sistemas de informação aplicado ao ambiente de um campus universitário, juntamente com a apresentação de diversas tecnologias e ferramentas que foram utilizadas para o desenvolvimento desse projeto. Além disso, será explorado sobre como essas tecnologias e ferramentas se comportam quando utilizadas em conjunto.

3.1 Sistema de informação integrada aos campus universitários

Um sistema, seja ele qual for, é constituído de duas partes, lógica e física, onde temos uma coleção de dados de um lado e a ligação lógica entre eles do outro, e isso dá origem a um organismo (MATTOS, 2017). O conceito de sistema pode ser aplicado a diversos contextos, incluindo organizações. De acordo com Claro (2013), uma organização pode ser entendida como um sistema formado por diversas partes ou elementos que se integram e trabalham em conjunto para atingir um objetivo comum.

Para Ferreira e Araújo (2018), um ambiente inteligente é aquele que é capaz de adquirir e aplicar conhecimento sobre o espaço e seus habitantes a fim de melhorar sua experiência nesse ambiente. Seguindo esse raciocínio surgiu o conceito de cidade inteligente que visa integrar as informações entre os serviços à população, enriquecendo o espaço urbano e oferecendo melhor qualidade de vida para os cidadãos. E das cidades inteligentes surgiram os campi inteligentes que se utilizam das ideias da cidade inteligente para implementar dentro do campus universitário soluções que melhorem o dia a dia do estudante (FERREIRA, ARAUJO, 2018).

3.1.1 Cidade e campus inteligente

O alto crescimento do fluxo de pessoas em direção às grandes cidades tem trazido grandes desafios para os governantes. Tais desafios estão diretamente ligados a prestação de serviços básicos e essenciais para essa população. Ainda mais desafiadora é a situação quando se observam as restrições legais e econômicas que cercam o financiamento do desenvolvimento por meio do dinheiro dos impostos, bem como a competição entre países, cidades e metrópoles por uma atração de investimentos mais agressiva (BERNARDES et al., 2017). De modo geral, em maior ou menor número as cidades estão se adequando a esse crescimento buscando meios rápidos e eficientes de atender a toda essa população com a implementação das “cidades inteligentes”.

Algumas cidades optaram pelas tecnologias *smart* como uma solução para resolver problemas específicos de grandes centros urbanos como congestionamento, falta de emprego, alagamentos, coleta de lixo dentre outros. Uma cidade inteligente combina tecnologia inteligente com sistemas de serviços públicos, transporte, educação, saúde e outros serviços municipais (WASHBURN, apud BERNARDES et al., 2017).

Figura 1 - Ilustração de uma cidade inteligente.

Fonte: IDS, 2021

Tradicionalmente, quando mencionamos cidades inteligentes ou campus inteligentes, geralmente se pressupõe a utilização da Internet das Coisas (IoT) como uma forma de explorar novas oportunidades e estratégias no uso da tecnologia. Para que projetos de cidade inteligente sejam criados e colocado a uso da população é

preciso que seja feito experimento em cima desses projetos para garantir o seu funcionamento, para isso a internet da coisa dentro das universidade abre espaço para a implementação do campus inteligente onde esses projetos ganham forças para ser implementados pois são ambientes menores e restrito e com bastante mão de obra para desenvolvimento desses projetos e que se assemelha bastante a uma cidade (TAROUCO et al., 2017).

O conceito do campus inteligente segue a mesma proposta e definição da cidade inteligente, onde ambos buscam meios mais rápidos e otimizados para oferecer um serviço de qualidade para as pessoas inseridas nesse meio (JACOSKI, et al., 2019). Porém a definição exata para campus inteligente não existe, pois, diversos autores têm uma visão diferente de campus inteligente. Para ilustrar o que cada autor pensa sobre o conceito de campus inteligente Ferreira e Araújo, (2018) agruparam as seguintes definições como mostrado na Tabela 1.

Tabela 1 - Definição de campus inteligente segundo Ferreira e Araújo.

Liu, X e Xu, G (2016)	*Smart Campus* é um é um ambiente integrado de trabalho, estudo e convivência baseado em Internet das Coisas.
Tikhomirov, V. (2015)	*Smart University* é um conceito que envolve uma modernização abrangente de todos os processos educacionais.
Kwok, L. (2015)	*Smart Campus* é um novo paradigma de pensamento pertencente a um ambiente de campus inteligente holístico que engloba pelo menos, mas não limitado a vários aspectos de inteligência, como o e-learning, redes sociais e comunicações para a colaboração no trabalho, sustentabilidade ambiental e de TIC com sistemas inteligentes de gerenciamento de sensores, cuidados médicos, gerenciamento de edifícios inteligentes com controle e vigilância automatizados de segurança e governança transparente do campus
Abuarqoub et al (2017)	O *Smart Campus* oferece serviços em tempo hábil, reduz o esforço e reduz os custos operacionais. Implica que a instituição adotará tecnologias avançadas para controlar e monitorar automaticamente instalações no campus e fornecer serviços de alta qualidade para a comunidade do campus, ou seja, estudantes e funcionários.

Yu et al (2011)	Os *campus* inteligentes são construídos para beneficiar os professores e alunos, gerenciar os recursos disponíveis e melhorar a experiência do usuário com serviços proativos. Um campus inteligente varia de uma sala de aula inteligente, que beneficia o processo de ensino dentro de uma sala de aula, para um campus inteligente que fornece muitos serviços proativos em um ambiente em todo o campus (...) Campus é um ambiente social onde estudantes universitários têm muitas interações com seus amigos.
Bandara et al (2016)	*Smart Campus* é uma iniciativa para utilizar TICs em um campus universitário para melhorar a qualidade e o desempenho dos serviços, reduzir custos e consumo de recursos e se envolver de forma mais eficaz e mais ativa com seus membros.
Xiao, N. (2013)	Smart Campus é resultado da aplicação da integração da computação em nuvem e da internet das coisas (...) A estrutura de aplicativos do campus inteligente é uma combinação de IoT e computação em nuvem baseada na computação de alto desempenho e Internet.

Fonte: Ferreira e Araújo (2018)

Apesar das universidades ao redor do mundo compartilharem algumas características iguais, é preciso levar em consideração o contexto na qual está inserida observando seus aspectos culturais, sociais e organizacionais, pois isso irá refletir na solução de problemas de forma inteligente (JACOSKI, et al., 2019).

As universidades, especialmente as públicas no Brasil, apresentam uma grande diversidade em sua estrutura organizacional, muitas vezes enfrentando desafios de falta de mão de obra suficiente para gerenciar todos os alunos. Nesse contexto, as universidades públicas se tornam ambientes propícios para a implementação de soluções inteligentes que atendam às necessidades do campus.

Entre as diversas universidades públicas federais brasileiras que buscam implementar o conceito de campus inteligente, destacam-se instituições como a Unicamp, UFFJ e UFRB. Essas universidades já possuem em seus campi diversos projetos em andamento que envolvem o desenvolvimento e aplicação de soluções inteligentes para otimizar suas operações e promover um ambiente acadêmico mais eficiente. Na Unicamp, por exemplo, alguns de seus projetos envolvem melhoria no

sistema de circulação de ônibus interno do campus, sistemas de controle de iluminação, sistema para controle de *smart lock* dentre outros. Segundo o próprio site da SmartCampus Unicamp (CAMPUS-UNICAMP, 2023) o campus inteligente tem como objetivo obter informações do campus que será levada para uma inteligência de controle para que sejam tomadas medidas mais assertivas e tornar o dia a dia no campus mais produtivo e eficiente. A Figura 2 a seguir ilustra um projeto utilizando esse conceito, o qual tem a finalidade de informar aos usuários do restaurante universitário (RU) do campus em questão, a quantidade de pessoas que tem atualmente na fila para acessar o RU, fazendo assim com que, quem precisa utilizá-lo possa verificar como está o movimento e assim aproveitar o momento em que a fila estiver menor.

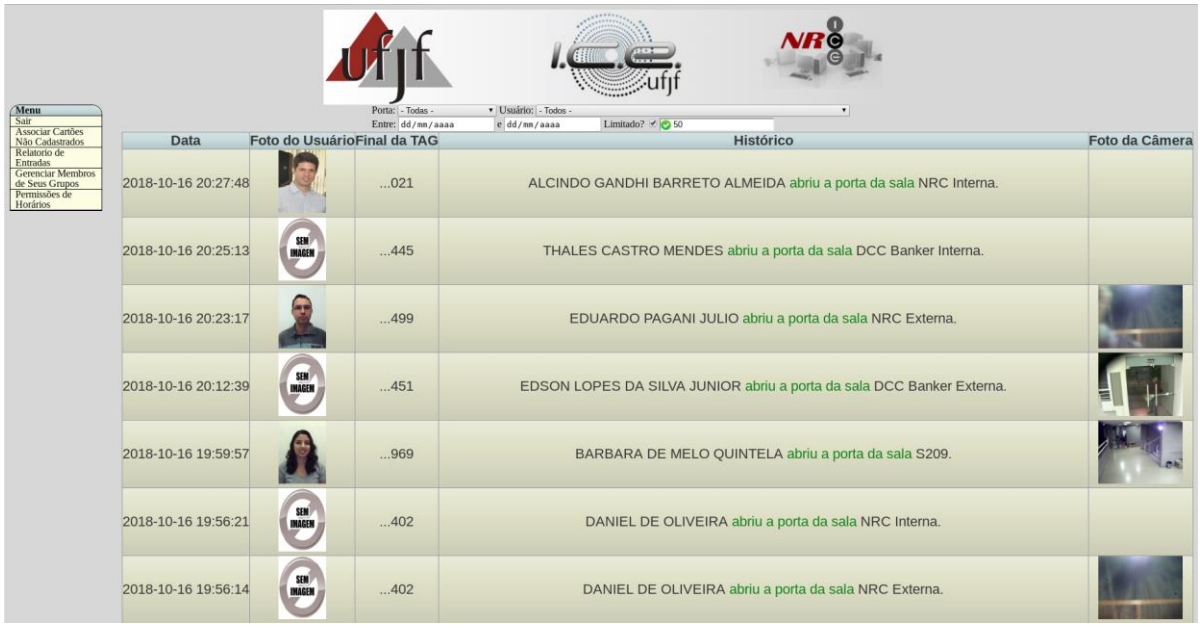
Figura 2 - Imagem da câmera de segurança da porta do RU da UNICAMP.

Fonte: SMART CAMPUS UNICAMP, 2023.

Outros campus, assim como a UNICAMP, também já trazem o conceito de internet das coisa para construir o campus inteligente, por exemplo a Universidade Federal de Juiz de Fora (UFJF), que já possui vários projetos dentro do conceito de campus inteligente (UFJF, 2023), como o sistemas U²Pass, que é utilizado para controlar o acesso de pessoas a determinadas salas do campus com um cartão RFID pessoal e um *hardware* capaz de ler este tipo de cartão e que possui comunicação

com as câmera de segurança que eventualmente fotografam, assim que o sistema faz a leitura. Esse sistema é fundamental quando se tem um alto fluxo de pessoas a um local que possui muitos patrimônios importantes para o funcionamento total do campus. Logo, esse sistema auxilia nesse controle para que somente pessoas com autorização prévia possam acessá-los. A Figura 3 a seguir mostra como é a visão do sistema por parte dos administradores.

Figura 3 - Sistema de administração do U²Pass



Fonte: UFJF, 2023

Com o intuito de dar início à implementação do conceito de Campus Inteligente na UFES de Alegre, propõe-se a implementação deste projeto como um primeiro passo para a aplicação desse conceito na instituição. O projeto tem como foco aprimorar o controle dos laboratórios e proporcionar maior comodidade aos usuários, permitindo o acesso a informações essenciais de qualquer lugar, por meio de dispositivos móveis conectados à rede de internet.

Dentro da UFES, o campus Alegre já conta com alguns sistemas como o sistema RU UFES, que é um projeto com objetivo de controlar a demanda de refeição a ser preparada para evitar desperdício. Assim, espera-se que o conceito de campus inteligente se expanda e ganhe força dentro da universidade, tornando-se centralizado e integrado a todas as suas áreas. Essa abordagem não apenas beneficiará os

usuários finais, mas também terá impactos positivos na economia e nas questões ambientais, pela administração da instituição.

3.2 Tecnologias empregadas

As tecnologias empregadas em um projeto ou em uma organização são os recursos tecnológicos utilizados para atingir os objetivos do projeto ou da organização. Elas podem incluir *hardware*, *software*, equipamentos e outros tipos de tecnologia. A escolha das tecnologias empregadas depende de vários fatores, como os objetivos do projeto ou da organização, o orçamento disponível, as necessidades dos usuários e os requisitos de segurança e privacidade.

3.2.1 Linguagem de programação

Uma linguagem de programação é uma método de padronizar a linguagem de máquina na qual são enviadas instruções para o computador para executar tarefas. Essas linguagens seguem algumas regras sintáticas na qual dizem respeito a como devem ser escritas e o que deve ter em seu conteúdo (GOTARDO, 2015). Para esse projeto a linguagem de programação Java Script.

Javascript é uma linguagem de programação mais voltada para o meio web, porém sua flexibilidade e adaptabilidade faz com que se torne multiplataforma, atendendo a todas as plataformas desde que tenham interpretadores Javascript (FLANAGAN, 2013). Além da sua baixa curva de aprendizado para aqueles que já possuem um certo conhecimento em qualquer outra linguagem de programação.

A linguagem Javascript é uma linguagem de alto nível, dinâmica e interpretada que se baseia na sintaxe da linguagem Java, mas que, porém, não é necessário saber Java para saber Javascript (FLANAGAN, 2013).

3.2.2 Frameworks

Os *frameworks* são estruturas de códigos genéricos conjuntos, que permitem com que sejam desenvolvidos sistemas baseados no código. Os *framework*

funcionam com uma base modelo para novos sistemas, quando usado fornecem uma estrutura na qual o sistema pode ser criado de maneira mais rápida e eficaz (BARRO, 2022).

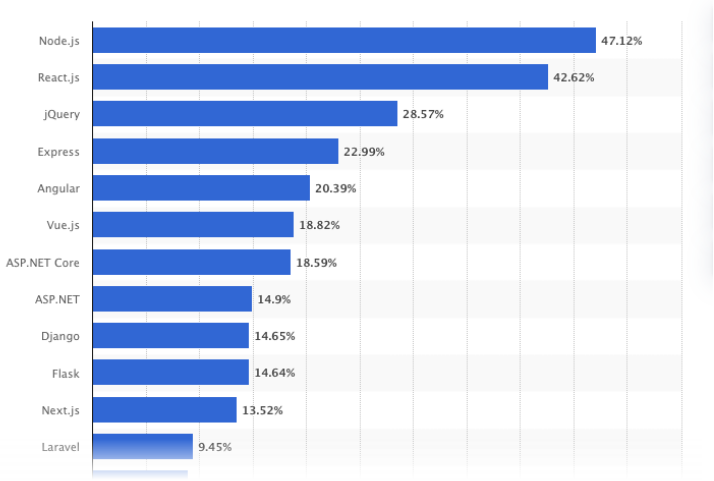
3.2.2.1 React-Native e Node JS

O React-Native é um *framework* Javascript criado pelo facebook na qual seu uso se dá como base para criação de *softwares* para dispositivos móveis android e ios, contando com diversas bibliotecas que facilitam o desenvolvimento e os deixam mais completos, assim também contam com uma comunidade que ajuda na manutenção do *framework* para que ele sempre se mantenha atualizado seguindo o ‘que há de mais recente no ambiente de desenvolvimento móvel (TREINAWEB, 2020).

Assim como o React, o Node.js também é um *framework* Javascript, porém, com sua alta escalabilidade, tendo diversas bibliotecas que deixam o sistema ainda mais completo, além da possibilidade de rodar uma aplicação Javascript sem precisar utilizar um navegador, fazendo com que seu uso seja voltado para o desenvolvimento de micro serviços e componentes de aplicação *server-side* (NODE JS, 2023).

A adoção desses dois *frameworks* se dá pela sua curva de aprendizado por ambos compartilharem da mesma linguagem de programação, e por conta de sua popularidade quando comparado a outros *frameworks* que desempenham a mesma função como é o caso do Node.js e React.js como mostrado na Figura 4, somado ao número de recursos e suporte para o desenvolvimento de sistemas móveis.

Figura 4 - Ranking de tecnologias mais populares em 2022



Fonte: *Statista,2023*

3.2.3 SGBD

Um Sistema de Gerenciamento de Banco de Dados (SGBD) é um *software* que atua no controle de um ou vários bancos de dados. Esse *software* contém interfaces gráficas com diversas ferramentas que auxiliam o desenvolvedor, gerenciar e controlar toda a estrutura de banco de dados de seu projeto (RAMAKRISHNAN, 2008).

Ainda de acordo com Ramakrishnan, (2008) Dentro de um projeto é possível gerenciar todo o banco de dados de maneira manual, porém para isso é preciso que seja escrito diversos código onde esse código vai refletir no gerenciamento do banco, tais como: comando para buscar nomes de uma tabela e o valores que ela armazena (RAMAKRISHNAN, 2008).

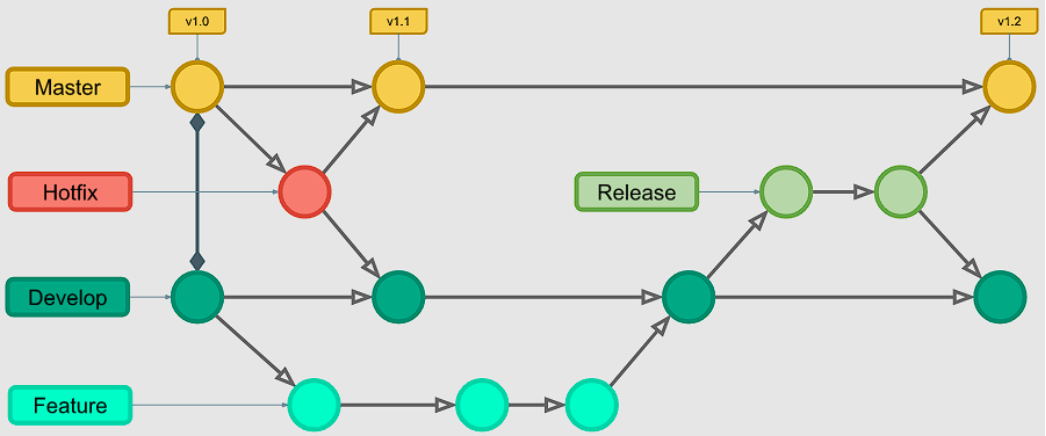
3.2.4 Controle de versão

A fim de viabilizar um efetivo controle sobre as versões do sistema, é imprescindível a adoção de um mecanismo de controle de versões no projeto. Neste contexto, a plataforma de desenvolvimento colaborativa GitHub (GITHUB, 2023) tem sido amplamente acolhida pela comunidade de desenvolvedores. Em conjunção com o GitHub, é comum empregar o gerenciamento de versões por meio do Git workflow,

uma abordagem que facilita o controle e rastreamento das alterações efetuadas ao longo do desenvolvimento do software.

Para melhor exemplificar o funcionamento do *git workflow* no versionamento de código, a Figura 5 mostra como são feitos os relacionamentos entre as *branch*. A ramificação *master* é onde é controlada a versão principal, a ramificação *hotfix* é para correções urgentes da *master*, a ramificação *develop* é para controlar desenvolvimento de novas funcionalidades para a próxima versão, a ramificação *feature* é para desenvolvimento das funcionalidades separadamente e pôr fim a release é para controlar a versão antes de serem enviadas para o versionamento principal que é feito na *master* .

Figura 5 - Fluxo de versionamento de código usando a estrutura git workflow.



Fonte: Alura (2023).

3.2.5 Ferramentas CASE

Os engenheiros de *software* empregam esses métodos para diminuir o tempo e o esforço necessários para desenvolver seus aplicativos. Eles também querem padronizar o processo, aumentar o controle sobre o projeto e automatizar algumas tarefas rotineiras (SILVA; VIDEIRA,2001).

Desde muito cedo foi percebido a necessidade de ferramentas ou para auxiliar nos desenvolvimentos de *softwares*. As primeiras ferramentas foram os compiladores,

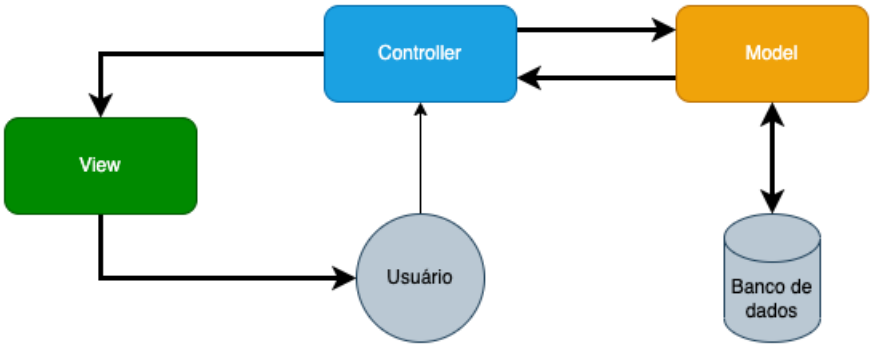
tradutores e assembladores, logo em seguida foi visto a necessidade de ferramentas para escrever código, debugar e versionar o código que estava sendo criado. Com a necessidade e criação de várias ferramentas que apoiassem o desenvolvimento, o número de ferramentas CASE cresceu consideravelmente, sendo necessária a criação de uma classificação para essas ferramentas, que em uma primeira fase, são classificadas em Ferramentas *Upper-Case*, as quais são utilizadas na elaboração do *software*, e Ferramentas *Lowercase*, utilizadas na implementação do *software* (SILVA; VIDEIRA,2001). As ferramentas, Figma, Notion e Draw.io são ferramentas CASE que se enquadram na classificação Uppercase, pois são voltadas para a prototipação, documentação, levantamento de requisitos e modelagem de diagramas. Já as ferramentas tais como Visual Studio Code, Postman e Github são utilizadas junto no desenvolvimento do *software*, sendo classificadas assim na primeira fase como ferramentas *Lowercase*.

3.2.6 Arquitetura do sistema

O padrão MVC que do inglês significa *Model-View-Controller,* tem uma dinâmica simples onde todas as solicitações de aplicativos são direcionadas para a camada Controller, que acessa a camada Modelo para processar tal pedido e, finalmente, exibe o resultado da camada View. O padrão MVC separa as camadas apresentação, lógica negócios e gerenciamento fluxo de aplicação, aumentando capacidade de reutilização e manutenção do projeto (LUCIANO et al., 2011).

O MVC tem uma arquitetura separada em três partes essenciais como é mostrado na Figura 6.

Figura 6 - Arquitetura MVC



Fonte: Do autor (2022)

A *Model* é a camada responsável por realizar a comunicação com banco de dados para extrair as informações, entretanto a *Model* não se limita apenas a comunicação com bancos de dados, podendo ser arquivos XML ou qualquer outro tipo de arquivo (DE LEMOS, et al., 2013).

A *View* é a camada responsável por exibir para o usuário final os dados obtidos através da requisição, não importando quais dados venha ou qual a ordem (DE LEMOS, et al., 2013).

A *Controller* é a camada responsável por pegar dados de entrada que vem da requisição do usuário e aplicar regras de negócio em cima desses dados para que quando enviado para a *Model* ele possa salvar esses dados ou ter retorno de mais dados para que sejam exibidos na view para o usuário (DE LEMOS, et al., 2013).

3.2.7 Teste de sistema

A estratégia de testes tem que estar de acordo com o andamento do projeto, sendo necessário que ela seja flexível e personalizada, porém, ao mesmo tempo, deve ser rígida para acompanhar o planejamento (PRESSMAN, 2011).

No processo de criação de um novo *software*, uma das maiores preocupações é em relação a qualidade do produto final. Para mitigar essa preocupação é necessário que se tenha uma estratégia de teste de *software*.

A estratégia de teste de *software* é um roteiro onde é definido os passos para que possa ser elaborado um plano de teste para nosso sistema, planejando quais testes devem ser aplicados, como devem ser aplicados, quando devem ser aplicados e quanto de recurso será necessário (PRESSMAN, 2007).

Na estratégia de teste do software proposto, foram considerados alguns tipos de testes voltados para a plataforma mobile. A seguir, serão apresentados alguns desses tipos de testes:

Testes funcionais: Esse teste também é conhecido como teste “caixa-preta”, para esse caso não é considerado o funcionamento interno do *software*. Para esse teste, os dados de entradas são fornecidos e é esperado um resultado conhecido como ilustrado na Figura 7. Caso o resultado não seja igual aquele esperado, o teste falha e caso a comparação seja igual, o teste é aceito com sucesso. Essa técnica pode ser aplicada a qualquer nível dos testes (NETO, 2007).

Figura 7 - Teste de caixa preta.

Fonte: O autor (2022)

Teste de sistema: Esse teste busca falhas e o mau comportamento do *software* em um ambiente que é o mesmo em que o usuário final utiliza, validando assim que o usuário final não irá encontrar problemas ao utilizar o *software* no dia a dia (NETO, 2007).

Teste de usabilidade: abrange diversos tipos de tarefas, tais como o desempenho do sistema, testar a disposição de textos e imagens no mesmo, a navegabilidade dentre outros, procurando assim problemas ligados a usabilidade do sistema em um contexto geral ou comparar com outros sistemas, a fim de se melhorar o que está sendo criado (FERREIRA, 2002).

3.2.8 Integração entre as tecnologias

Todas as tecnologias, arquitetura e ferramentas citadas acima são fundamentais para gerar um produto final que é o *software*. Cada uma delas tem um papel fundamental para garantir o desempenho, qualidade e controle do projeto.

A linguagem de programação Javascript, junto com o framework React-Native é a responsável por criar e exibir telas para o usuário onde constam todas as funções do sistema e também permitem que o usuário faça interações com as mesmas.

Utilizado o Javascript junto com a ferramenta Node.js, é possível desenvolver a parte de processamento das funções que serão solicitadas no aplicativo. Dessa forma, é possível aplicar toda a lógica necessária para retornar informações para o usuário ou salvar as informações que ele enviar. Para que a informação possa ser salva de forma pertinente, é necessário utilizar um banco de dados relacional para que a informação seja entregue de forma rápida, completa e disponível a qualquer momento para o usuário. Nessa etapa, tem-se a comunicação do *back-end*, junto ao banco de dados para buscar informações ou salvá-las, sendo possível, portanto, ter uma aplicação móvel.

3.3 Engenharia de Software

A Engenharia de Software é uma disciplina essencial que se dedica ao desenvolvimento sistemático e eficiente de software. Ela engloba princípios, métodos e técnicas para projetar, construir, testar e manter sistemas de alta qualidade, atendendo às necessidades dos usuários, garantindo segurança e confiabilidade, e cumprindo prazos e orçamentos.

Nas próximas seções, serão abordados alguns temas relacionados à Engenharia de Software.

3.3.1 Desenvolvimento incremental

Essa metodologia é especialmente indicada para projetos pessoais, e-commerce e principalmente para aqueles em que não é viável levantar todos os requisitos desde o início do projeto. Geralmente, nesses casos, tem-se uma visão geral do projeto e as versões são construídas com base em pequenas entregas incrementais ao longo do desenvolvimento (SOMMERVILLE, 2011). De acordo com PRESSMAN (2011), nas fases iniciais das implementações, é viável empregar uma equipe reduzida de pessoas e aumentar progressivamente essa quantidade à medida que novos incrementos são incorporados ao projeto.

Cada versão do sistema incorpora novas funcionalidades ou soluções para problemas específicos, sendo essas entregues ao cliente. Conforme apontado por PRESSMAN (2001, p. 62), “Quando se utiliza um modelo incremental, frequentemente, o primeiro incremento é um produto essencial” (PRESSMAN, p.62, 2001)”, permitindo ao cliente avaliar se o sistema atende aos requisitos solicitados, conforme destacado por SOMMERVILLE (2011).

3.3.2 Levantamento preliminar de requisitos

O levantamento de requisitos é uma das etapas iniciais de um desenvolvimento de sistema onde é descrito todas as funcionalidades e restrições que o sistema deve conter. Esse processo está diretamente ligado a necessidade do cliente para que o sistema sirva a uma finalidade específica (SOMMERVILLE, 2011).

Atualmente temos duas classificações para os requisitos de sistemas, os funcionais e não funcionais:

1. “Requisitos funcionais. São declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações” (SOMMERVILLE, p 59, 2011).

2. “Requisitos não funcionais. São restrições aos serviços ou funções oferecidos pelo sistema. Incluem restrições de timing, restrições no processo de desenvolvimento e restrições impostas pelas normas.” (SOMMERVILLE, p 59, 2011).

3.3.3 Diagrama de caso de uso

Os casos de são técnicas de descoberta de requisitos. Essa técnica já é fundamental na linguagem de modelagem unificada (UML - do inglês unified modeling language). De maneira mais simplificada, o caso de uso identifica os autores que vão estar envolvidos em uma em uma interação e dá nome para essas interações, e com essa identificação feita é adicionada informações que descrevem como será a interação com o sistema (SOMMERVILLE, 2011)

“Os casos de uso são documentados por um diagrama de caso de uso de alto nível “(SOMMERVILLE, p.74, 2011). O conjunto desses casos dos casos de uso tem como finalidade representar todas as interações que estão presentes nos requisitos do sistema. Para representar atores que vão estar no caso de uso é usada uma figura conhecida como “palito”. Os atores podem ser outras pessoas ou até mesmo outro sistema que interage com aquele caso. Já as classes de interação são representadas por uma elipse e a ligação entre atores e classe é feita por uma linha, opcionalmente essa ligação também pode ser representada por flecha onde pode-se ficar mais objetivo a origem da interação e o destino (SOMMERVILLE, 2011).

3.3.4 Narrativa caso de uso

Os casos de uso são importantes em um desenvolvimento de *software* pois eles identificam as interações entre usuários e seu sistema ou outros sistemas. Para um melhor entendimento do caso de uso, cada um deve conter uma descrição textual, e este por sua vez pode ser ligado a outros modelos UML que vão fazer com que o cenário do caso de uso fique mais detalhado. (SOMMERVILLE, 2011).

3.3.5 Modelagem de domínio

A modelagem de domínio pode ser vista com uma representação visual de conceitos e objetos do mundo real em um domínio de problema. Para ilustrar essa

representação é utilizado a notação da UML que é definido um conjunto de diagramas onde não são definidas as operações (LARMAN, 2002). Dentro da UML existem vários diagramas que ajudam na ilustração dos modelos, tais como diagrama de classes, diagrama de pacotes, diagrama de componentes, dentre outros.

3.3.6 Diagrama de pacotes e componentes

O diagrama de pacotes pode ser utilizado com outros diagramas da UML ou não, pois sua aplicação é independente. Esse diagrama fornece uma visão geral do sistema onde é possível observar a todos subsistemas que o compõem e tem o propósito de apresentar as divisões lógicas do sistema e as interações entre elas.

O diagrama de componentes está diretamente ligado a como foi desenvolvido o sistema a nível de linguagem de programação. Esse diagrama, como o próprio nome já diz, mostra os componentes do sistema e como foi implementado a nível de módulos, bibliotecas, *WebService*, formulário, arquivos etc. de também mostra como foi a interação entre eles para que o sistema se comporte como o ‘que foi modelado (GUEDES, 2018).

3.4 Banco de dados

Um banco de dados é uma coleção de dados inter-relacionados, utilizados para armazenar informações referentes ao software criado. Essa estrutura de dados é de fundamental importância para o desenvolvimento do sistema. Para efetuar o desenvolvimento de um sistema de forma adequada, é essencial ter conhecimento ou planejar a forma como os dados serão armazenados no banco e como ocorrerá a relação entre eles.

3.4.1 Modelagem de dados

A Modelagem de dados é uma descrição das informações que estão em um banco de dados (HEUSER, 2009). Com a análise dessa modelagem pode-se

entender os dados e quais tecnologias de armazenamento e gerenciamento de dados serão melhor aplicadas para esses dados(AWS, 2022).

Ainda de acordo com Heuser (2009) nos projetos de banco de dados geralmente são considerados dois níveis de abstração de modelo de dado, o do modelo conceitual e o do modelo lógico

3.4.2 Modelo lógico

Um modelo lógico é uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Assim, o modelo lógico é dependente do tipo particular de SGBD que está sendo usado. O modelo lógico descreve a estrutura do banco de dados, conforme vista pelo usuário do SGBD (HEUSER, 2009).

3.4.3 Modelo físico

O modelo físico de um banco de dados relacional é uma representação concreta da estrutura lógica e conceitual do banco de dados. Ele descreve como as relações entre as tabelas e os campos são armazenadas e organizadas em termos de arquivos e índices. Esse modelo é fundamental para o bom desempenho e a eficiência do banco de dados, pois determina como os dados serão armazenados fisicamente nos discos do servidor (RAMAKRISHNAN, 2008).

3.5 Backend

Backend é a parte do sistema de software que lida com a lógica de negócios, a manipulação de dados e a comunicação com outros sistemas ou serviços. É responsável pelo processamento e armazenamento de dados, gerenciamento de servidores, interação com bancos de dados e APIs, segurança, autenticação e autorização de usuários, entre outras responsabilidades (TREINAWEB, 2017).

Para se ter um *backend* robusto e simples é preciso que sua estrutura seja de fácil entendimento e bem separada em módulos.

3.5.1 Routes

As *routes* são encarregadas de encaminhar as solicitações HTTP recebidas pelo servidor para a função apropriada na controller que foi responsável por manipulá-las. Tipicamente, as *routes* são definidas em um arquivo distinto do restante do código da aplicação e são especificadas com base no método HTTP (como GET, POST, DELETE etc.) e no caminho da URL solicitada (EXPRESS.JS, 2023).

3.5.2 Controller

Um *controller* representa a função que administra a lógica de negócios de uma rota específica da aplicação web. A função primordial da *controller* é receber os dados submetidos pela requisição, processá-los, e, se preciso, interagir com o banco de dados antes de enviar uma resposta ao cliente (STROKLABS, 2018).

3.5.3 Útil

O *utils* é um módulo onde foi definido um conjunto de funções e variáveis utilitárias que podem ser usadas para realizar tarefas comuns, como formatação de strings, criação de objetos de erro, codificação e decodificação de dados, manipulação de datas e muito mais.

3.6 Frontend

O *frontend* é a parte da aplicação que interage diretamente com o usuário final, ou seja, é a camada responsável pela apresentação e interatividade do site ou aplicativo. O desenvolvimento *frontend* inclui a criação de elementos visuais, como *layout*, tipografia, cores, imagens, ícones e animações, além da implementação de recursos interativos, como botões, formulários, menus, efeitos de rolagem e muito mais (TOTVS,2022) .

3.6.1 Components

Os *components* são blocos de construção fundamentais usados para criar a interface do usuário no aplicativo. Cada *component* é uma unidade independente que pode ser reutilizada em diferentes partes do aplicativo. Os *components* podem conter outros *components*, permitindo a criação de uma hierarquia de *components* para construir a interface do usuário. Os *components* em *React Native* são escritos em Javascript e foram estilizados usando o *CSS-like StyleSheet API* (MDN WEB DOCS, 2023). Eles são a base para a construção de aplicativos escaláveis e modulares em React Native.

3.6.2 Config

A *config* refere-se ao arquivo que define configurações específicas do projeto, como a configuração do ambiente de desenvolvimento, as configurações do aplicativo, as configurações de compilação, entre outras. As configurações são usadas para personalizar e ajustar o comportamento da aplicação durante o desenvolvimento e a compilação.

3.6.3 Context

O *Context* é o recurso que permite compartilhar dados entre componentes sem precisar passar explicitamente esses dados como *props* (parâmetros). O *Context* permite criar um "canal" de comunicação global entre componentes, que é usado para compartilhar dados, configurações, temas e outras informações importantes para a aplicação. O *Context* é especialmente útil quando se lida com componentes profundamente aninhados ou quando há muitos componentes que precisam compartilhar os mesmos dados. O *Context* é implementado por meio do *Provider*, que define o contexto global, e pelo *Consumer*, que permite que os componentes acessem esse contexto. O uso adequado do *Context* ajuda a simplificar o código e melhorar a escalabilidade da aplicação.

3.6.4 Global

O *global* refere-se a um objeto que pode ser acessado em qualquer lugar da aplicação, independentemente de sua hierarquia de componentes. As variáveis globais são úteis para armazenar dados que precisam ser compartilhados entre vários componentes ou para definir funções que precisam ser acessadas em todo o aplicativo. Para esse projeto o *global* tem como principal função armazenar variáveis para controlar a folha de estilos da aplicação.

3.6.5 Hooks

Os *hooks* são funções especiais que permitem usar recursos avançados do React, como o *state* e o *lifecycle*, em componentes funcionais (REACTJS, 2021). Os *hooks* são uma maneira de reutilizar a lógica de um componente sem precisar criar uma classe. Alguns dos *hooks* mais comuns em React Native são o *useState*, que permite criar e gerenciar o *state* de um componente funcional, e o *useEffect*, que permite executar efeitos colaterais em resposta a mudanças no state ou em outros eventos. Os *hooks* são fáceis de usar e ajudam a simplificar o código, tornando o desenvolvimento em React Native mais rápido e eficiente.

3.6.6 Routes

O *Routes* se refere às rotas ou caminhos que o aplicativo pode seguir para navegar entre as diferentes páginas ou telas. As rotas podem ser gerenciadas por um "*router*", que é um componente React que mapeia as rotas para as páginas correspondentes.

3.6.7 Utils

O *utils* é semelhante a homônima no *backend*. Possui módulos que contêm funções utilitárias que podem ser reutilizadas em diferentes partes do aplicativo. Essas funções podem ser usadas para realizar tarefas comuns, como formatação de dados, manipulação de *strings*, validação de entrada, gerenciamento de datas, entre outras.

4. Metodologia

Considerando a relevância dos problemas identificados no âmbito acadêmico, e com o objetivo de garantir a efetividade deste trabalho, foi adotada uma abordagem de pesquisa aplicada. Por meio dessa metodologia, combinada com análises dos processos de Gestão de Laboratórios de Informática existentes, e referências que apontam o avanço tecnológico, foi possível desenvolver um sistema que promova maior eficiência nesse processo e estimule a interação dos alunos com os laboratórios de informática.

Para desenvolver esse projeto junto com a pesquisa aplicada foi utilizada a metodologia de desenvolvimento incremental que segundo Sommerville (2011) prima a criação de pequenas versões do seu *software* que são disponibilizadas para o cliente, onde pode-se obter *feedbacks* dos mesmos e, através destes, sejam adicionadas mais funcionalidades ou melhorar aquilo que já está feito, sempre sobrescrevendo a versão anterior.

O levantamento e a análise dos requisitos foram definidos em reuniões diretamente com o atual coordenador de laboratórios onde o mesmo já tinha um sistema web em funcionamento e apoiou totalmente em toda a análise dos requisitos e funcionalidades que necessitam estar no sistema. Juntamente com os requisitos, também foi definida a prototipação para apoiar na criação das telas e assim entregar um software com o *layout* bem simples e objetivo, pensado na melhor experiência para o usuário final.

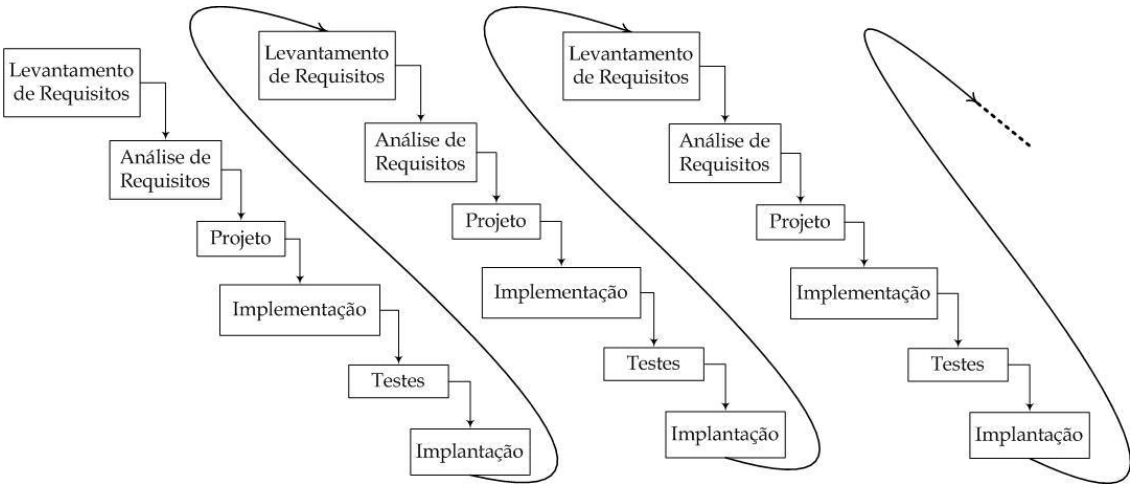
Para ter um controle sobre as versões disponibilizadas ao longo do projeto foi utilizado o controle de versão GitHub, que centralizou o código desenvolvido.

Para um entendimento mais aprofundado do sistema, nas próximas subseções foram descritas as etapas, já citadas e demais etapas, que foram utilizadas como apoio no desenvolvimento do projeto.

4.1 Desenvolvimento incremental

Com base na abordagem de desenvolvimento incremental adotada, a Figura 8 ilustra as etapas pelas quais o software foi construído. O processo teve início com o levantamento dos requisitos, descrevendo as funcionalidades que deveriam ser incorporadas ao sistema. Esses requisitos foram cuidadosamente analisados, e com base neles, foi possível dar início ao planejamento e à implementação do sistema.

Figura 8 - Estrutura do desenvolvimento incremental.



Fonte: Wiki IFSC Campus São José (2006)

Conforme o desenvolvimento progredia, novas funcionalidades foram incorporadas ao software, com base nos requisitos definidos e analisados anteriormente. Em cada etapa, foram realizados testes para garantir o correto funcionamento do sistema e evitar possíveis problemas.

Através da abordagem incremental, o projeto foi sendo construído de forma iterativa e progressiva, permitindo ajustes e melhorias contínuas, de acordo com as necessidades do cliente e as demandas do projeto. Ao final, o sistema foi testado, implantado e entregue, refletindo o resultado do processo de desenvolvimento incremental.

Para que o sistema siga os requisitos levantados e para que possam ser avaliados e testados previamente pelo coordenador de laboratórios, o

desenvolvimento foi entregue em pequenas partes até que todos os requisitos estivessem sido adicionados ao sistema.

4.2 Levantamento preliminar de requisitos

Assim como já abordado na seção 3.3.2, o levantamento preliminar de requisitos é uma etapa crucial no processo de desenvolvimento de software. Trata-se de uma fase inicial onde os analistas e demais stakeholders buscam obter uma compreensão abrangente das necessidades e expectativas do cliente em relação ao sistema a ser construído.

Durante o levantamento preliminar de requisitos, são considerados tanto os requisitos funcionais, que descrevem as funcionalidades e comportamentos do sistema, como os requisitos não funcionais, que abrangem aspectos de desempenho, segurança, usabilidade e outros critérios que definem a qualidade e as restrições do sistema a ser desenvolvido.

4.2.1 Requisitos funcionais

Para essa etapa de trabalho foram definidos os seguintes requisitos funcionais:

RF01 - Exibir regulamentos de utilização dos laboratórios - O sistema deve permitir que qualquer usuário possa ter acesso aos documentos em formato de pdf sobre a utilização, a infraestrutura, o contato e os regulamento dos laboratórios.

RF02 - Solicitar monitoria voluntária - O sistema deve permitir que o usuário possa ter acesso a uma *webview* onde poderá se candidatar à monitoria voluntária, informando-o antes sobre o regulamento da candidatura.

RF03 - Exibir o monitoramento de ocorrências - O sistema deve exibir somente para o coordenador todas as ocorrências que aconteceram dentro dos laboratórios.

RF04 - Aplicar um posicionamento sobre o andamento da ocorrência - O sistema deve permitir ao coordenador que atualize a ocorrência aplicando um posicionamento para ela sendo esses posicionamentos:

- Resolvido: Onde a ocorrência relatada já foi resolvida
- Em análise: Onde a ocorrência está sendo analisada
- Deletar: Onde a ocorrência já foi resolvida ou precisa ser deletada por outros motivos.

RF05 - Notificar sobre o posicionamento da ocorrência - O sistema deve enviar um *push-notification* para o monitor que relatou a ocorrência sempre que uma das posições do RF04 tenha sido aplicada.

RF06 - Exibir o monitoramento de suportes solicitados - O sistema deve exibir, somente para o coordenador, todas as solicitações de suporte.

RF07 - Aplicar um posicionamento sobre o andamento do suporte - O sistema deve permitir ao coordenador que atualize a solicitação de suporte aplicando um posicionamento para ele, sendo esses posicionamentos:

- Resolvido: Onde a solicitação de suporte já foi resolvida
- Em análise: Onde a solicitação de suporte está sendo analisada
- Deletar: Onde a solicitação de suporte já foi resolvida ou precisa ser deletada por outros motivos.

RF08 - Notificar sobre o posicionamento da solicitação de suporte - O sistema deve enviar um *push-notification* para o monitor que relatou a ocorrência sempre que uma das posições do RF04 tenha sido aplicada.

RF09 - Exibir listagem de monitores - O sistema deve exibir uma lista com todos os monitores vigente.

RF10 - Excluir monitores - O sistema deve permitir que seja removido qualquer monitor vigente.

RF11 - Editar horário dos monitores - O sistema deve permitir que qualquer monitor possa ter seu horário alterado.

RF12 - Validar a edição de horário dos monitores - O sistema deve validar a edição de horário dos monitores para que não haja conflito entre horários.

RF13 - Importar novos monitores - O sistema deve permitir que possa ser adicionado novos monitores através da importação de uma planilha no formato xlsx que deverá estar armazenada no dispositivo do coordenador.

RF14 - Visualizar grade de horários dos monitores - O sistema deve exibir os horários dos monitores vigentes em cada dia da semana.

RF15 - Enviar notificações push - O sistema deve permitir somente ao coordenador, enviar *push notification* personalizada para todos os usuários do sistema.

RF16 - Solicitar Suporte - O sistema deve permitir que qualquer usuário possa criar uma solicitação de suporte que será enviada ao coordenador.

RF17 - Exibir listagem de solicitações de suporte feitas pelo próprio usuário - O sistema deve exibir uma listagem com todas as solicitações de suportes feitas pelo próprio usuário.

RF18 - Solicitar cartão magnético - O sistema deve permitir a qualquer usuário que tenha feito reserva de sala anteriormente, que solicite que lhe seja liberado o cartão magnético para acesso ao laboratório reservado.

RF19 - Exibir listagem de solicitações de cartão magnético - O sistema deve permitir que o coordenador e gestores de acesso às salas da UFES possam visualizar a listagem de solicitação de cartão magnético.

RF20 - Aplicar posicionamento sobre solicitação de cartão magnético - O sistema deve permitir que o coordenador e gestores de acesso às salas da UFES possam aplicar um posicionamento sobre o controle do cartão magnético solicitado, sendo esses posicionamentos:

- Entregue: Quando é feita a entrega do cartão magnético para o solicitante.

- Devolvido: Quando é feita a entrega do cartão magnético para os gestores de acesso a salas ou coordenador de laboratório.

RF21 - Realizar login de usuário - O sistema deve permitir que o usuário efetue login no sistema utilizando as seguintes informações: número de matrícula e senha única fornecida pela universidade.

RF22 - Autenticação pelo sistema de login da UFES - O sistema deve autenticar o usuário juntamente com o sistema de autenticação de alunos da UFES (portal do aluno)

RF23 - Realizar login de gestores de salas - O sistema deve permitir que o gestor de sala efetue login no sistema utilizando as seguintes informações: combinação de números única fornecida pelo sistema e senha única.

RF24 - Receber notificação - O sistema deve permitir que todos os usuários possam receber e visualizar *push notification* enviados pelo coordenador ou por atualizações de suporte ou ocorrência.

RF25 - Enviar notificação ao atualizar horários de monitor - O sistema deve enviar um *push notification* toda vez que o monitor tiver seu horário alterado.

RF26 - Enviar notificação ao criar novo informe de ocorrência - O sistema deve notificar ao coordenador via push notification sempre que uma nova ocorrência for criada.

RF27 - Enviar notificação ao solicitar suporte - O sistema deve enviar um push-notification ao coordenador sempre que uma solicitação de suporte for aberta.

RF28 - Enviar notificação ao solicitar cartão magnético - O sistema deve enviar um *push-notification* ao coordenador sempre que uma houver solicitação de cartão magnético

RF29 - Informar ocorrência - O sistema deve permitir que o monitor possa informar que houve uma ocorrência no laboratório e que será enviada ao coordenador.

RF30- Exibir listagem de ocorrências informadas pelo próprio monitor - O sistema deve exibir uma listagem com todas as ocorrências informadas feitas pelo próprio monitor.

Os dados referente a esses requisitos funcionais são informados nas narrativas de caso de uso.

4.2.2 Requisitos não funcionais

Para essa etapa de trabalho foi definido os seguintes requisitos não funcionais:

RNF01 - O sistema deve ser desenvolvido usando o *framework react-native* para criar a aplicação e *Node.js* para criar as API`s.

RNF02 - O sistema deve usar a linguagem de programação Javascript.

RNF03 - O sistema deve ser responsivo em todos os tipos de aparelhos.

RNF04 - O sistema deve ter uma configuração para cada tipo de ambiente de desenvolvimento

RNF05 - O sistema deve ser disponibilizado pela loja de aplicativos Google Play.

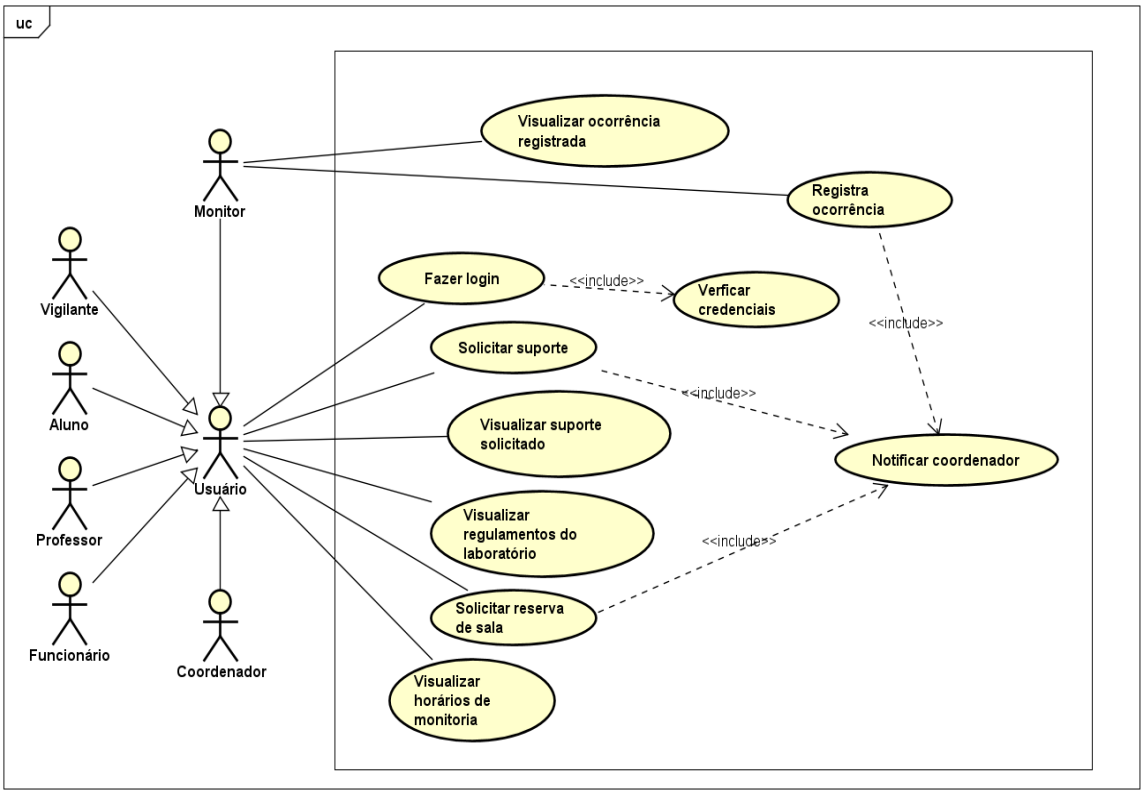
RNF06 - O sistema deve apresentar uma interface simples e intuitiva.

RNF07 - O controle de versão do sistema deve ser feito através do github.

4.3 Diagrama de caso de uso

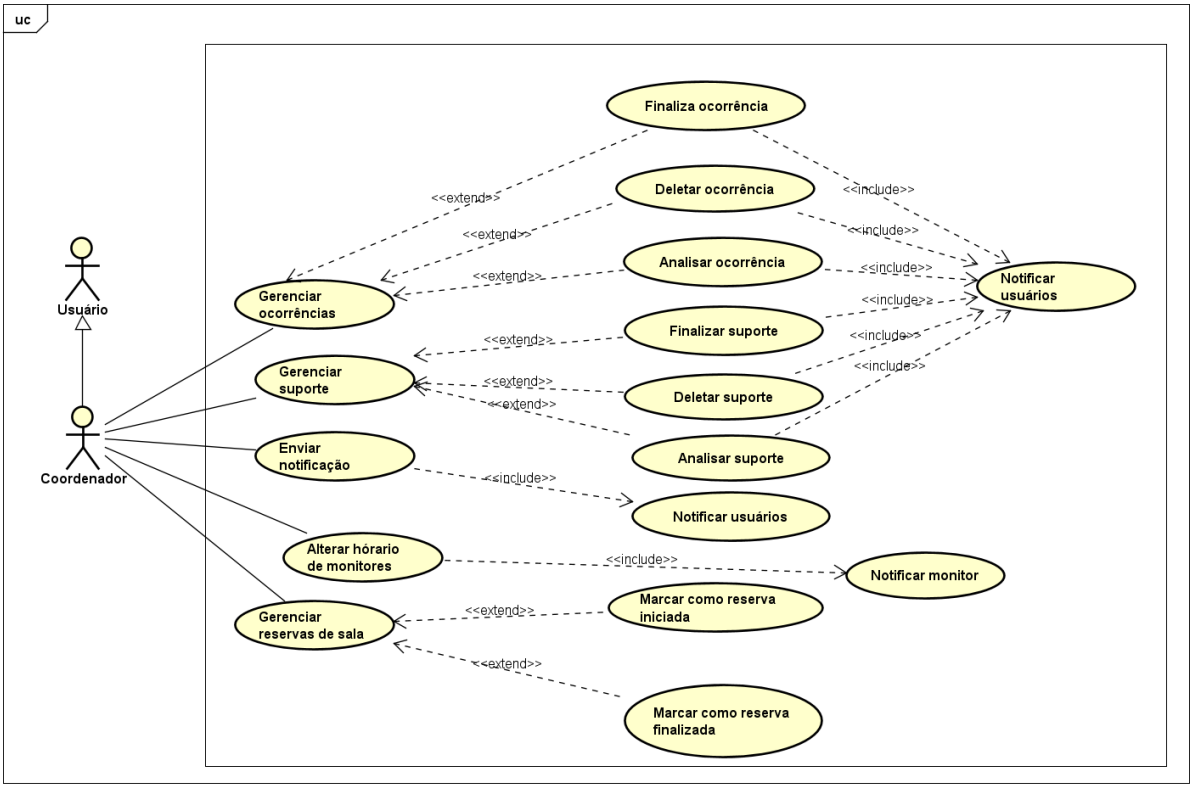
Os diagramas de caso de uso, conforme explicados na seção 3.3.3, são fundamentais para a elaboração dos requisitos funcionais. Para cada funcionalidade, é designado um autor responsável pela interação direta com a mesma. Para este trabalho, foram elaborados os seguintes diagramas de caso de uso conforme ilustrado na Figura 9 e 10:

Figura 9 - Diagrama de caso do sistema para usuários.



Fonte: O autor (2022)

Figura 10 - Diagrama de caso de uso do sistema para coordenador.



Fonte: O autor (2022)

4.4 Narrativa caso de uso

A seguir serão apresentadas duas descrições textuais sobre os casos de uso 001, para exibição de horários de monitores e o caso de uso 002, que é o envio de notificação informativa. As demais narrativas de caso de uso estarão disponíveis no Apêndice A.

UC001 - Exibir horário de monitoria

Descrição: Esse caso de uso descreve como deve ser feita a exibição dos horários de monitorias do semestre vigente.

Atores: Coordenador, monitor e usuário

Pré-condições: O usuário do sistema deve estar logado.

Pós-condições:

Fluxo normal

01 - O usuário clica no botão de menu de barra lateral

02 - O usuário seleciona o menu de monitor

03 - O usuário seleciona no menu de monitor o item "horário de monitoria”

04 - O sistema exibe os horários dos monitores vigentes

Fluxo Alternativo

01 - O sistema exibe o menu principal

02 - O usuário clica em “Horário de monitoria”

03 - O sistema exibe os horários dos monitores vigentes

UC002 - Envio de notificação informativa

Descrição: Esse caso de uso descreve como foi feito o envio e recebimento de notificações informativas sobre o funcionamento do laboratório.

Atores: Coordenador, monitor e usuário

Pré-condições: Todos usuários devem estar logados no sistema.

Pós-condições: Uma nova notificação será salva no banco de dados.

Fluxo normal:

01 - O coordenador acessar o menu lateral do sistema

02 - O coordenador seleciona o menu de “coordenador”

03 - O coordenador seleciona dentro do menu coordenador o item “enviar notificação”

04 - O sistema exibe a tela para envio de notificação

05 - O sistema solicita um título

06 - O sistema solicita uma mensagem que será enviada

07 - O coordenador clica em enviar

08 - O sistema salva a notificação no banco

09 - O sistema exibe um modal de confirmação de envio

Fluxo alternativo: Sem fluxo alternativo

Fluxo de exceção: Sem fluxo de exceção

FE01 - Ocorre falha no sistema e a notificação não é salva

01 - O sistema exibe a MSG001

02 - O sistema encerra a operação

Mensagens do sistema:

MSG001: “Erro ao enviar notificação. Entre em contato com o suporte”

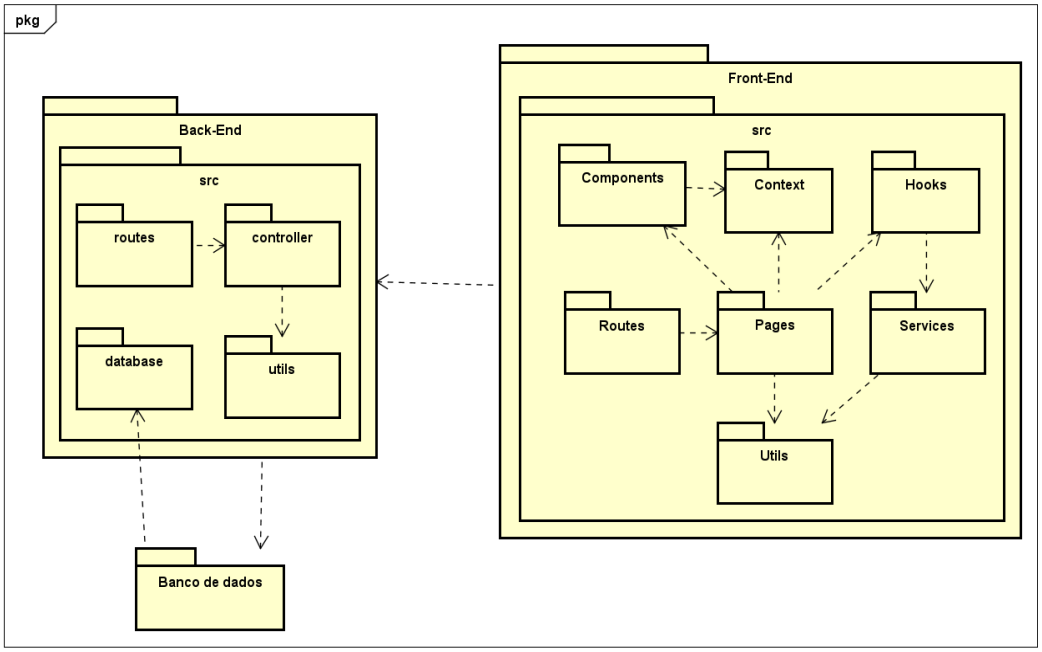
4.5 Modelagem de domínio

Diferente de outros projetos que tradicionalmente usam o diagrama de classe para ilustrar a integração e comunicações do sistema, bem como as diversas entidades programáveis que constitui seu domínio (naturalmente sendo representados por classes e objetos), para esse projeto foi ilustrado o diagrama de pacotes e o diagrama de componentes para mostra o comportamento dos subsistemas e dos componentes dentro do software, pois a linguagem que foi utilizado não segue o padrão orientada a objeto e sim orientada a protótipo e possui algumas características do padrão orientado a objetos (GRILO; FORTE, 2008).

4.5.1 Diagrama de pacotes

Para esse trabalho foi elaborado um diagrama de pacotes conforme apresentado na Figura 11.

Figura 11 - Diagrama de pacotes do sistema.

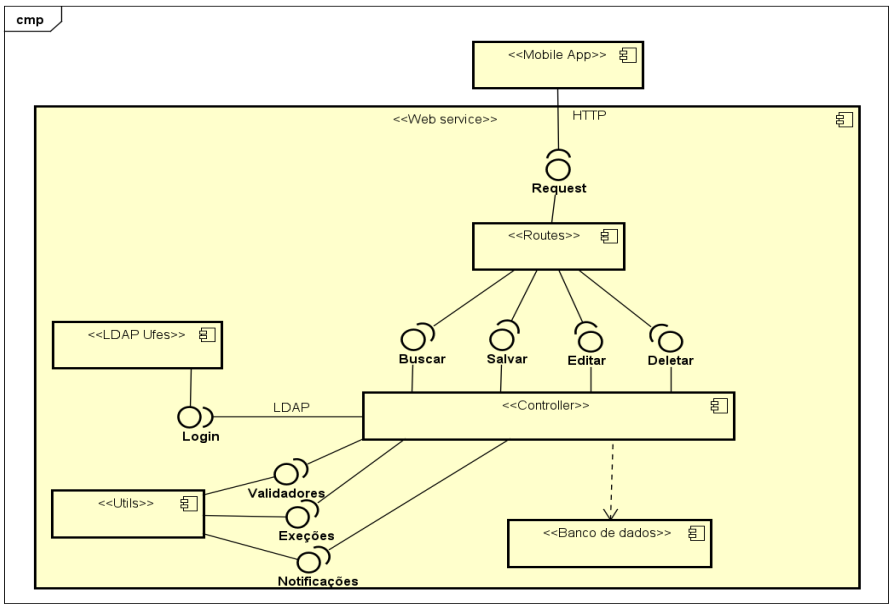


Fonte: O autor (2022)

4.5.2 Diagrama de componente

Para este trabalho foi elaborado um diagrama de componente, representado na Figura 12.

Figura 12 - Diagrama de componente do sistema.



Fonte: O autor (2022).

4.6 Modelo de banco de dados

Para o desenvolvimento deste trabalho, foram utilizados os modelos de banco de dados físico e lógico. Assim como abordado na seção 3.4.2 e 3.4.3, o modelo físico descreve a estrutura real de armazenamento dos dados, enquanto o modelo lógico representa a organização conceitual dos mesmos.

4.6.1 Modelo Lógico

Abaixo segue a representação do modelo lógico das tabelas Support e User do banco de dados, os demais modelos estão localizados no apêndice A.

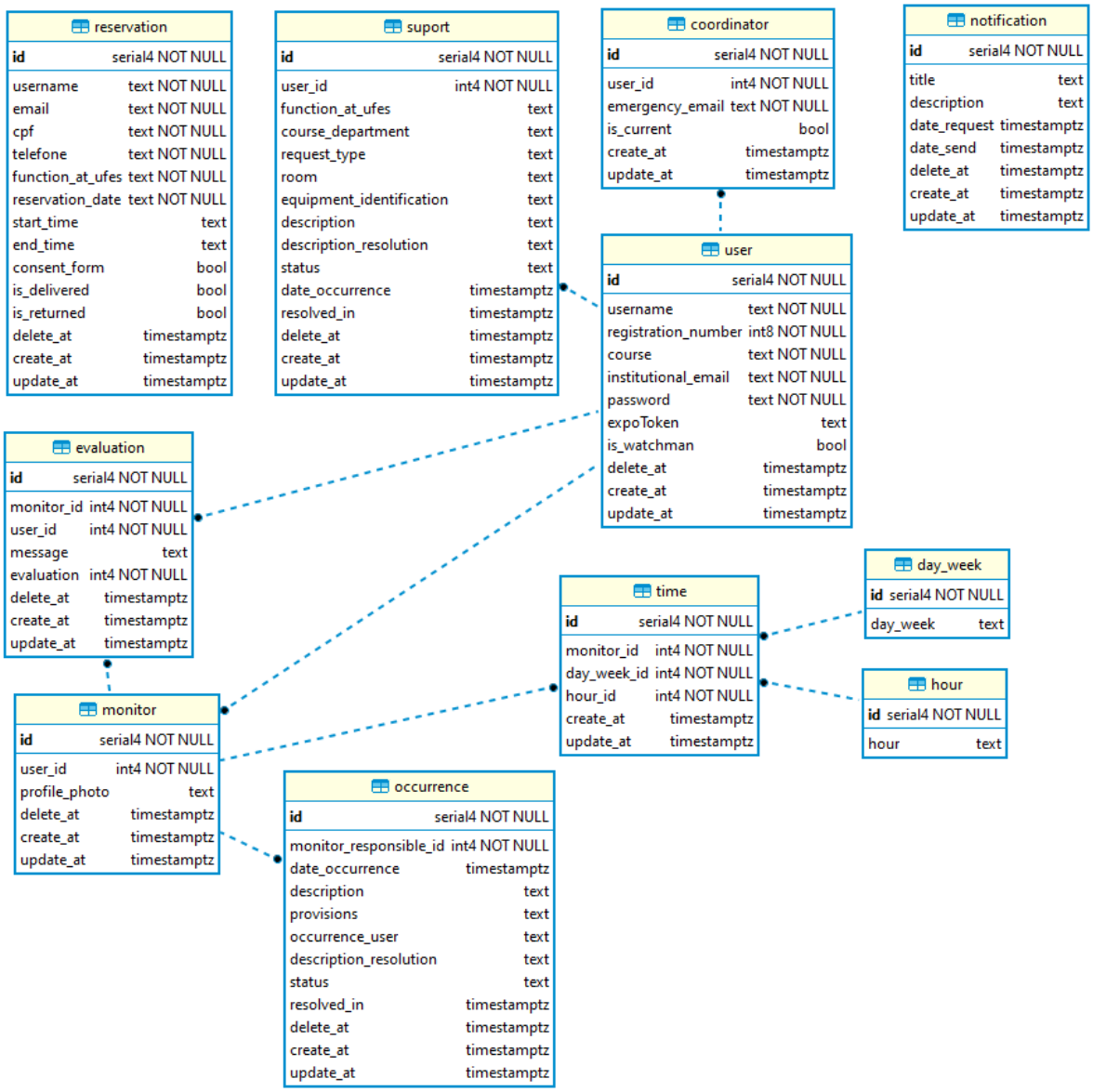
Support (id, user_id, function_at_ufes, course_department, request_type, room, equipment_identification, description, description_resolution, status, date_occurrence, resolved_in, delete_at, create_at, update_at)

User (id, username, registration_number, course, instutional_email, password, expoToken, is_watchman, delete_at, create_at, update_at)

4.6.2 Modelo Físico

A partir da concepção do modelo lógico é possível definir o Modelo Físico conforme ilustrado na Figura 13 a versão em português pode ser encontrada no apêndice C.

Figura 13 - Modelo físico do banco de dados.



Fonte: Do autor (2022)

4.7 Descrição geral do banco de dados

A seguir, será apresentada a descrição da tabela “*occurrence”* do modelo conceitual do banco de dados, como amostras, as descrições das demais tabelas estarão no Apêndice B.

A Tabela 2 apresenta um exemplo de estrutura de arquivo da tabela occurrence.

Tabela 2 - Tabela de occurrence.

Atributo	Tipo	Descrição
Id	SERIAL(4)	Chave primária da tabela
monitor_resposible_id	INT(4)	Chave estrangeira referente a tabela monitor.
date_occurrence	TIMESTAMP	Data e hora da ocorrência
description	TEXT	Descrição
provisions	TEXT	Provisões tomadas
occurence_user	TEXT	Nome do usuário envolvido
description_resolution	TEXT	Descrição da resolução
status	TEXT	Andamento da ocorrência
resolved_in	TIMESTAMP	Data e hora que foi resolvido
delete_at	TIMESTAMP	Data e hora que foi deletado
create_at	TIMESTAMP	Data e hora que foi criado
update_at	TIMESTAMP	Data e hora que foi atualizado

Fonte : O autor (2022)

4.8 Garantia da qualidade e testes de validação do sistema

Para esse *software* foram utilizados três tipos de teste, que são: Teste funcional, teste de sistema e teste de usabilidade.

O teste funcional foi aplicado a todas as requisições que foram feitas ao servidor, a fim de garantir que os dados que estão sendo inseridos no aplicativo possam ser os mesmo que é esperado no servidor, antes de salvá-los no banco de dados.

Para realizar o teste de sistema foi necessário um processo de construção do aplicativo e do servidor, onde todos dados que são apresentados ao usuário final, sejam exibidos também para esse caso de teste, a fim de garantir que após a construção do aplicativo e do servidor para envio aos usuário final os mesmos não tenham falhas ou divergências de informações.

Para o teste de usabilidade foram feitos testes mais focados na parte de interação com o usuário. Esse teste foi aplicado a cada entrega do aplicativo a fim de se obter feedbacks dos usuários, além do que o teste já pegar de falha para que assim na entrega final deste, o sistema possa fornecer a melhor experiência para o usuário final com navegação simples e desempenho satisfatório.

5. Desenvolvimento da aplicação

O desenvolvimento do sistema foi realizado utilizando Node.js para o *backend* e react-native para o *frontend,* ambas utilizando como linguagem base o Javascript. Cada uma dessas tecnologias possui sua própria estrutura distinta de projeto, a fim de atender melhor ao desenvolvimento.

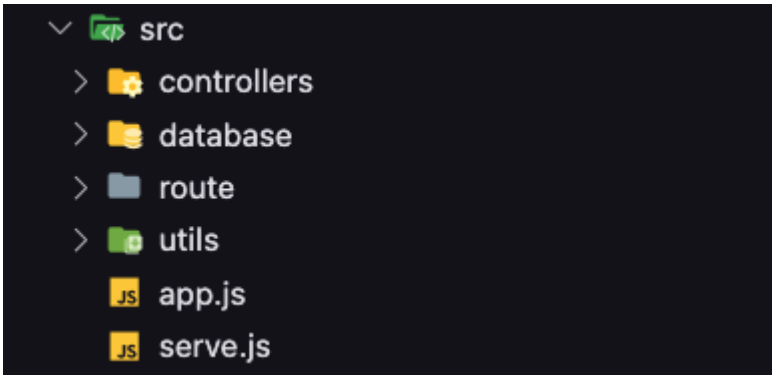
5.1 Backend

Para que fosse possível ter um sistema simples e robusto foi necessário elaborar um *backend* que se adequa às necessidades do projeto. Para isso nas seções abaixo será mostrado como foi realizado o desenvolvimento do mesmo.

5.1.1 Estrutura do projeto

Para obter o melhor entendimento do *backend* e melhorar a manutenção do mesmo foi desenvolvido na seguinte estrutura, conforme retratado na figura 14.

Figura 14 - Esquema de pasta do projeto.



Fonte: Do autor (2023)

Na tabela 3 é descrito de forma breve cada módulo da estrutura do projeto.

Tabela 3 - Descrição dos módulos da estrutura do projeto

Módulo	**Função/Descrição**
Controle	Recebe a solicitação HTTP, aplica as regras de negócio e faz a comunicação com a base de dados
Database	Cria a comunicação com banco, estrutura de tabelas e inserção de dados base nas tabelas.
Routes	Responsável por receber as solicitações HTTP e redirecionar para a controller certa
Utils	Contém funções utilitárias que podem ser reaproveitadas por todo backend

Fonte: Do autor (2023)

5.1.2 Routes

A Figura 15 ilustra como são definidos os redirecionamentos para o módulo de monitor na *routes*.

Figura 15 - Estrutura da routes para monitor

```
var monitorController = require('../controllers/monitorController');

router.post('/', monitorController.readWorksheet);
router.get('/timeMonitor/:id_monitor', monitorController.getTimeByMonitor);
router.delete('/:id_monitor', monitorController.removeMonitor);
router.get('/', monitorController.getMonitors);
router.put('/:id_monitor', monitorController.updateTimeMonitor);
router.get('/timesMonitor', monitorController.getTimesMonitors);
router.put('/timesMonitorOverlap/:id_monitor',monitorController.updateTimeMonitorOverlap);

module.exports = router;
```

Fonte: Do autor (2023)

As *routes* são desenvolvidas definindo o método HTTP da requisição (GET, POST, DELETE etc.) e também o *path* (caminho para o recurso) juntamente com os parâmetros, caso necessário. Por fim é definido o método na *controller* que será executada quando houver requisição para a rota definida.

5.1.3 Controller

A Figura 16 retrata como foi estruturada a função *getMonitors* definida dentro da controller *MonitorController*. Essa função é responsável por buscar todos os monitores que estão ativos, utilizando o *Knex.js* para facilitar a comunicação com o banco de dados.

Figura 16 - Estrutura da função getMonitor

```
async getMonitors(req, res,next){
    try{
        const monitors = await knex()
        .select('monitor.*','user.username')
        .from('monitor').join('user','user.id','=','monitor.user_id')
        .whereNull('monitor.delete_at')

        return res.status(200).json(monitors);
    }catch(error){
        erros.bussinesException(error)
        next(error)
    }
},
```

Fonte: Do autor (2023).

5.1.4 Conexão com banco de dados

Para a conexão do servidor *backend* com o banco de dados PostgreSQL, foram utilizados biblioteca *Knex.js* onde é possível construir consultas SQL dinamicamente, executar migrações de banco de dados, realizar transações e muito mais, tudo com uma interface fácil de usar e configurar.

Com a utilização do Knex.js foi definido um arquivo de configuração chamado *knexfile* onde nele criamos a conexão de bancos para os diferentes ambientes.

A figura 17 exibe como é feita a conexão para os bancos de dados de *production* e *developer,* onde é necessário definir o *host*, nome da base de dados, usuário e senha para a conexão.

Figura 17 - Estrutura do arquivo de configuração knex

```
module.exports = {
  development: {
    connection: {
      host: process.env.DB_HOST,
      database: process.env.DB_NAME,
      user: process.env.DB_USER,
      password: process.env.DB_PASSWORD,
    },
    migrations: {…
    },
    seeds: {…
    },
  },

  production: {
    connection: {
      host: process.env.DB_HOST,
      database: process.env.DB_NAME,
      user: process.env.DB_USER,
      password: process.env.DB_PASSWORD,
    },
    migrations: {…
    },
    seeds: {…
    },
  },
};
```

Fonte: Do autor (2023)

5.1.5 Útil

A figura 18 ilustra a estrutura da função *bussinesExceptionValidation* que é definida no módulo *utils* para que, sempre que um erro for encontrado no projeto, ele possa ter um comportamento padrão de falha, sendo gerado um *id* do tipo *hex* para que seja possível identificar com mais facilidade um erro quando ocorrer.

Figura 18 - Estrutura da função bussinesExceptionValidation

```
async bussinesExceptionValidation(error){
    error.status = 203;
    let id = crypto.randomBytes(10).toString('hex');
    console.log({
        erro_id : id,
        message :error.message
    })
    error.message = `${error.message}\n ID: ${id}`
    return {error}
},
```

Fonte : Do autor (2023)

5.2 Frontend

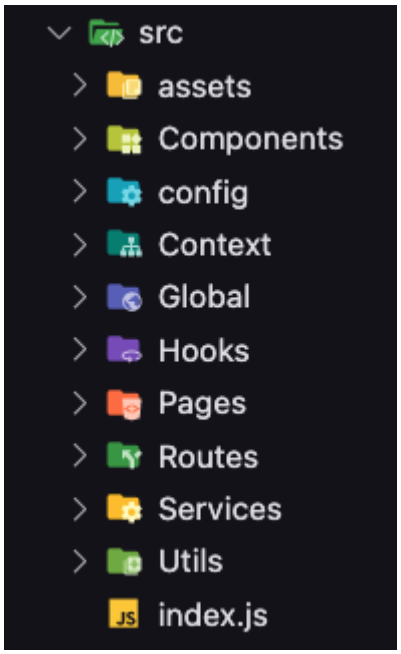
O frontend que é responsável pela comunicação direta com usuário final precisa estar bem estruturado e com informações de fácil acesso em sua interface fazendo assim com que se torne um aplicativo de fácil manuseio e de layout limpo para melhor experiencia do usuário, nas seções abaixo será abordado um como foi realizado o desenvolvimento do mesmo.

5.2.1 Estrutura do frontend

Para o *frontend* foi pensado em uma estrutura que, além da separação das camadas essenciais para o desenvolvimento, também foi feita a separação dos módulos para se ter um código bem segmentado, facilitando novas implementações e manutenções futuras.

A figura 19 mostra como foi planejada e desenvolvida a estrutura para o frontend.

Figura 19 - Estrutura de pasta do front end.



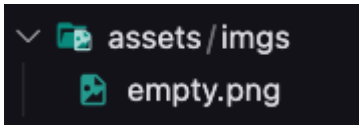
Fonte: Do autor (2023)

5.2.2 Assets

Os *assets* referem-se a arquivos estáticos, como imagens, fontes, arquivos de som ou vídeo, que são usados pela aplicação. Os ativos são carregados na memória do dispositivo durante o processo de compilação da aplicação e podem ser acessados por meio do sistema de módulos do React Native.

A figura 20 ilustra o esquema de pasta para os assets do sistema, onde são separados por tipos de arquivo (imagem, pdf etc.).

Figura 20 - Estrutura da pasta assets.

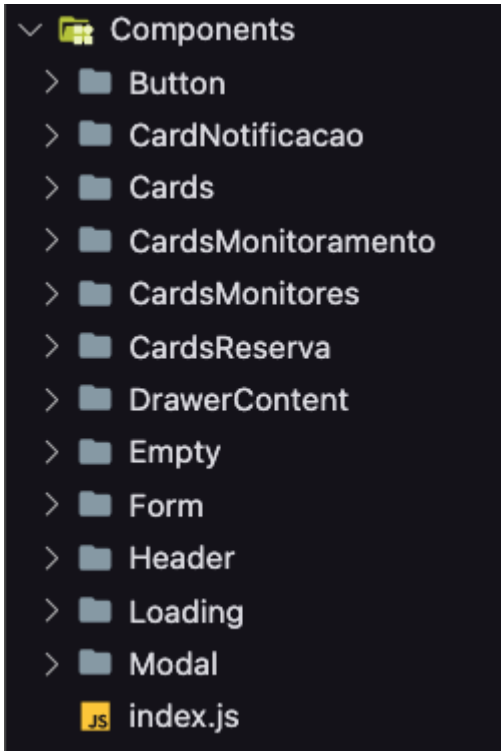


Fonte: Do autor (2023)

5.2.3 Components

A figura 21 ilustra a disposição dos componentes na estrutura de pasta de *components*, onde cada pasta é nomeada com o nome do *component* implementado.

Figura 21 - Estrutura da pasta Components;



Fonte: Do autor (2023)

5.2.4 Config

A figura 22 exibe a função de configuração de acesso aos serviços do *firebase*, onde são definidas chaves privadas que se referem somente ao projeto em questão. Por questões de confidencialidade e segurança do projeto, as chaves principais foram trocadas por “x” a fim de se esconder as chaves do projeto.

Figura 22 - Função de configuração do acesso à ferramenta Firebase.

```
import firebase from 'firebase/compat/app';
import 'firebase/compat/storage';

const firebaseConfig = {
    apiKey: 'xxxxx',
    authDomain: 'xxxxx.firebaseapp.com',
    projectId: 'meu-lab-xxxx',
    storageBucket: 'meu-lab-xxxxx.appspot.com',
    messagingSenderId: 'xxxxxxxxxxx',
    appId: 'xxxxxxxxxxxxxxxxxxxxxx',
};
if (!firebase.apps.length) {
    firebase.initializeApp(firebaseConfig);
}
export { firebase };
```

Fonte: Do autor (2023)

5.2.5 Context

A figura 23 retrata uma definição de um *Context* para as funções de autenticação do aplicativo e dados referente ao usuário.

Figura 23 - Funções de autenticação no aplicativo

```
function AuthProvider({ children }) {
    const [user, setUser] = useState();
    const [loading, setLoading] = useState(false);
    const { closeModalInfo, showModalInfo, getTokenExpo } = useApplication();

    const login = async (userLoginInfo) => {…
    };

    const loginAsWatchman = async (userLoginInfo) => {…
    };

    const cadastrar = async (dadosUsuario) => {…
    };

    const getUser = async () => {…
    };

    const loggout = async () => {…
    };

    useEffect(() => {…
    }, []);

    return <UsuarioContext.Provider value={{ login, user, loggout, loading, cadastrar, loginAsWatchman }}>{children}</UsuarioContext.Provider>;
}
```

Fonte: Do autor (2023)

5.2.6 Global

A figura 24 mostra como é definida no *global* as cores e fontes, para que sejam definidas em quaisquer lugares do sistema.

Figura 24 - Arquivo de variáveis globais do aplicativo relacionada a estilos

```
export default {
    colors: {
        primary: '#2c2c79',
        secondary: '#00FFFF',
        secondary_light: 'rgba(255,135,44, 0.3)',
        success: '#00c800',
        success_light: 'rgba(18,164,84,0.5)',
        attention: '#E83F5B',
        attention_light: 'rgba(232, 63, 91, 0.5)',
        danger: '#FF0000',
        alert: '#ffbf00',
        alert_light: 'rgba(253, 233, 16, 0.5)',
        black: '#000',
        shape: '#FFFFFF',
        title: '#363F5F',
        text: '#969CB2',
        text_dark: '#000000',
        background: '#E6E9ED',
        transparent: 'rgba(232, 63, 91, 0.0)',
        disable: '#dcdcdc',
    },

    fonts: {
        regular: 'Poppins_400Regular',
        medium: 'Poppins_500Medium',
        bold: 'Poppins_700Bold',
        thin: 'Poppins_100Thin',
        light: 'Poppins_300Light',
        semi: 'Poppins_600SemiBold',
    },
};
```

Fonte: Do autor (2023)

5.2.7 Hooks

A figura 25 ilustra como é definido o *hook* para o Módulo de Ocorrências, como funções que são definidas somente dentro do *hooks,* e só podem ser definidas dentro de outros *hooks.*

Figura 25 - Hooks de ocorrência onde são definidas as funções usadas em tela.

```
export default () => {
    const navigation = useNavigation();
    const { closeModalInfo, showModalInfo } = useApplication();
    const [loading, setLoading] = useState(false);

    const [ocorrencias, setOcorrencias] = useState();

    const [empty, setEmpty] = useState(false);

    useEffect(() => {…
    }, [ocorrencias]);

    const registrarOcorrencia = async (ocorrencia) => {…
    };

    const getOcoorenciasByIdMonitor = async () => {…
    };

    return {
        registrarOcorrencia,
        getOcoorenciasByIdMonitor,
        ocorrencias,
        loading,
        empty,
    };
};
```

Fonte: Do autor (2023)

Neste *hook* anterior são definidas as funções e objetos que foram utilizadas na estrutura da tela de ocorrências, como exemplo a função “registrarOcorrência” e o objeto de “ocorrências”. Sempre que o *hooks* é executado ele inicializa os *useStates* e o *useEffect*, onde o *useState* é responsável por gerenciar o estado de um objeto e o *useEffect* é responsável por disparar uma função sempre que houver uma alteração em uma de suas dependências que são definidas dentro das chaves. Com isso, sempre que o useState de ocorrências sofre alteração, é disparado o *useEffect* que vai chamar a função *getOcorrenciaByIdMonitor*. Esta preenche o *useState* de ocorrência com o retorno da função obtida, e o *useState* (com o valor *empty*) recebe um valor *true* ou *false*, indicando se existem ocorrências ou não. No caso do *useState* receber um valor *true*, a tela que utiliza esse *hooks* é atualizada automaticamente exibindo a informação para o usuário de que não existem ocorrências ou, para o caso de *empty* receber *false*, será exibida as ocorrências.

A figura 26 apresenta a estrutura de pasta para definições dos *hooks*.

Figura 26 - Estrutura de pasta dos hooks.

Fonte: Do autor (2023)

5.2.8 Pages

O termo *pages* é comumente usados para se referir às telas ou páginas individuais do aplicativo. Cada página contém um ou mais componentes que compõem a interface do usuário.

A figura 27 ilustra como é a estrutura da page de *home* e como os componentes são utilizados dentro dela.

Figura 27 - Estrutura da tela Home separada em componentes.

```
export default function Home({ navigation }) {
    return (
        <>
            <HeaderMenu
                onNavigation={() => navigation.openDrawer()}
                title="Meu Lab UFES" />
            <Container>
                <Imagem
                    source={require('../../../assets/icon.png')}⋯
                    }}
                />
                <Line>
                    <ButtonMenu
                    icon="clock-outline" ⋯
                    navigate={() => navigation.navigate('HorariosMonitores')} />
                    <ButtonMenu
                    icon="monitor-star" ⋯
                    navigate={() => navigation.navigate('SejaMonitor')} />
                </Line>
                <Line>
                    <ButtonMenu
                    icon="calendar" ⋯
                    navigate={() => navigation.navigate('ReservaSala')} />
                    <ButtonMenu
                    icon="book-information-variant"⋯
                      navigate={() => navigation.navigate('RegulamentoUtilizacao')} />
                </Line>
            </Container>
        </>
    );
}
```

Fonte: Do autor (2023)

As *pages* são estruturadas utilizando os componentes já criados ou folhas de estilos próprias. Para a page de *Home*, que é a tela inicial do sistema, foi utilizado o componente de *header* que contém os menus do *app* e os componentes de botões que são utilizados exclusivamente na *Home,* mas que também são criados dentro do módulo de componentes para manter uma melhor estrutura do projeto. Dentro dos *botes* e do *header* é definido a ação ao clicar e qual o seu ícone e título.

5.2.9 Routes

A figura 28 retrata como são definidas as rotas para um usuário que já está autenticado no aplicativo.

Figura 28 - Estrutura das Routes para usuário autenticados.

```
export default function Routes() {
    const Stack = createNativeStackNavigator();
    const Drawer = createDrawerNavigator();

    function DrawerNavigator() {…
    }

    return (
        <Stack.Navigator
            initialRouteName="Home"…
            }}>
            <Stack.Screen name="Home" component={DrawerNavigator} />
            <Stack.Screen name="SejaMonitor" component={SejaMonitor} />
            <Stack.Screen name="ReservaSala" component={ReservaSala} />
            <Stack.Screen name="RegulamentoUtilizacao" component={RegulamentoUtilizacao} />
            <Stack.Screen name="SuporteTecnico" component={SuporteTecnico} />
            <Stack.Screen name="Infraestrutura" component={Infrarestrutura} />
            <Stack.Screen name="Contato" component={Contato} />
            <Stack.Screen name="UsoLab05" component={UsoLabCinco} />
            <Stack.Screen name="SolicitarSuporte" component={SoliciarSuporte} />
            <Stack.Screen name="Solicitacoes" component={Solicitacoes} />
            <Stack.Screen name="RegistraOcorrencia" component={RegistraOcorrencia} />
            <Stack.Screen name="RegistrosDeOcorrencia" component={RegistrosDeOcorrencia} />
            <Stack.Screen name="MonitoramentoDeOcorrencia" component={MonitoramentoDeOcorrencia} />
            <Stack.Screen name="MonitoramentoDeSuporte" component={MonitoramentoDeSuporte} />
            <Stack.Screen name="DetalhesSuporte" component={DetalhesSuporte} />
            <Stack.Screen name="DetalhesOcorrencia" component={DetalhesOcorrencia} />
            <Stack.Screen name="ListaMonitores" component={ListaMonitores} />
            <Stack.Screen name="DetalhesMonitor" component={DetalhesMonitor} />
            <Stack.Screen name="HorariosMonitores" component={HorariosMonitores} />
            <Stack.Screen name="RegistraNotificacao" component={RegistraNotificacao} />
            <Stack.Screen name="Notificacoes" component={Notificacoes} />
            <Stack.Screen name="SolicitarCartao" component={SolicitarCartao} />
            <Stack.Screen name="SolicitacaoReserva" component={SolicitacaoReserva} />
            <Stack.Screen name="DetalhesReserva" component={DetalhesReserva} />
        </Stack.Navigator>
    );
}
```

Fonte: Do autor (2023)

As *routes* são definidas dentro das *stacks* que são ferramentas da biblioteca *navigator*. Elas são definidas utilizando como parâmetro o *name* é o *component* na qual se refere a tela que será renderizada.

5.2.10 Services

O *services* é o módulo que contém lógica de negócios reutilizável que pode ser compartilhado entre diferentes componentes. Esses *services* podem ser usados para gerenciar a lógica de dados, chamar APIs externas, processar dados, gerenciar estados e muito mais.

A figura 29 retrata como é definida a *services* para reserva de sala onde é feita a comunicação com o *backend* do sistema.

Figura 29 - Definições das api's para a services de reserva do aplicativo.

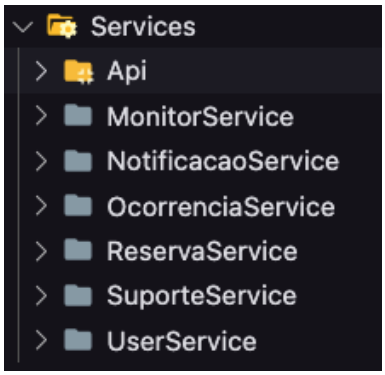
```
import api from '_Services/Api';
const urlReserva = '/reservation';

export const getReservaAtiva = async () => {
    return await api.get(urlReserva);
};

export const marcarComoEntregue = async (idReserva) => {
    return await api.put(`${urlReserva}/delivered/${idReserva}`);
};

export const marcarComoDevolvido = async (idReserva) => {
    return await api.put(`${urlReserva}/returned/${idReserva}`);
};

export const registrarReserva = async (suporte) => {
    return await api.post(urlReserva, suporte);
};
```

Fonte: Do autor (2023)

Dentro da *services*, é desenvolvida a função que será chamada quando for necessário realizar uma requisição ao *backend*. Nessa função, é utilizada uma biblioteca responsável por controlar todas as requisições do sistema. Essa biblioteca oferece algumas funções que são usadas para definir qual será o método HTTP. Após definir o método, foi necessário passar alguns parâmetros, como o caminho que será chamado e, se necessário, o parâmetro de rota e o corpo da requisição. O corpo da requisição é composto por um objeto ou *array* de objetos que serão enviados para serem salvos, atualizados, consultados ou deletados no banco de dados.

A figura 30 mostra como é a estrutura da pasta *service* onde são pastas separadas por módulos.

Figura 30 - Estrutura da pasta service.



Fonte: Do autor (2023)

5.2.11 Utils

A figura 31 ilustra como é definida uma constante que contém valores que são usados em todos os campos de *selectBox* do aplicativo.

Figura 31 - Variável utilitária para definição de itens em um selectBox.

```
export const simNaoValues = [
    {
        label: 'Sim',
        value: 1,
        key: 1,
    },
    {
        label: 'Não',
        value: 0,
        key: 0,
    },
];
```

Fonte: Do autor (2023)

5.2 Git hub

Para as entregas e publicações do sistema foi utilizada a *branch main* que é onde vai estar todo código da última entrega feita. Para essa ramificação também foi aplicado os mesmos teste da *branch develop* com a adição do teste de sistema, a fim de garantir que antes de construir a versão final da entrega e publicar as alterações o sistema possa estar sem falhas relacionadas a código.

A ramificação auxiliar desempenham um papel importante no controle de versões, porém, algumas delas não se fazem necessárias em sistemas como este projeto, por se tratar de um desenvolvedor somente, que no caso é a ramificação *release*, e *feature* (HADLER, 2018). A função que essas duas ramificações atuam foram desempenhadas nas ramificações principais (*main e develop*). Para as demais ramificações terão as seguintes funções:

- Ramificação *hotfix*: Foi utilizada quando teve algum bug na ramificação principal e que está em utilização e esse bug atrapalhar no funcionamento do sistema.
- Ramificação *bugfix*: Teve comportamento semelhante ao *hotfix,* porém foi utilizada sempre que identificado um bug na ramificação principal que não atrapalhe a utilização do sistema ou na ramificação develop que está em desenvolvimento.

5.3 Banco de dados

Para realizar as funções desse projeto não foi necessário um banco de dados com alta performance em tempo de consulta e nem um banco com grande capacidade de tráfego em consultas. Logo, foi utilizado o POSTGRESQL (POSTGRESQL, 2023), que junto com o SGBD DBeaver (DBEAVER, 2023), proporcionaram uma qualidade satisfatória e velocidade em questão de banco de dados, e um gerenciamento dos dados consistente contido no próprio banco.

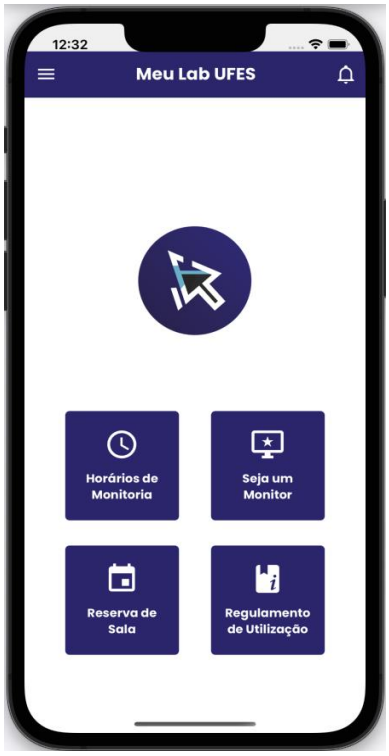
5.4 Interface

As interfaces do sistema foram desenvolvidas para ter uma visualização limpa sem muitas “poluições” visuais, mas que fornece ao usuário todas as informações que são necessárias para que ele possa entender todo o fluxo do sistema e utilizar todas as funções que o sistema disponibiliza.

A Figura 32 apresenta a tela inicial do sistema após o usuário estar autenticado no mesmo. A tela é o menu principal onde o usuário será enviado para as demais funcionalidades do sistema, e foi desenvolvida para que o usuário possa ter acesso

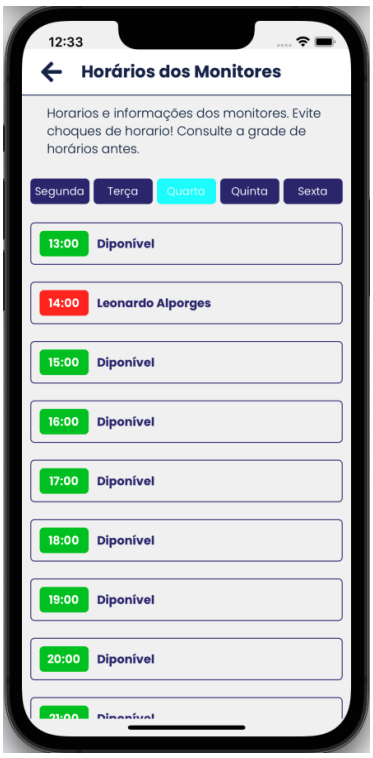
de forma rápida às funcionalidades que são consideras essenciais do aplicativo. Tais funcionalidades são componentes de botão genérico, porém cada uma tem uma implementação específica ao clicar neles. Já a figura 33 exibe uma das telas principais do aplicativo, onde o usuário tem acesso aos horários de monitoria disponíveis e assim será possível saber qual é o melhor momento para ir aos laboratórios. As demais interfaces do sistema estão disponíveis no apêndice

Figura 32 - Tela inicial do aplicativo após usuário logado.



Fonte : Do autor.

Figura 33 - Tela de visualização dos horários dos monitores.



Fonte: Do autor.

5.5 Ciclo de trabalho - Implementações

Como na seção 4.1, o sistema foi desenvolvido utilizando a metodologia de desenvolvimento incremental, onde Sommerville (2011) e Pressman (2011) compartilham da mesma ideia de que cada incremento a ser desenvolvido é necessário que o desenvolvimento passe por todas as etapas previstas até a conclusão do mesmo, antes de iniciar as demais implementações, que serão novos incrementos.

A tabela 4 apresenta as primeiras tarefas elaboradas para iniciar o desenvolvimento do sistema. Esta foi elaborada definindo um título para identificar a tarefa, seguido de uma breve descrição, módulo no qual pertence sendo cada um deles referente a um ator ou funcionalidade do sistema e em qual ambiente será executada podendo ser *back* para a parte que se diz respeito a aplicação web e *front* que é a parte de interface com o usuário. As demais tarefas que foram essenciais para o desenvolvimento do sistema são apresentadas no apêndice D.

Tabela 4 - Backlog de tarefas iniciais do sistema.

Backlog	Descrição	Módulo	Ambiente
Estrutura inicial do app	Criar o app utilizando o react native	Configuração	Front
Estrutura inicial do backend	Criar o backend utilizando o node.js	Configuração	Back
Integrar os serviços do firebase no app	Integrar os serviços do firebase no app para consumo de storages e controle de variáveis remotas	Configuração	Front
Adicionar push notification no app	Integrar os serviços do expo push notification para registrar e salvar o id dos devices dos usuários	Configuração	Back
Gerar aplicativo na google play	Criar uma hospedagem do aplicativo no google play store	Configuração	Front
Hospedar o backend em um servidor	Adicionar o backend no servidor da UFES para desenvolvimento	Configuração	Back
Criar banco de dados de produção e desenvolvimento	Criar o banco de dados que será usado em produção e em desenvolvimento	Configuração	Back
Criar primeiros protótipos do app para avaliação	Criar os protótipos do app para serem enviados ao orientador para aprovação	Configuração	Front
Criar tela de login	Desenvolver a tela de login do app para que usuário possa se autenticar e usar os serviços	Usuário	Front

Fonte: Do autor.

6. Garantia da qualidade e processo de teste

A garantia da qualidade e o processo de teste são cruciais para o desenvolvimento de softwares de alta qualidade, pois a garantia da qualidade

possibilita a conformidade do software com os requisitos e a satisfação do cliente, enquanto o processo de teste é responsável por identificar e corrigir possíveis problemas e defeitos antes da entrega. O teste é realizado para que o sistema não apresente problemas em ambiente de produção (SOMMERVILLE, 2011).

Os testes para esse sistema foram elaborados pelo autor, sendo que a execução dos Testes de Funcionalidades e Testes de Sistema foram realizados pelo mesmo. Para os Testes de Usabilidade, os agentes de execução foram separados em 5 diferentes públicos-alvo, como ilustrado na tabela 5.

Tabela 5 - Públicos-alvo dos testes de usabilidade.

Públicos alvos	Descrição
A1	Alunos do curso de sistema de informação ou computação
A2	Alunos de outros cursos dentro da ufes
A3	Alunos que são ou já foram monitores dos laboratórios
C1	Coordenador ou professor do departamento de computação
P1	Policiais que fazem o controle de chaves das dependências da UFES

Fonte: Do autor (2023)

6.1 Testes funcionais

Os Testes Funcionais foram executados todos utilizando aparelhos com o sistema operacional Android, no modo físico (usando aparelhos de fato) e emuladores (softwares que simulam um aparelho) para garantir que demais dispositivos com sistemas operacionais diferentes sejam examinados, bem como resoluções e *DPI*[1]. O objetivo é avaliar que o sistema funcione igual para todos e sem problemas de *layout* (responsividade) ou funcionalidade.

Para esses testes, foi elaborado e aplicado o roteiro que é mostrado na tabela 6. Esse roteiro conta com testes em todos os requisitos funcionais levantados para esse projeto, os quais são separados por seção, referência, nome do teste, descrição breve, dados de entrada e resultados esperados.

[1] DPI refere-se ao número de pontos contido dentro de uma polegada de uma imagem (SONY, 2015).

Tabela 6 - Roteiro de teste funcional do sistema

Seção	Referência	Nome teste	Descrição	Dados de entrada	Resultado esperado
Login	TF - 01	Logar no sistema com sucesso	Verificar se o login de usuário está funcionando corretamente	Login de um usuário já existente.	Usuário entrar com sucesso no aplicativo.
	TF - 02	Logar como vigilante no sistema com sucesso	Verificar se o login de usuário está funcionando corretamente	Login de um usuário já existente.	Usuário entrar com sucesso no aplicativo.
	TF - 03	Login errado	Verificar se o sistema vai impedir o login de um usuario invalido	Login de usuario com senha ou número de matricula errados	Usuário não consegue entrar no app e é apresentado um modal de erro
Usuário	TF - 04	Enviar uma solicitação de suporte	Verificar se a solicitação de suporte vai ser enviada para o coordenador	Dados solicitados na tela de "Solicitar Suporte"	Solicitação salva no banco e modal de sucesso apresentado para usuário
	TF - 05	Ver a solicitação de suporte feita	Verificar se será exibida a solicitação de suporte na listagem após ter realizado uma.	Nenhum	Exibir todas solicitações em aberto feito pelo usuário
	TF - 06	Verificar os detalhes da solicitação feita	Verificar se será exibido corretamente os detalhes solicitação de suporte na listagem após ter realizado uma.	Nenhum	Exibir todos detalhes da solicitações em aberto feito pelo usuário

Fonte: Do autor (2023).

A partir desse roteiro pode-se registrar vários erros e bugs encontrados ao longo do teste, esses erros foram sendo registrados através da plataforma Google Docs (GOOGLE,2023) onde foi possível anexar imagens e vídeos para deixar claro onde aconteceram os erros e o que os causou.

A imagem 34 mostra como foi feito o registro dos *bugs* e erros, bem como a ação do autor referente ao que foi registrado. Esse registro foi feito, em grande parte, por um aluno do curso de Sistema de Informação da UFES - campus Alegre, onde o mesmo testava todos os cenários mapeados em todo pré-lançamento de correção e implementação do sistema, que também pode ser considerado como um Teste Regressivo (DEVMEDIA,2011). Além de testar o roteiro proposto, através deste foi possível pegar *bugs* que estavam escondidos em pequenos fluxos de navegação no sistema, além também de serem feitas algumas sugestões relacionadas a usabilidade do sistema, incrementando a qualidade do aplicativo, para esse quesito em questão. No apêndice E são exibidos todos os registros feitos.

Figura 34 - Registro de teste do sistema como coordenador.

----- TESTE DE SISTEMA COMO COORDENADOR ----

AUTOR : GIULIANO PRADO MORAES

1-Nas informações de uma Ocorrência, deveria ter mais um dado que é o Status da Ocorrência, mostrando em que pé ela está.
R: Já tem esse campo. Ele fica em no final da tela como "Descrição da solução"

2- Em Suportes, quando mostra a lista deles, deveria ter uma legenda no rodapé mostrando o que cada cor representa, O usuário não tem que adivinhar, principalmente se mudamos o coordenador
R: Adicionado.

3- Quando se faz uma reserva, ao mudar seu status, a cor indicativa da reserva fica sempre azul, Temos que rever as cores indicativas das ocorrências/reservas
OBS: Vermelho sempre indica atenção, ou algo a ser solucionado, então deveria ser a cor que aparece para os itens abertos e não atendidos. O amarelo para análise e verde para atendido estão ok,Mas temos que diferenciar a cor do rejeitado ou não atendido com o aberto
R: Cores adicionadas. Esquema de cor abaixo :

3,1- O vermelho deve ser a cor do em aberto, no lugar do azul
R: item 3

Fonte: Do autor (2023).

6.2 Testes de sistema

Os Teste de Sistema (assim como os Testes Funcionais) também foram testados em dispositivos físicos e emuladores para garantir que, quando estiver rodando o ambiente de produção, não aconteçam erros inesperados referentes a sistemas e *layout* fora do padrão desejado.

O Teste de Sistema também segue o mesmo roteiro do Teste Funcional definido na tabela 6, porém em ambiente no qual o usuário final terá acesso.

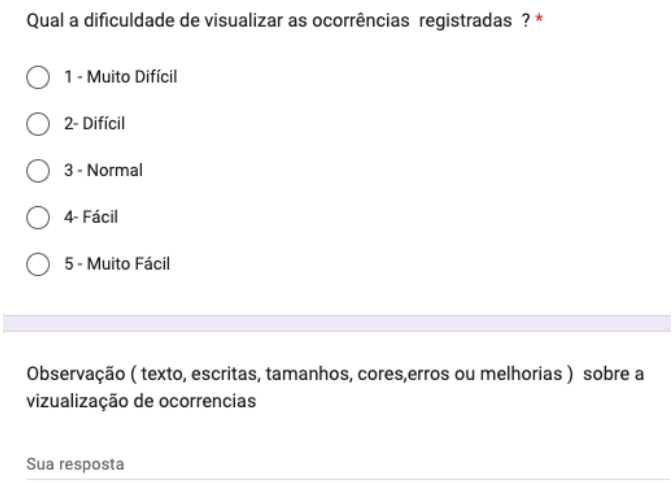
6.3 Testes de usabilidade

Os Testes de Usabilidade foram totalmente direcionados aos usuários efetivos do sistema, para que através da utilização de versões em pré-produção do aplicativo, possa-se obter *feedbacks* sobre *layout*, funcionalidades e acessibilidade do aplicativo, e assim corrigir e implementar novas funcionalidades para garantir que o usuário tenha a melhor experiência possível utilizando o aplicativo.

Para os Testes de Usabilidade, foram elaborados formulários onde os autores têm 5 opções, entre as quais: muito difícil, difícil, normal, fácil e muito fácil. Visa-se, portanto, a seleção de uma das perguntas sobre a dificuldade de utilizar algumas telas e funcionalidades e avaliar com uma das respostas propostas.

Na figura 35 é mostrada uma das várias perguntas do teste de usabilidade 04 para os autores C1, as demais perguntas para esse teste e também os demais formulários de teste elaborados são apresentados no apêndice E.

Figura 35 - Formulário sobre usabilidade do sistema.

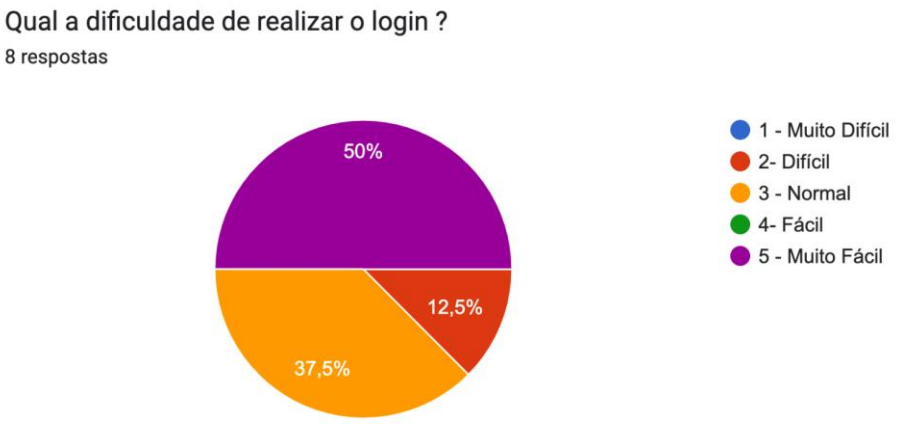


Fonte: Do autor (2023).

Nos Testes de Usabilidade objetivou-se obter *feedbacks* dos testadores para que fosse possível melhorar ou alterar alguns layouts, ajustar o fluxo do sistema e também encontrar inconsistências que estivessem relacionados ao fluxo que o usuário deveria seguir no sistema, baseado no perfil de usuário (ator) do aplicativo. Desta forma, pretendeu-se verificar se a mesma facilidade de utilização encontrada por um aluno dos cursos de Computação também fosse encontrada por um aluno dos cursos de Ciências Agrárias e afins, os quais não tem tanto contato com tecnologia

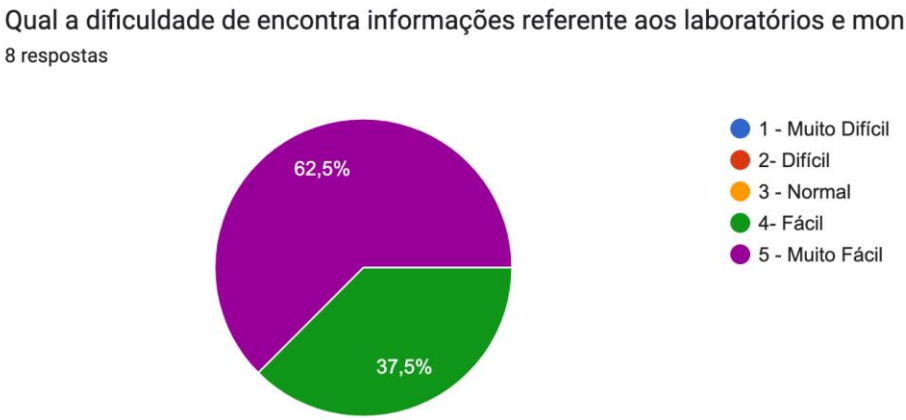
computacional mais avançada na faculdade. Nas imagens 38, 39, 40 e 41 são exibidos os gráficos relacionados à usabilidade do sistema, em partes que são consideradas de maior importância no sistema, relacionadas a funcionalidades fins, tais como a autenticação de usuário e a visualização de horário de monitoria. As imagens 36 e 37 são relacionadas às perguntas “qual a dificuldade de realizar o login ?” e “Qual a dificuldade de encontrar informações referentes aos laboratórios e monitoria?” e representa os atores que são dos cursos de Computação e afins.

Figura 36 - Gráfico de respostas de alunos de computação sobre o login.



Fonte: Do autor (2023).

Figura 37 - Gráfico de respostas de alunos de computação sobre acesso às informações de laboratório.



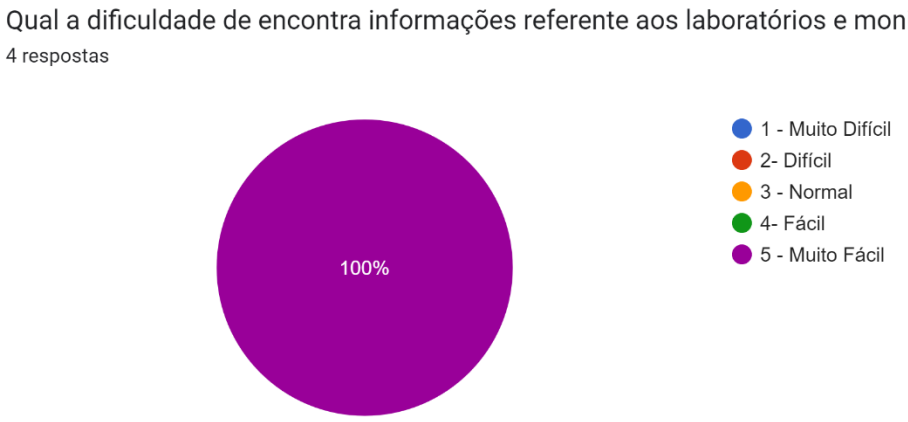
Fonte: Do autor (2023).

Já nas imagens 38 e ,39 estão relacionadas às mesmas perguntas anteriores e representa os autores que são de outros cursos não relacionados com a computação.

Figura 38 - Gráfico de respostas de alunos de outros cursos sobre o login

Fonte: Do autor (2023).

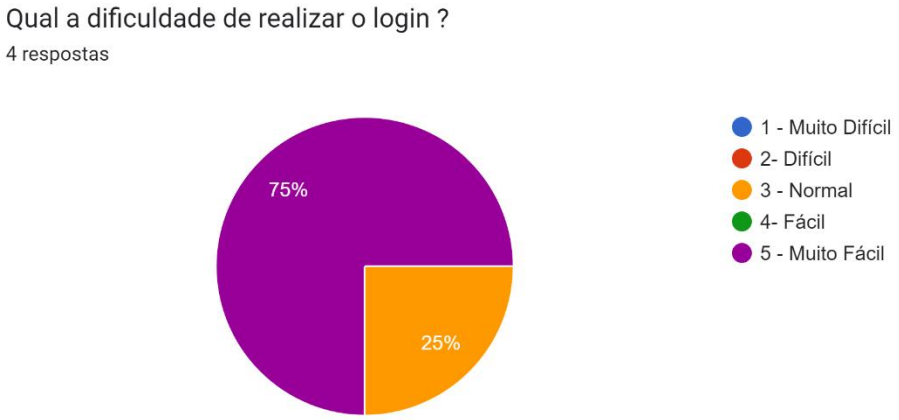
Figura 39 - Gráfico de respostas de alunos de outros cursos sobre acesso a informações de laboratório.



Fonte: Do autor (2023)

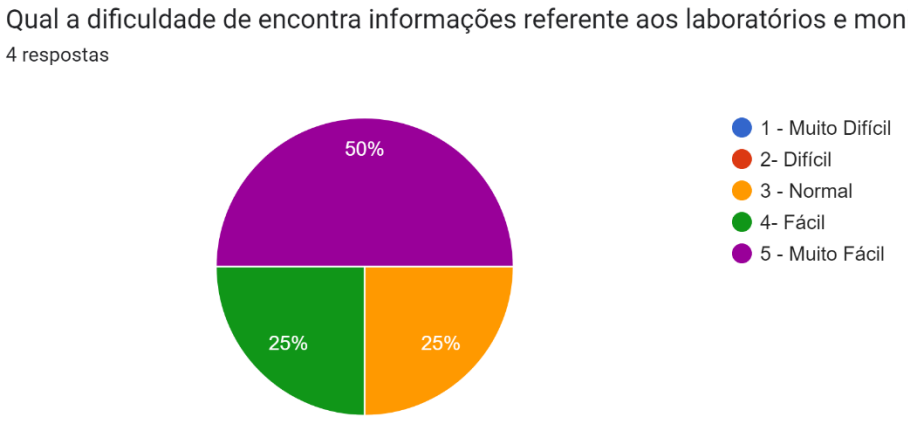
Já as imagens 40 e 41 são relacionadas às mesmas perguntas e representam os autores que são ou que já foram monitores dos laboratórios de informática.

Figura 40 - Gráfico representando respostas de alunos que são ou já foram monitores dos laboratórios, sobre o login.



Fonte: Do autor

Figura 41 - Gráfico representando respostas de alunos que são ou já foram monitores dos laboratórios, sobre acesso a informações de laboratório.



Fonte: Do autor

6.4 Análise dos testes

Como já exposto anteriormente, os testes garantem a qualidade, performance e usabilidade do sistema. Com os Testes Funcionais e Testes de Sistema realizados, foi possível aprimorar alguns fluxos de usabilidade, além da possibilidade de encontrar e resolver problemas que só aconteceriam com o usuário final que, por exemplo, pode

ser citado a diferença no nome propriedade onde fica disponível o número de matrícula do professor e número de matrícula de aluno após o retorno da requisição de login com a UFES, onde só foi possível encontrar o erro após um teste em produção que descobriu-se que o nome da propriedade não era o mesmo para os dois tipos de autores. A imagem 42 mostra o trecho que código onde foi necessária a correção para que o *registration_number* que se refere ao número de matrícula, possa armazenar tanto o número de matrícula do aluno quanto do professor.

Figura 42 - Código que mostra como é feita atribuição de número de matrícula a um usuário

```
const {
  username = cn,
  functionAtUfes = eduPersonScopedAffiliation,
  institutional_email = mail,
  registration_number = userUFES.matGrad ?? userUFES.matSiape,
  course = cursoGrad ?? "",
} = userUFES;
```

Fonte: Do autor

7. Considerações finais e trabalhos futuros

Ao finalizar este trabalho foi concluído também o desenvolvimento da aplicação para o gerenciamento dos laboratórios, dentro do prazo estabelecido. Essa aplicação, que foi hospedada nos servidores da instituição, proporciona aos docentes e discentes um acesso mais ágil e facilitado às informações e ao gerenciamento dos laboratórios. Os usuários em geral que possuem dispositivos móveis com sistema operacional Android agora podem baixar o aplicativo e utilizá-lo fora das dependências do campus para verificar os horários em que haverá um monitor disponível no laboratório para auxiliá-los, além de se candidatar e se para o processo de monitoria dos laboratórios, tudo em um único sistema. Além disso, eles podem contar com notificações informativas em tempo real sobre os laboratórios ou outras questões do campus relacionadas, por meio de notificações enviadas pelo coordenador do laboratório.

O desenvolvimento deste trabalho possui um grande valor, uma vez que permitiu a exploração do processo de criação de um sistema desde o início como elaboração de arquitetura até o fim com a hospedagem do sistema utilizando os servidores da UFES, bem como a seleção de ferramentas, métodos, técnicas e

tecnologias para sua conclusão. Isso possibilitou um aprofundamento em várias áreas que foram abordadas ao longo do período de graduação como a gestão de qualidade de software, engenharia de requisitos de software, padrões de projeto de software e banco de dados.

Foram utilizadas ferramentas que são comumente empregadas no mercado de trabalho atual, o que foi benéfico para o autor, que já estava inserido nesse ambiente, pois permitiu a integração dos conhecimentos adquiridos no mercado, os quais não são abordados durante o curso, com os conhecimentos fundamentais adquiridos durante a graduação, que são essenciais para compreender a criação de um sistema como um todo.

Como resultado, foi desenvolvido um sistema que emprega técnicas, *frameworks* e linguagens de programação de vanguarda, juntamente com os princípios de gerenciamento, controle de implementação, estudo de casos e documentação aprendidos no curso de formação acadêmica. Isso resultou em um sistema simples, robusto e passível de implementações e manutenções futuras.

7.1 Melhorias e trabalhos futuros

O gerenciamento dos laboratórios e a disponibilização de informações são processos que dependem de intercâmbios significativos de informações para garantir um sistema que atenda efetivamente a todos os seus usuários. Atualmente, para se candidatar a uma monitoria voluntária e adicionar monitores no sistema é necessário que o formulário de candidatura seja preenchido em uma *webview* que é uma tela de navegador disponibilizada dentro do próprio aplicativo, e nela contém um formulário de inscrição para monitoria voluntária do *Google Forms* (GOOGLE, 2023). Nesse sentido, sugere-se uma reformulação no módulo de Cadastro de Monitores Voluntários e no módulo de Adição de Novos Monitores no sistema, a fim de aprimorar a clareza na troca de informações entre as partes envolvidas, e permitir com que todos os dados e informações sobre o Gerenciamento dos Laboratórios fiquem somente no aplicativo, não sendo necessário utilizar *webview* com formulários e preenchimentos de planilhas para registro dos monitores escolhidos.

7.1.1 Informativos gerais do campus

O aplicativo poderia ser extensível a fim de ampliar a quantidade de módulos disponíveis, os quais poderiam fornecer informações sobre outros setores e serviços da UFES. Logo, sugere-se a implementação de outro módulo que facilite o acesso a informações gerais e essenciais sobre o campus, tais como: restaurante universitário, quadra de esportes, horário de funcionamento do campus e localização dos setores da UFES. Esses módulos podem ser descritos como um acesso rápido através de um menu do aplicativo, o qual disponibiliza uma tela contendo basicamente links para outros apps e sites ligados à UFES, sobretudo aqueles voltados ao campus de Alegre.

7.1.2 Integrações com outros sistemas

Como sugestão para fomentar a interação entre os sistemas já desenvolvidos por estudantes da UFES, propõe-se a implementação de um módulo que promova a divulgação desses outros sistemas assim como sugerido na seção 7.1.1, porém esse seria voltado exclusivamente para projetos os discentes, permitindo que novos alunos (e aqueles que ainda não têm conhecimento) possam conhecer e explorar outros sistemas criados, mas que não receberam ampla divulgação no campus.

7.1.3 Disponibilização para sistemas operacionais IOS

Para que o sistema possa ser utilizado por todos os estudantes da UFES, propõe-se a disponibilização do sistema para a plataforma IOS, pois como é uma plataforma onde é necessário um pagamento de uma conta de desenvolvedor anual e por não ser disponibilizado verba para custear essa conta em projetos de trabalho de conclusão de curso, o mesmo não pode ser disponibilizado para este sistema operacional móvel.

REFERÊNCIAS BIBLIOGRÁFICAS

ALURA, **Git Flow: entenda o que é, como e quando utilizar.** Disponível em: <http://www.alura.com.br/artigos/git-flow-o-que-e-como-quando-utilizar>. Acesso em: 20 jan. 2023.

AWS**. O que é modelagem de dados?.** Disponível em: <https://aws.amazon.com/pt/what-is/data-modeling/>. Acesso em: 10 nov. 2022.

BARRO, B. B. **O Que São Frameworks e Quais os Mais Utilizados**. Disponível em: <https://www.hostinger.com.br/tutoriais/frameworks>. Acesso em: 24 jan. 2023.

BRIA, Francesca; MOROZOV, Evgeny. **A cidade inteligente: tecnologias urbanas e democracia**. Ubu Editora, 2020.

CAMPUS-UNICAMP, S. **Smart Campus - Unicamp**. Disponível em: <https://smartcampus.prefeitura.unicamp.br/>. Acesso em: 20 jan. 2023.

Ciclo de Vida Interativo e Incremental.Wiki IFSC Campus São José.,2006. Disponível em: <https://wiki.sj.ifsc.edu.br/wiki/index.php/Ciclo_de_Vida_Iterativo_e_Incremental>. Acesso em: 03 de Dezembro de 2022.

Claro, José Alberto. (2013). **Livro "Sistemas de Informações Gerenciais"**, 1 Edição, 2013, São Paulo.

DBEAVER. **Ferramenta de banco de dados universal.** Disponível em: <https://dbeaver.io>. Acesso em: 20 jan. 2023.

DEVMEDIA. **Teste de Regressão.** Disponível em: https://www.devmedia.com.br/teste-de-regressao/23038. Acesso em: 12 fevereiro de 2023.

DE LEMOS, M. F. et al. **APLICABILIDADE DA ARQUITETURA MVC EM UMA APLICAÇÃO WEB(WebApps)**. RE3C - Revista Eletrônica Científica de Ciência da Computação, v. 8, n. 1, 2013.

EXPRESS.JS. **Express Routing**. [S.l.], 2023. Disponível em: https://expressjs.com/pt-br/guide/routing.html. Acesso em: 21 abr. 2023.

FERREIRA, F. H. C.; ARAÚJO, R. M. **Campus Inteligentes: Conceitos, aplicações, tecnologias e desafios**. **RelaTe-DIA**, 2018.

FERREIRA, Kátia Gomes; DE CURSO, Monografia de Final; DA SILVA, Clarindo Isaías Pereira. Teste de usabilidade. **Monografia de Final de Curso: Especialização em Informática, Universidade de Minas Gerais. Belo Horizonte, Brasil**, 2002.

FLANAGAN, D. **JavaScript: O Guia Definitivo.** [s.l.] Bookman Editora, [s.d.]. 2013.

GITHUB. **Plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git**. Disponível em: <https://github.com>. Acesso em: 20 jan. 2023.

GOOGLE. **Google Forms**. Disponível em: https://docs.google.com/forms. Acesso em: 11 mai. 2023.

GOOGLE. **Google Documentos**. Disponível em: https://docs.google.com/document. Acesso em: 11 mai. 2023.

GOTARDO, R. **Linguagem de programação**. Rio de Janeiro: Seses. [s.l: s.n.]. 2015.

GUEDES, G. T. A. **UML 2 - Uma Abordagem Prática**. [s.l.] Novatec Editora, 2018.

HADLER, M. **Utilizando o fluxo Git Flow**. Disponível em: <https://medium.com/trainingcenter/utilizando-o-fluxo-git-flow-e63d5e0d5e04>. Acesso em: 20 jan. 2023.

HARRISON, C.; DONNELLY, I. A. **A theory of smart cities**. White Paper, IBM Corporation, 2011. Disponível em: https://journals.isss.org/index.php/proceedings55th/article/view/1703/572. Acesso em 20/01/2023.

HEUSER, Carlos Alberto. **Projeto de banco de dados: Volume 4 da Série Livros didáticos informática UFRGS**. Bookman Editora, 2009.

IDS. **Passos para ser uma cidade inteligente**. Disponível em: <https://ids.inf.br/cidade-inteligente-cidade-digital>. Acesso em: 02 jan. 2023.

JACOSKI, C. A.; LISSANDRO, M. **CAMPUS INTELIGENTE: UM NOVO PARADIGMA NA ORGANIZAÇÃO DAS UNIVERSIDADES**. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/190551/101_00104.pdf?sequence=1&isAllowed=y>. Acesso em: 14 out. 2022.

LARMAN, C. (2002) **Applying UML and Patterns – An Introduction to Object Oriented Analysis and Design and the Unified Process**, Prentice-Hall Inc.

LUCIANO, J.; WALLISON, ;.; ALVES, J. B. **PADRÃO DE ARQUITETURA MVC: MODEL-VIEW-CONTROLLER**. Disponível em: <https://unifafibe.com.br/revistasonline/arquivos/revistaepeqfafibe/sumario/20/16112011142249.pdf>. Acesso em: 24 jan. 2023.

MATTOS, A. C. M. **SISTEMAS DE INFORMAÇÃO**. [s.l.] Saraiva Educação S.A, 2017.

MEIRELLES, F. S. **Pesquisa do Uso da TI - Tecnologia de Informação nas Empresas**, FGVcia. 2022.

NEVES, Ana Régia de M.; SARMANHO, Kaê U.; JR., Francisco C. Nascimento; MEIGUINS, Bianchi S.. **Iniciativa Smart Campus: um estudo de caso em progresso na Universidade Federal do Pará**. In: WORKSHOP DE COMPUTAÇÃO URBANA (COURB), 1. , 2017, Belém. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2017 . ISSN 2595-2706.

NITAHARA, A. **Estudo mostra que pandemia intensificou uso das tecnologias digitais**. Disponível em: <https://agenciabrasil.ebc.com.br/geral/noticia/2021-11/estudo-mostra-que-pandemia-intensificou-uso-das-tecnologias-digitais>. Acesso em: 14 out. 2022.

NETO, A, C, D. **Introdução a teste de Software**. Engenharia de Software Magazine, v. 1, 2007.

NODE.JS. **Software de código aberto, multiplataforma baseado no interpretador V8 do Google**. Disponível em: <https://nodejs.org/pt-br/>. Acesso em: 24 jan. 2023.

OLIVEIRA, J. C.; CHALUB, A. (EDS.). **Pandemia acelera o uso de serviços públicos digitais**. [s.l.] Agência Câmara de Notícias, 2021.

POSTGRESQL. **The World's Most Advanced Open Source Relational Database**. Disponível em: <www.postgresql.org>. Acesso em: 20 jan. 2023.

PRESSMAN, R. S. **Engenharia de Software UMA ABORDAGEM PROFISSIONAL**. [s.l.] AMGH Editora Ltda, 2016.

RAMAKRISHNAN, R.; GEHRKE, J. Sistemas de gerenciamento de banco de dados - 3.ed. [s.l.] AMGH Editora, 2008.

REACTJS. **Introdução aos Hooks**. [S.l.], 2021. Disponível em: https://pt-br.legacy.reactjs.org/docs/hooks-intro.html. Acesso em: 21 abr. 2023.

SODRÉ, R. **Celular revoluciona o jeito de estudar nas universidades**. Disponível em: <https://www.otempo.com.br/brasil/celular-revoluciona-o-jeito-de-estudar-nas-universidades-1.928577>. Acesso em: 14 out. 2022.

Silva, A; Videira,C. (2001). UML, Metodologias e Ferramentas CASE. Disponível em: <https://www.researchgate.net/publication/309428520_UML_Metodologias_e_Ferramentas_CASE>. Acesso em: 14 out. 2022.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

STROKLABS. **Como organizar e estruturar projetos com Node.js**. [S.l.], 2018. Disponível em: https://medium.com/@stroklabs/como-organizar-e-estruturar-projetos-com-node-js-4845be004899. Acesso em: 21 abr. 2023.

STATISTA**. Most used web frameworks among developers 2022**. Disponível em: <https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/>. Acesso em: 24 jan. 2023.

TAROUCO, L. M. R. et al. I**nternet das Coisas na Educação, trajetória para um campus inteligente**. Anais dos Workshops do VI Congresso Brasileiro de Informática na Educação (CBIE 2017). Anais...Brazilian Computer Society (Sociedade Brasileira de Computação - SBC), 2017.

TOTVS. **Front-end**. [S.l.], 2022. Disponível em: https://www.totvs.com/blog/developers/front-end/. Acesso em: 21 abr. 2023.

TREINAWEB, **O que é o React Native?** Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-o-react-native>. Acesso em: 24 jan. 2023.

UFJF. **UFJF+ Campus Inteligente**. Disponível em: <www.campusinteligente.ufjf.br/>. Acesso em: 20 jan. 2023.

WASHBURN, D. et al. **Helping CIOs understand "smart city" initiatives: defining the smart city, its drivers, and the role of the CIO**. Cambridge, MA: Forrester Research, Inc. , 2010. Disponível em: http://public.dhe.ibm.com/partnerworld/pub/smb/smarterplanet/ forr_help_cios_und

WEISS, M. C.; BERNARDES, R. C.; CONSONI, F. L. **CIDADES INTELIGENTES: casos e perspectivas para as cidades brasileiras**. Revista Tecnológica da Fatec Americana, v. 5, n. 1, p. 01–13, 2017. Disponível em: https://www.fatec.edu.br/revista/index.php/RTecFatecAM/article/view/137. Acesso em: 16 jan. 2023.

APÊNDICE A – Narrativas de caso de uso

UC001 - Realizar login no sistema

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário se autentique no sistema.

Atores: Coordenador, monitor e usuário.

Pré-condições:

Possui matrícula ativa no sistema universitário, pois será utilizado o sistema de autenticação da instituição, incorporado ao aplicativo.

FA1 - Possuir credenciais geradas pelo gestor do aplicativo.

Fluxo normal :

1. [RS] O sistema exibe uma tela com dois campos de entrada.
2. [EV] O autor clica no campo de entrada de número de registro.
3. [EV] O autor digita seu número de registro.
4. [EV] O autor clica no campo de input de senha.
5. [EV] O autor digita sua senha pessoal de acesso aos sistemas da ufes.
6. [EV] O autor clicar em logar.
7. [RS] O sistema faz a autenticação das credenciais do usuário no sistema UFES global
8. [RS] O sistema encaminha o autor para tela inicial do aplicativo.
9. [RS] O sistema salva as informações de login do autor para eventuais funcionalidades.

Fluxo Alternativo :

FA1 - Login para vigilantes

1. [RS] O sistema exibe uma tela com dois campos de entrada.
2. [EV] O autor clica no campo de input de número de registro.
3. [EV] O autor digita seu número de registro criado pelo autor do trabalho.
4. [EV] O autor clica no campo de input de senha.
5. [EV] O autor digita sua senha pessoal criada pelo autor do trabalho.
6. [EV] O autor clicar em logar.
7. [RS] O sistema realiza a autenticação das credenciais do usuário no sistema UFES global
8. [RS] O sistema encaminha o autor para tela inicial do aplicativo.
9. [RS] O sistema salva as informações de login do autor para eventuais funcionalidades.

Pós-condições:

FN - Um usuário é adicionado ao sistema proposto.

FA1 - Um usuário é adicionado ao sistema proposto.

Fluxo de exceção:

FE1 - O ator já fez login anteriormente no aplicativo.

1. [RS] O sistema atualiza suas informações de login.

FE2 - O ator inseriu alguma informação de login errada.

1. [RS] O sistema exibe um modal com a mensagem [MSG001].

Mensagens:

MSG001 - Erro! Informações de login invalida.

UC002 - Solicitar suporte

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa solicitar suporte

Atores: Monitor e usuário.

Pré-condições:

O Autor está autenticado no sistema.

Fluxo normal:

1. [RS] O sistema exibe uma tela com campos para inserir os dados do suporte.
2. [EV] O autor preenche todos os campos no formulário.
3. [EV] O autor clica no botão enviar.
4. [RS] O sistema armazena a solicitação.
5. [RS] O sistema envia uma notificação ao coordenador.
6. [RS] O sistema exibe um modal com a mensagem [MSG001].

Fluxo Alternativo:

Não possui.

Pós-condições:

FN - Uma nova solicitação de suporte é salva no banco de dados.

Fluxo de exceção:

FE1 - O ator não preencheu corretamente os campos de entrada.

1. [RS] O sistema exibe um modal com a mensagem [MSG002].

Mensagens:

MSG001 - Sucesso! Sua solicitação de suporte já foi enviada para o coordenador responsável.

MSG002 - Erro! Verifique se todos os campos estão preenchidos.

UC003 - Visualizar suporte solicitado

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa visualizar os suportes solicitados.

Atores: Monitor e usuário.

Pré-condições:

Usuário está autenticado no sistema.[FA1] [FA2]

FA2 - O autor não possui solicitação de suporte.

Fluxo normal :

1. [RS] O sistema exibe uma tela com uma lista de suportes solicitados.

Fluxo Alternativo:

FA1- Clique em um dos suportes solicitados.

1. [RS] O sistema exibe uma tela com todos os detalhes do suporte solicitado pelo usuário.

FA2- Usuário não possuir solicitações de suporte.

1. [RS] O sistema exibe uma tela com a mensagem [MSG001]

Pós-condições:

Não possui.

Fluxo de exceção:

Não possui.

Mensagens:

MSG001 - Ainda não existe suporte para análise !

UC004 - Gerenciar solicitações de suporte

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o autor possa gerenciar o status das solicitações de suporte

Atores: Coordenador.

Pré-condições:

O autor está autenticado no aplicativo.

FA1,FA2,FA3 - Existir solicitações de suporte.

Fluxo normal:

1. [RS] O sistema exibe uma tela com todas as solicitações de suporte realizadas
2. [EV] O autor clica em uma das solicitações de suporte.
3. [RS] O sistema exibe uma tela com informações detalhadas sobre a solicitação que foi aberta.

Fluxo Alternativo :

FA1 - Alterar status da solicitação de suporte concluída.

1. [EV] O autor clica no campo de entrada para descrição.
2. [EV] O autor clica no botão de concluído.
3. [RS] O sistema salva o novo status.
4. [RS] O sistema envia uma notificação ao usuário solicitante do suporte informando sobre a alteração de status.
5. [RS] O sistema exibe uma tela com a mensagem [MSG001]

FA2 - Alterar status da solicitação de suporte em análise.

1. [EV] O autor clica no campo de entrada para descrição.
2. [EV] O autor clica no botão de “em análise”.
3. [RS] O sistema salva o novo status.
4. [RS] O sistema envia uma notificação ao usuário solicitante do suporte informando sobre a alteração de status.
5. [RS] O sistema exibe uma tela com a mensagem [MSG002]

FA3 - Alterar status da solicitação de suporte para deletada.

1. [EV] O autor clica no campo de entrada para descrição.
2. [EV] O autor clica no botão de deletar.
3. [RS] O sistema salva o novo status.
4. [RS] O sistema envia uma notificação ao usuário solicitante do suporte informando sobre a alteração de status.
5. [RS] O sistema exibe uma tela com a mensagem [MSG003]

Pós-condições:

FA1 - O novo status do suporte é salvo no banco de dados.

FA2 - O novo status do suporte é salvo no banco de dados .

FA3 - O novo status do suporte é salvo no banco de dados .

Fluxo de exceção:

FE1 - O Autor não possui suporte para serem gerenciados.

1. [RS] O sistema exibe uma tela com a mensagem [MSG004]

Mensagens:

MSG001 - Pronto! Agora o usuário responsável será notificado.

MSG002 - O suporte foi colocado em análise e o usuário responsável será notificado.

MSG003 - O suporte foi deletado e o usuário responsável será notificado.

MSG004 - Ainda não existe suporte para análise

UC005 - Realizar reserva de sala

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa solicitar a reserva de uma das salas relacionadas ao DECOMP

Atores: Monitor e usuários.

Pré-condições: Usuário está autenticado no sistema.

Fluxo normal :

1. [RS] O sistema exibe uma tela com campos para inserir os dados da reserva.
2. [EV] O autor preenche todos os campos no formulário.
3. [EV] O autor clica no botão enviar.
4. [RS] O sistema armazena a solicitação.
5. [RS] O sistema envia uma notificação ao coordenador.
6. [RS] O sistema exibe um modal com a mensagem [MSG001]

Fluxo Alternativo:

Não possui.

Pós-condições:

Uma nova solicitação de reserva de sala é salva no banco de dados.

Fluxo de exceção:

FE1 - O ator não preencheu corretamente os campos de entrada.

1. [RS] O sistema exibe um modal com a mensagem [MSG002]

Mensagens:

MSG001 - Sua solicitação de reserva já foi enviada para os responsáveis.

MSG002 - Erro! Verifique se todos os campos estão preenchidos.

UC006 - Visualizar reserva de sala solicitados

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa visualizar as reservas de salas solicitadas.

Atores: Monitor e usuário.

Pré-condições:

Usuário está autenticado no sistema.[FA1] [FA2]

Fluxo normal :

1. [RS] O sistema exibe uma tela com uma lista de reservas de salas solicitadas.

Fluxo Alternativo:

FA1- Clique em uma das reservas solicitadas.

1. [RS] O sistema exibe uma tela com todos os detalhes da reserva de sala solicitada pelo usuário.

FA2- Usuário não possuir solicitações de reserva de sala.

1. [RS] O sistema exibe uma tela com a mensagem [MSG001].

Pós-condições:

Não possui.

Fluxo de exceção:

Não possui.

Mensagens:

MSG001 - Ainda não existem reservas para análise!

UC007 - Gerenciar solicitações de reserva de sala

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o autor possa gerenciar o status das solicitações de reserva de sala

Atores: Coordenador.

Pré-condições:

O autor autenticado no aplicativo. [FA1] [FA2]

Fluxo normal

1. [RS] O sistema exibe uma tela com todas as solicitações de reserva de sala realizadas
2. [EV] O autor clica em uma das solicitações de reserva.
3. [RS] O sistema exibe uma tela com informações detalhadas sobre a reserva que foi aberta.

Fluxo Alternativo :

FA1 - Alterar status da solicitação da reserva para “chave entregue”.

1. [EV] O autor clica no botão de “chave entregue”.
2. [EV] O sistema salva o novo status.

3. [RS] O sistema envia uma notificação ao usuário solicitante da reserva, informando sobre a alteração de status da reserva.
4. [RS] O sistema envia uma notificação ao coordenador, informando sobre a alteração de status da reserva.
5. [RS] O sistema exibe uma tela com a mensagem [MSG001].

FA2 - Alterar status da solicitação da reserva para “chave devolvida”.

1. [EV] O autor clica no botão de “chave devolvida”.
2. [EV] O sistema salva o novo status.
3. [RS] O sistema envia uma notificação ao usuário solicitante da reserva, informando sobre a alteração de status da reserva.
4. [RS] O sistema envia uma notificação ao coordenador, informando sobre a alteração de status da reserva.
5. [RS] O sistema exibe uma tela com a mensagem [MSG001].

Pós-condições:

FA1 - O novo status da reserva é salvo no banco de dados.

FA2 - O novo status da reserva é salvo no banco de dados .

Fluxo de exceção:

Não possui.

Mensagens:

MSG001 - O cartão magnético foi marcado como entregue e o coordenador será notificado!

MSG002 - O cartão magnético foi marcado como devolvido e o coordenador será notificado!

UC008 - Registrar ocorrência no laboratório

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa registrar qualquer ocorrência dentro dos laboratórios de monitoria.

Atores: Monitor.

Pré-condições:

O Autor está autenticado no sistema.

Fluxo normal :

1. [RS] O sistema exibe uma tela com campos para inserir os dados da ocorrência.
2. [EV] O autor preenche todos os campos no formulário.
3. [EV] O autor clica no botão enviar.
4. [RS] O sistema armazena a ocorrência.
5. [RS] O sistema envia uma notificação ao coordenador.
6. [RS] O sistema exibe um modal com a mensagem [MSG001]

Fluxo Alternativo:

Não possui.

Pós-condições:

FN - Um novo registro de ocorrência é salvo no banco de dados.

Fluxo de exceção:

FE1 - O ator não preencheu corretamente os campos de entrada.

1. [RS] O sistema exibe um modal com a mensagem [MSG002]

Mensagens:

MSG001 - A Ocorrência foi enviada para análise e o coordenador responsável será notificado.

MSG002 - Erro! Verifique se todos os campos estão preenchidos.

UC009 - Visualizar registros de ocorrência

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa visualizar os registros de ocorrência no laboratório.

Atores: Monitor.

Pré-condições:

O autor está autenticado no sistema.[FA1] [FA2]

Fluxo normal :

1. [RS] O sistema exibe uma tela com uma lista de registro de ocorrências.

Fluxo Alternativo:

FA1- Clique em um dos registros de ocorrência.

1. [RS] O sistema exibe uma tela com todos os detalhes da ocorrência registrada solicitada pelo autor.

FA2- O Autor não realizou registros de ocorrência.

1. [RS] O sistema exibe uma tela com a mensagem [MSG001]

Pós-condições:

Não Possui.

Fluxo de exceção:

Não Possui.

Mensagens:

MSG001 - Ainda não existem ocorrências para análise!

UC010 - Gerenciar registros de ocorrências

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o autor possa gerenciar o status dos registros de ocorrência.

Atores: Coordenador.

Pré-condições:

O autor está autenticado no aplicativo. [FA1] [FA2][FA3]

Fluxo normal

1. [RS] O sistema exibe uma tela com todos os registros de ocorrência.
2. [EV] O autor clica em um dos registros de ocorrência.
3. [RS] O sistema exibe uma tela com informações detalhadas sobre a ocorrência que foi registrada.

Fluxo Alternativo

FA1 - Alterar status da ocorrência para concluída.

1. [EV] O autor clica no campo de entrada para descrição.
2. [EV] O autor clica no botão de concluído.
3. [RS] O sistema salva o novo status.
4. [RS] O sistema envia uma notificação ao usuário que registrou a ocorrência informando sobre a alteração de status.
5. [RS] O sistema exibe uma tela com a mensagem [MSG001]

FA2 - Alterar status da ocorrência para "em análise".

1. [EV] O autor clica no campo de entrada para descrição.
2. [EV] O autor clica no botão de "em análise".
3. [RS] O sistema salva o novo status.
4. [RS] O sistema envia uma notificação ao usuário que registrou a ocorrência informando sobre a alteração de status.
5. [RS] O sistema exibe uma tela com a mensagem [MSG002]

FA3 - Alterar status da ocorrência para deletada.

1. [EV] O autor clica no campo de entrada para descrição.
2. [EV] O autor clica no botão de deletar.
3. [RS] O sistema salva o novo status.
4. [RS] O sistema envia uma notificação ao usuário que registrou a ocorrência informando sobre a alteração de status.
5. [RS] O sistema exibe uma tela com a mensagem [MSG003]

Pós-condições:

FA1 - O novo status da ocorrência é salvo no banco de dados.

FA2 - O novo status da ocorrência é salvo no banco de dados .

FA3 - O novo status da ocorrência é salvo no banco de dados .

Fluxo de exceção:

Não Possui.

Mensagens:

MSG001 - Pronto! Agora o usuário responsável será notificado.

MSG002 - A ocorrência foi colocada em análise e o usuário responsável será notificado.

MSG003 - A ocorrência foi deletada e o usuário responsável será notificado.

UC011 - Gerenciar monitores

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o autor possa gerenciar os monitores do laboratório.

Atores: Coordenador.

Pré-condições:

O autor está autenticado no aplicativo. [FA1] [FA2] [FA3] [FA4]

FA1, FA3,FA4 - Possuir monitores cadastrados.

FA4 - Possuir monitores cadastrados com horários.

Fluxo normal

1. [RS] O sistema exibe uma tela com todos os monitores.

Fluxo Alternativo:

FA1 - Remover um monitor.

1. [EV] O autor clica no botão vermelho com um X
2. [RS] O sistema exibe uma tela com a mensagem de pergunta [MSG001].
3. [EV] O autor clica no botão “sim”.
4. [RS] O sistema remove o monitor.
5. [RS] O sistema envia uma notificação ao monitor que foi removido.
6. [RS] O sistema exibe uma tela com a mensagem [MSG002].

FA2 - Importar monitores.

1. [EV] O autor clica no botão "Importar novos monitores”
2. [RS] O sistema exibe uma tela para o usuário selecionar o arquivo CSV onde estão os monitores.
3. [EV] O sistema importa os novos monitores e horários.
4. [RS] O sistema envia uma notificação aos novos monitores informando que agora eles são monitores no aplicativo.
5. [RS] O sistema exibe uma tela com a mensagem [MSG003].

FA3 - Alterar horário de monitoria do monitor para um horário disponível.

1. [EV] O autor clica no botão com ícone de relógio do monitor desejado.
2. [RS] O sistema exibe a tela de horário do monitor com dias da semana e horários.
3. [EV] O autor seleciona os horários que deseja adicionar ou remover.

4. [EV] O autor clica em salvar modificações.
5. [RS] O sistema salva o(s) novo(s) horário(s) do monitor.
6. [RS] O sistema envia uma notificação ao monitor informando sobre a alteração de horário.
7. [RS] O sistema exibe uma tela com a mensagem [MSG004].

FA4 - Alterar horário de monitoria do monitor para um horário indisponível.

1. [EV] O autor clica no botão com ícone de relógio do monitor desejado.
2. [RS] O sistema exibe a tela de horário do monitor com dias da semana e horários.
3. [EV] O autor seleciona os horários que deseja adicionar ou remover.
4. [EV] O autor clica em salvar modificações.
5. [RS] O sistema identifica que o horário não está disponível.
6. [RS] O sistema exibe um modal com a mensagem [MSG005]
7. [EV] O autor clica no botão "Sobrescrever"
8. [RS] O sistema salva os novos horários dos monitores.
9. [RS] O sistema envia uma notificação aos monitores informando sobre a alteração de horários.
10. [RS] O sistema exibe uma tela com a mensagem [MSG006]

Pós-condições:

FA1 - O monitor é removido do banco de dados.

FA2 - Os novos monitores são adicionados no banco de dados junto aos seus horários.

FA3 - O monitor tem seu horário atualizado no banco de dados .

FA4 - Os monitores têm seus horários atualizados no banco de dados .

Fluxo de exceção:

FE1- Autor não deseja remover monitor

1. [EV] O autor clica no botão vermelho com um X

2. [RS] O sistema exibe uma tela com a mensagem de pergunta [MSG001]
3. [EV] O autor clica no botão “não”.
4. Retorna ao passo 1 do FN

FE2- Autor não deseja sobrescrever horários de monitores

1. [EV] O autor clica no botão com ícone de relógio do monitor desejado.
2. [RS] O sistema exibe a tela de horário do monitor com dias da semana e horários.
3. [EV] O autor seleciona os horários que deseja adicionar ou remover.
4. [EV] O autor clica em salvar modificações.
5. [RS] O sistema identifica que o horário não está disponível.
6. [RS] O sistema exibe um modal com a mensagem [MSG005]
7. [EV] O autor clica no botão "Cancelar"
8. Retorna para o passo 02 do FE2.

Mensagens:

MSG001 - Tem certeza de que deseja remover o monitor *NOME DO MONITOR* ?

MSG002 - Pronto! Agora o usuário não faz mais parte da monitoria.

MSG003 - A importação ocorreu com sucesso..

MSG004 - A alteração ocorreu com sucesso e o monitor será notificado.

MSG005 - Já existem monitores nos seguintes horários *NOME E HORARIOS* deseja manter? Os monitores desses horários devem ser realocados para outros horários.

MSG006 - A alteração ocorreu com sucesso e os monitores foram notificados.

MSG007 - Pronto! Agora o usuário não faz mais parte da monitoria.

UC012 - Visualizar regulamentos

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa visualizar os regulamentos dos laboratórios

Atores: Coordenador, Monitor e usuário.

Pré-condições:

O Autor está autenticado no sistema.

Fluxo normal

1. [RS] O sistema exibe uma tela informativa de regulamentos com botões que levam ao arquivo de regulamentos.
2. [EV] O autor clica no botão para baixar o pdf.

Fluxo Alternativo:

Não possui.

Pós-condições:

Fluxo de exceção:

Não possui.

Mensagens:

Não possui.

UC013 - Visualizar contatos

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa visualizar os contatos de coordenadores dos laboratórios.

Atores: Coordenador, Monitor e usuário.

Pré-condições:

O Autor está autenticado no sistema.

Fluxo normal

1. [RS] O sistema exibe uma tela informativa com os contatos do coordenador dos laboratórios.

Fluxo Alternativo:

Não possui.

Pós-condições:

Não possui

Fluxo de exceção:

Não possui.

Mensagens:

Não possui.

UC014 - Visualizar informações de infraestrutura dos laboratórios

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa visualizar informações de infraestrutura dos laboratórios

Atores: Coordenador, Monitor e usuário.

Pré-condições:

O Autor está autenticado no sistema.

Fluxo normal

1. [RS] O sistema exibe uma tela informativa sobre a infraestrutura dos laboratórios.

Fluxo Alternativo:

Não possui.

Pós-condições:

Não possui.

Fluxo de exceção:

Não possui.

Mensagens:

Não possui.

UC015 - Usuário se tornar monitor

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa visualizar as informações e preencher formulário para se tornar monitor.

Atores: Coordenador, Monitor e usuário.

Pré-condições:

O Autor está autenticado no sistema.

Fluxo normal

1. [RS] O sistema exibe uma tela informativa sobre o processo de monitoria

Fluxo Alternativo:

FA1 - Usuário quer ser tornar monitor

1. [EV] O Autor confirma que leu os termos.
2. [EV] O autor clica no botão de registrar-se
3. [RS] O sistema exibe o formulário para preenchimento de informações.

Pós-condições:

Não possui.

Fluxo de exceção:

Não possui.

Mensagens:

Não possui.

UC016 - Visualizar notificações recebidas

Descrição: Esse caso de uso descreve como deve ser o fluxo para que o usuário possa visualizar as notificações recebidas no aplicativo.

Atores: Coordenador, Monitor e usuário.

Pré-condições:

O Autor está autenticado no sistema.

Fluxo normal

1. [RS] O sistema exibe uma tela com a lista de todas as notificações recebidas.

Fluxo Alternativo:

Não possui.

Pós-condições:

Não possui.

Fluxo de exceção:

Não possui.

Mensagens:

Não possui.

APÊNDICE B – Descrição das tabelas do modelo conceitual do banco de dados

Tabela de coordinator.

Atributo	Tipo	Descrição
id	SERIAL4	Chave primária da tabela coordinator.
user_id	INT(4)	Chave estrangeira referente a tabela user.
emergency_email	TEXT	Email para emergências
is_current	BOOL	Controle para saber se o coordenador está ativo.
create_at	TIMESTAMP	Data e hora de criação do coordenador
update_at	TIMESTAMP	Data e hora de atualização do coordenador

Fonte : O autor (2022)

Tabela de day_week.

Atributo	Tipo	Descrição
Id	SERIAL(4)	Chave primária da tabela.
day_week	TEXT	Dia da semana

Fonte : O autor (2022)

Tabela 5 - Tabela de hour

Atributo	Tipo	Descrição
Id	SERIAL(4)	Chave primária da tabela
hour	TEXT	Hora do dia

Fonte : O autor (2022)

Tabela de monitor.

Atributo	Tipo	Descrição
id	SERIAL(4)	Chave primária da tabela
user_id	INT(4)	Chave estrangeira referente a tabela user.
profile_photo	TEXT	Foto de perfil
delete_at	TIMESTAMP	Data e hora que foi excluído
create_at	TIMESTAMP	Data e hora que foi criado
update_at	TIMESTAMP	Data e hora que foi atualizado

Fonte : O autor (2022).

Tabela de notification.

Atributo	Tipo	Descrição
id	SERIAL(4)	Chave primária da tabela
title	TEXT	Título da notificação
description	TEXT	Descrição da notificação
date_request	TIMESTAMP	Data e hora da solicitação de envio
date_send	TIMESTAMP	Data e hora do envio
delete_at	TIMESTAMP	Data e hora que foi deletado
create_at	TIMESTAMP	Data e hora que foi criado
update_at	TIMESTAMP	Data e hora que foi atualizado

Fonte : O autor (2022)

Tabela de reservation.

Atributo	Tipo	Descrição
id	SERIAL(4)	Chave primária da tabela
username	TEXT	Nome do usuário
email	TEXT	Email do usuário
cpf	TEXT	Cpf do usuário
telefone	TEXT	Telefone do usuário
function_at_ufes	TEXT	Função que exerce na UFES
reservation_date	TIMESTAMP	Data da reserva
start time	TEXT	Hora de início da reserva
end_time	TEXT	Hora de fim da reserva
consent_form	BOOL	Termo de consentimento
is_delivered	BOOL	Identificador de que foi entregue a chave
is_returned	BOOL	Identificado de que foi devolvida a chave
delete_at	TIMESTAMP	Data e hora que foi deletado
create_at	TIMESTAMP	Data e hora que foi criado
update_at	TIMESTAMP	Data e hora que foi atualizado

Fonte : O autor (2022)

Tabela de suport.

Atributo	Tipo	Descrição
id	SERIAL(4)	Chave primária da tabela
user_id	INT(4)	Chave estrangeira referente a tabela user.
function_at_ufes	TEXT	Função na ufes

course_department	TEXT	Departamento do curso
request_type	TEXT	Tipo de solicitação
room	TEXT	Laboratório
equipment_identification	TEXT	Identificação do equipamento
description	TEXT	Descrição do suporte
description_resolution	TEXT	Descrição da resolução
status	TEXT	Andamento do suporte
date_occurrence	TIMESTAMP	Data e hora da solicitação do suporte
resolved_in	TIMESTAMP	Data e hora que foi resolvido
delete_at	TIMESTAMP	Data e hora que foi deletado
create_at	TIMESTAMP	Data e hora que foi criado
update_at	TIMESTAMP	Data e hora que foi atualizado

Fonte : O autor (2022)

Tabela de time.

Atributo	Tipo	Descrição
Id	SERIAL(4)	Chave primária da tabela
monitor_id	INT(4)	Chave estrangeira referente a tabela monitor.
day_week_id	INT(4)	Chave estrangeira referente a tabela day_week.
hour_id	INT(4)	Chave estrangeira referente a tabela hour.
create_at	TIMESTAMP	Data e hora que foi criado
update_at	TIMESTAMP	Data e hora que foi atualizado

Fonte : O autor (2022)

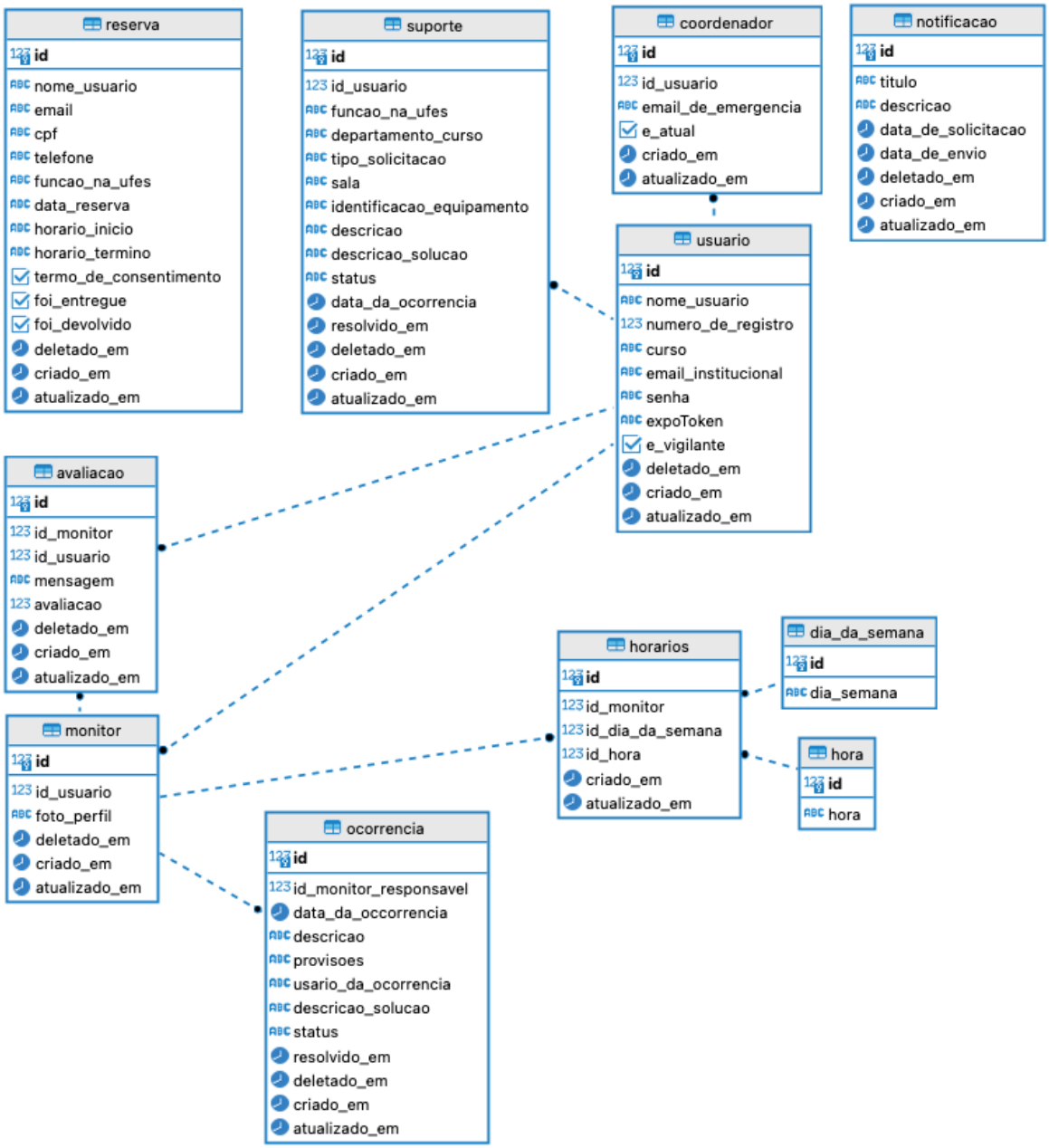
Tabela de user.

Atributo	Tipo	Descrição
id	SERIAL(4)	Chave primária da tabela
username	TEXT	Nome do usuário
registration_number	INT(8)	Número de matrícula do usuário
course	TEXT	Curso do usuário
institutional_email	TEXT	E-Mail institucional do usuário
password	TEXT	Senha do usuário
expoToken	TEXT	Token do aparelho do usuário
is_watchman	BOOL	Identificador se o usuário é vigilante
delete_at	TIMESTAMP	Data e hora que foi deletado
create_at	TIMESTAMP	Data e hora que foi criado
update_at	TIMESTAMP	Data e hora que foi atualizado

Fonte : O autor (2022)

APÊNDICE C – Modelo físico de banco de dados em português

A figura abaixo ilustra o modelo físico de banco de dados citado no capítulo 4.6.2 com a mesma modelagem porém usando o português para definir as tabelas e colunas.



APÊNDICE D - Tarefas elaboradas para o desenvolvimento do sistema.

Backlog	Descrição	Módulo	Ambiente
Estrutura inicial do app	Criar o app utilizando o react native	Configuração	Front
Estrutura inicial do backend	Criar o backend utilizando o node.js	Configuração	Back
Integrar os serviços do firebase no app	Integrar os serviços do firebase no app para consumo de storages e controle de variáveis remotas	Configuração	Front
Adicionar push notification no app	Integrar os serviços do expo push notification para registrar e salvar o id dos devices dos usuários	Configuração	Back
Gerar aplicativo na google play	Criar uma hospedagem do aplicativo no google play store	Configuração	Front
Subir o backend em um serviço de hospedagem	Adicionar o backend a um serviço de hospedagem em nuvem para desenvolvimento	Configuração	Back
Criar banco de dados de produção e desenvolvimento	Criar o banco de dados que será usado em produção e em desenvolvimento	Configuração	Back
Criar primeiros protótipos do app para avaliação	Criar os protótipos do app para serem enviados ao orientador para aprovação	Configuração	Front
Criar tela de login	Desenvolver a tela de login do app para que usuário possa se autenticar e usar os serviços	Usuário	Front
Criar api's de login	Desenvolver o login no servidor para que usuário possa se autenticar e usar os serviços	Usuário	Back
Criar conexão com LDPA da Ufes	Desenvolver o login com o LDAP da ufes no servidor para que usuário possa se autenticar e usar os serviços	Usuário	Back
Criar tela inicial do app	Desenvolver a tela inicial do app com os serviços essenciais que deverão ser prestados aos discentes	App	Front
Criar menu lateral do app	Desenvolver o menu lateral onde vão estar localizada os demais menus do app	App	Front
Criar api's para menu de suporte	Desenvolver no servidor as api's referente a suporte do app, tais como (consultar e registrar suportes)	Suporte	Back
Criar menu de suporte no app	Desenvolver menu aonde qualquer usuário logado no app vai ter acesso aos serviços de suporte	Suporte	Front
Criar api's de coordenador	Desenvolver no servidor as api's referente o coordenador do app, tais como (consultar monitores, horários,	Coordenador	Back

	registrar monitores, deletar etc..)		
Criar menus de coordenador no app	Desenvolver menu onde o coordenador vigente vai ter acesso aos serviços para coordenador	Coordenador	Front
Criar api's de monitor	Desenvolver no servidor as api's referente a monitores do app, tais como (consultar horários, visualizar e registrar ocorrências)	Monitor	Back
Criar menu de monitores no app	Desenvolver menu para monitores onde será possível consumir os serviços para monitores	Monitor	Front
Criar api's de reserva de sala	Desenvolver no servidor as api's referente a reserva de sala do app, tais como (registrar e consultar)	Reserva	Back
Criar menu para reserva de sala no app	Desenvolver menu para qualquer usuário onde o mesmo possa ter acesso aos serviços de reserva de sala	Reserva	Front

APÊNDICE E - Perguntas para teste de usabilidade.

E.1 Formulário de teste de usabilidade 01 para autores C1

Formulário de teste de usabilidade 01 para autores C1

Qual a dificuldade de realizar o login ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores, erros ou melhorias) sobre o Login :

Sua resposta

Qual a dificuldade de visualizar as ocorrências registradas ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre a vizualização de ocorrencias

Sua resposta

Qual a dificuldade de realizar o login ? *

- ◯ 1 - Muito Difícil
- ◯ 2- Difícil
- ◯ 3 - Normal
- ◯ 4- Fácil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores, erros ou melhorias) sobre o Login :

Sua resposta

Qual a dificuldade de visualizar as ocorrências registradas ? *

- ◯ 1 - Muito Difícil
- ◯ 2- Difícil
- ◯ 3 - Normal
- ◯ 4- Fácil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre a vizualização de ocorrencias

Sua resposta

Qual a dificuldade de alterar o status de uma ocorrência ? *

- ◯ 1 - Muito Difícil
- ◯ 2- Difícil
- ◯ 3 - Normal
- ◯ 4- Fácil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre os detalhes da ocorrencia :

Sua resposta

Qual a dificuldade de visualizar os suportes solicitados ? *

- ◯ 1 - Muito Dificil
- ◯ 2- Dificil
- ◯ 3 - Normal
- ◯ 4- Facil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre a solicitação de suporte :

Sua resposta

Qual a dificuldade de visualizar detalhes da solicitação de suporte ? *

- 1 - Muito Dificil
- 2- Dificil
- 3 - Normal
- 4- Facil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre os detalhes de solicitação de suporte :

Sua resposta

Qual a dificuldade de encontra informações sobre os monitores ? *

- 1 - Muito Dificil
- 2- Dificil
- 3 - Normal
- 4- Facil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre dificuldade de encontra informações dos monitores.

Sua resposta

Qual a dificuldade de importa os monitores para o app ? *

- 1 - Muito Dificil
- 2- Dificil
- 3 - Normal
- 4- Facil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre importa monitores para o app.

Sua resposta

Qual a dificuldade de remover os monitores do app ? *

- 1 - Muito Dificil
- 2- Dificil
- 3 - Normal
- 4- Facil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre remover monitores do app.

Sua resposta

Qual a dificuldade de atualizar os horario de monitores ? *

- 1 - Muito Dificil
- 2- Dificil
- 3 - Normal
- 4- Facil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre atualizar horarios de monitores.

Sua resposta

Qual a dificuldade de enviar notificações ? *

- 1 - Muito Dificil
- 2- Dificil
- 3 - Normal
- 4- Facil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre enviar notificações.

Sua resposta

Qual a dificuldade de visualizar as solicitações de reserva de sala ? *

- ◯ 1 - Muito Dificil
- ◯ 2- Dificil
- ◯ 3 - Normal
- ◯ 4- Facil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre a solicitação de reserva de sala.

Sua resposta

Qual a dificuldade de visualizar os detalhes e atualizar os status das solicitações de reserva de sala ? *

- ◯ 1 - Muito Dificil
- ◯ 2- Dificil
- ◯ 3 - Normal
- ◯ 4- Facil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre a dificuldade de atualizar o status da solicitação de reserva de sala.

Sua resposta

Sobre o aplicativo no geral. De 0 a 5 qual suas probabilidade de usar o aplicativo ? *

- 1 - Não utlizaria
- 2- Pouco provavel
- 3 - Não sei
- 4- Provavel
- 5 - Muito Provavel

Observações sobre a probabilidade de uso do aplicativo. *

- 1 - Não utlizaria
- 2- Pouco provavel
- 3 - Não sei
- 4- Provavel
- 5 - Muito Provavel

Observações gerais sobre esse teste realizado :

Sua resposta

Fonte : Do autor

E.2 Formulário de teste de usabilidade 02 para autores A2

Formulário de teste de usabilidade 02 para autores A2

Qual a dificuldade de realizar o login ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre o Login :

Sua resposta

Qual a dificuldade de solicitar um suporte ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre a solicitação de suporte

Sua resposta

Qual a dificuldade de verificar os detalhes do suporte solicitado ? *

- ◯ 1 - Muito Difícil
- ◯ 2- Difícil
- ◯ 3 - Normal
- ◯ 4- Fácil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores, erros ou melhorias) sobre os detalhes do suporte solicitado :

Sua resposta

Qual a dificuldade de solicitar o cartão magnético ? *

- ◯ 1 - Muito Dificil
- ◯ 2- Difícil
- ◯ 3 - Normal
- ◯ 4- Fácil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre a solicitação de cartão magnético :

Sua resposta

Qual a dificuldade de visualizar detalhes da solicitação de cartão magnético ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre os detalhes de solicitação de cartão magnético :

Sua resposta

Qual a dificuldade de encontra informações referente aos laboratórios e monitoria ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre dificuldade de encontra informações referente aos laboratórios e monitoria.

Sua resposta

Observações gerais sobre esse teste realizado :

Sua resposta

Fonte: Do autor.

E.3 Formulário de teste de usabilidade 03 para autores A3

Formulário de teste de usabilidade 03 para autores A3

Qual a dificuldade de registrar uma ocorrência ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre o registro de ocorrência .

Sua resposta

Qual a dificuldade de verificar os detalhes da ocorrência registrada ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observações gerais sobre esse teste realizado :

Sua resposta

Fonte : Do autor.

E.4 Formulário de teste de usabilidade 04 para autores P1

Formulário de teste de usabilidade 04 para autores P1

Qual a dificuldade de realizar o login ? *

- ◯ 1 - Muito Difícil
- ◯ 2- Difícil
- ◯ 3 - Normal
- ◯ 4- Fácil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre o Login :

Sua resposta

Qual a dificuldade de visualizar as solicitações de reserva de sala ? *

- ◯ 1 - Muito Dificil
- ◯ 2- Difícil
- ◯ 3 - Normal
- ◯ 4- Fácil
- ◯ 5 - Muito Fácil

Qual a dificuldade de alterar o status de uma solicitação de reservar de sala ? *

- ◯ 1 - Muito Dificil
- ◯ 2- Dificil
- ◯ 3 - Normal
- ◯ 4- Facil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre o status de uma solicitação de reservar de sala

Sua resposta

Observações gerais sobre esse teste realizado :

Sua resposta

Fonte : Do autor.

E.4 Formulário de teste de usabilidade 05 para autores A1

Formulário de teste de usabilidade 05 para autores A1

Qual a dificuldade de realizar o login ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre o Login :

Sua resposta

Qual a dificuldade de solicitar um suporte ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre a solicitação de suporte

Sua resposta

Qual a dificuldade de verificar os detalhes do suporte solicitado ? *

- ◯ 1 - Muito Difícil
- ◯ 2- Difícil
- ◯ 3 - Normal
- ◯ 4- Fácil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores, erros ou melhorias) sobre os detalhes do suporte solicitado :

Sua resposta

Qual a dificuldade de solicitar o cartão magnético ? *

- ◯ 1 - Muito Dificil
- ◯ 2- Difícil
- ◯ 3 - Normal
- ◯ 4- Fácil
- ◯ 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre a solicitação de cartão magnético :

Sua resposta

Qual a dificuldade de visualizar detalhes da solicitação de cartão magnético ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre os detalhes de solicitação de cartão magnético :

Sua resposta

Qual a dificuldade de encontra informações referente aos laboratórios e monitoria ? *

- 1 - Muito Difícil
- 2- Difícil
- 3 - Normal
- 4- Fácil
- 5 - Muito Fácil

Observação (texto, escritas, tamanhos, cores,erros ou melhorias) sobre dificuldade de encontra informações referente aos laboratórios e monitoria.

Sua resposta

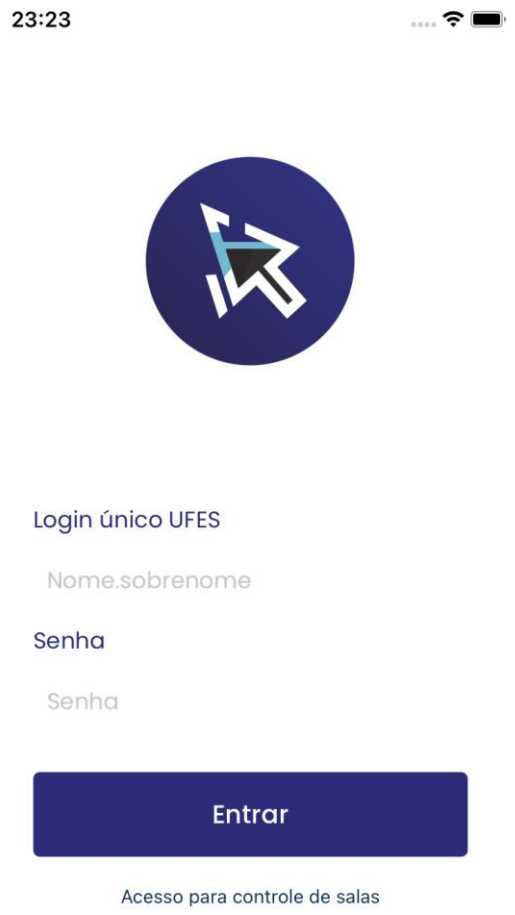
Observações gerais sobre esse teste realizado :

Sua resposta

Fonte : Do autor.

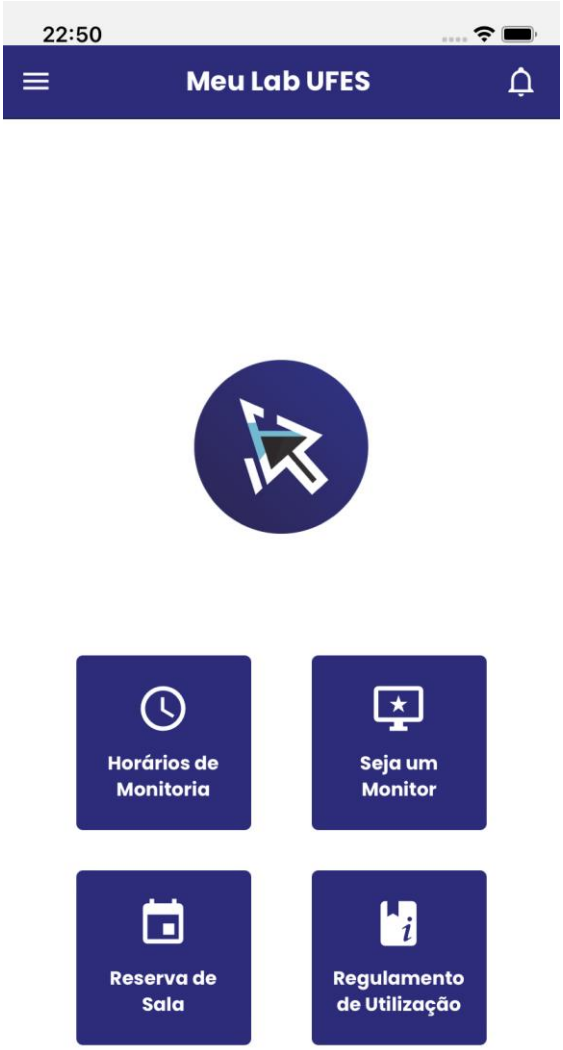
APÊNDICE F - Interfaces do sistema

Tela para usuário realizar login no sistema



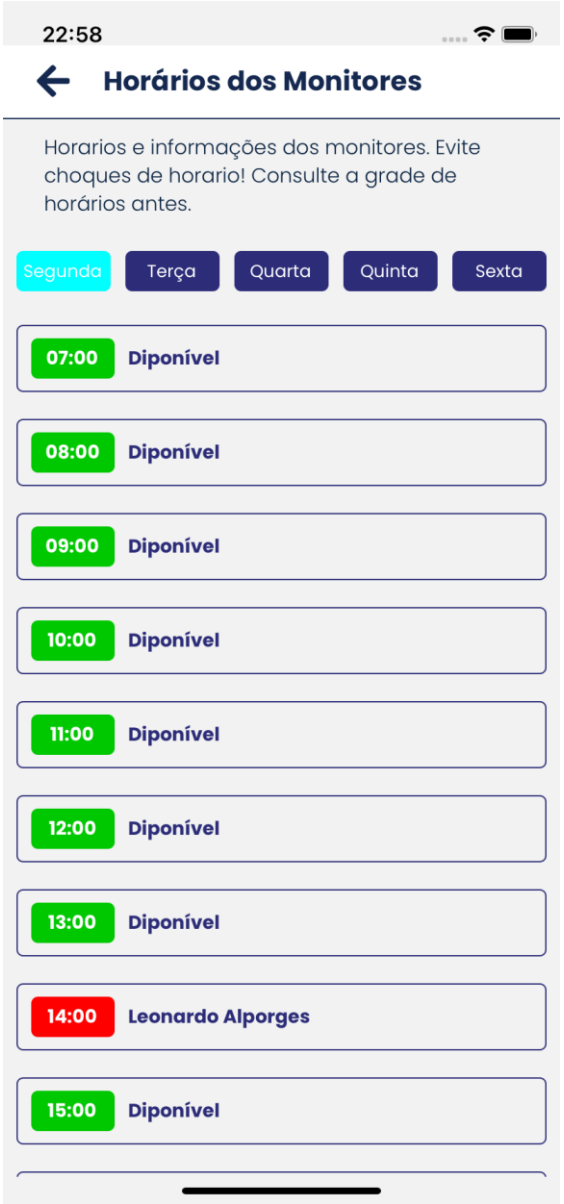
Fonte : Do autor.

Tela de início do sistema após logado



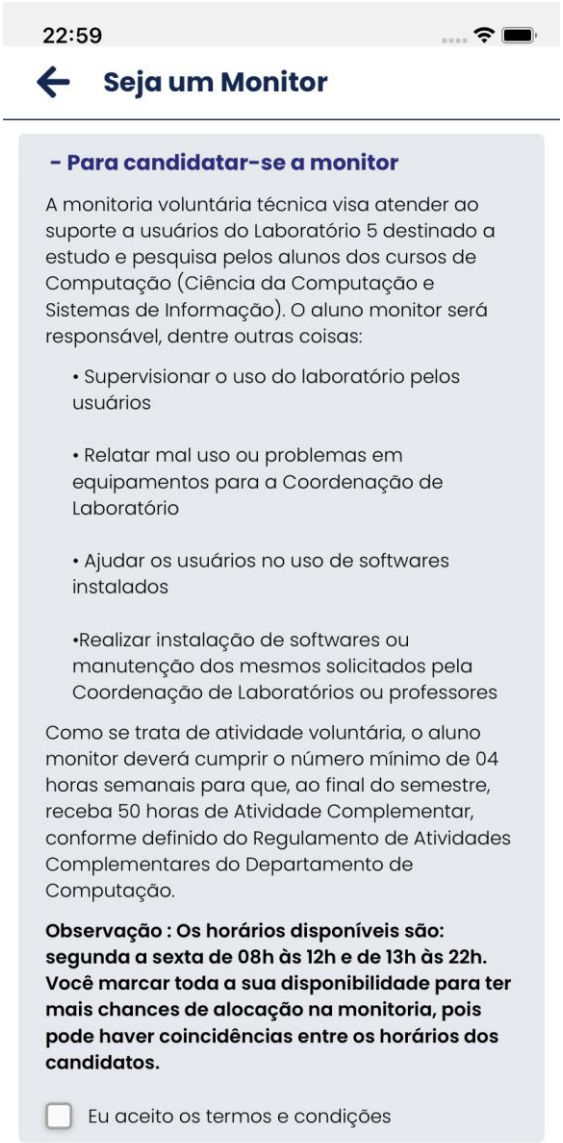
Fonte : Do autor.

Tela de horário disponível dos monitores



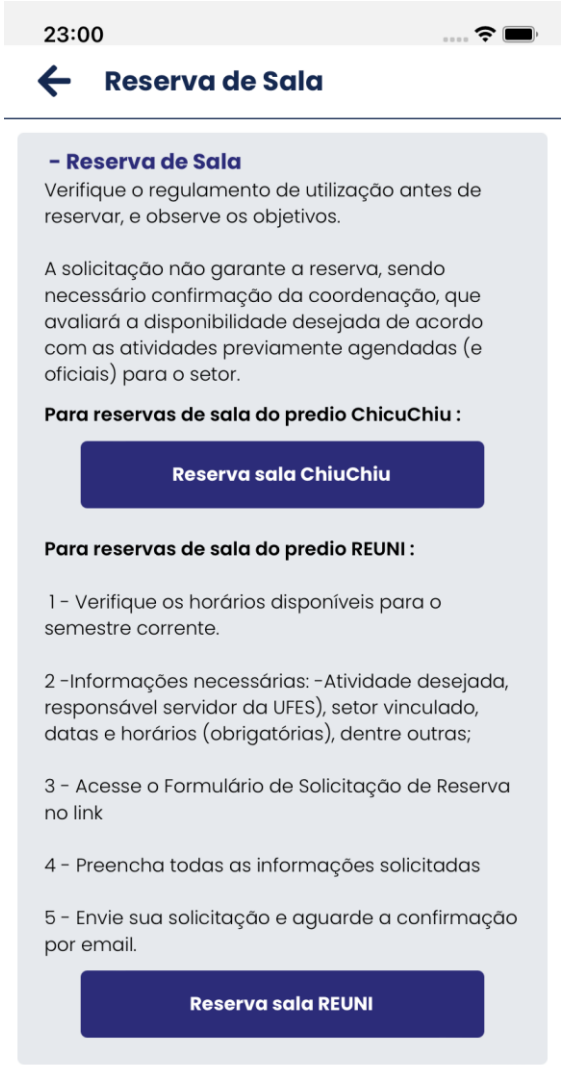
Fonte : Do autor.

Tela com informativos para usuário se tornar monitor



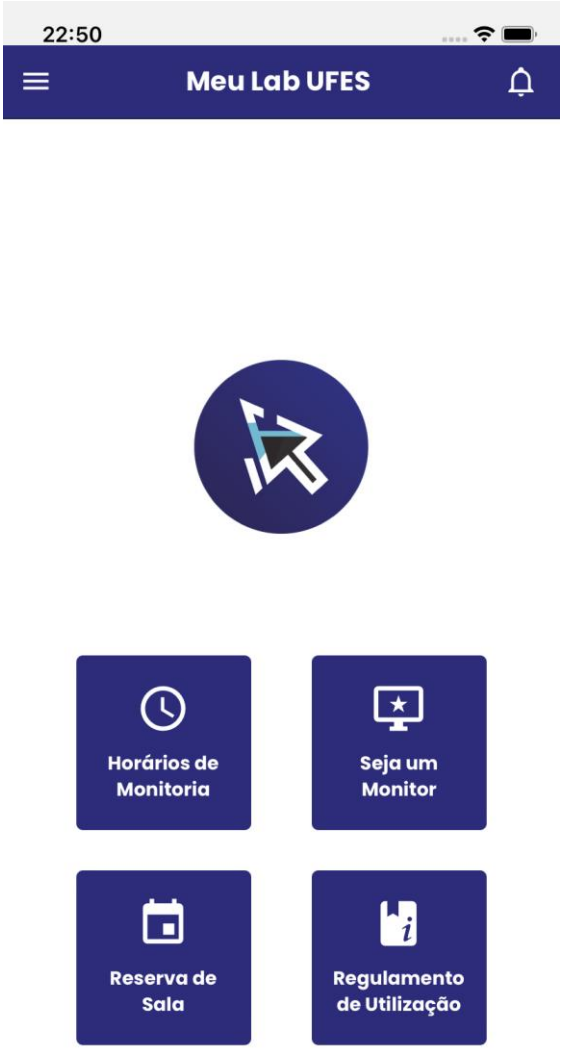
Fonte : Do autor.

Tela com informativos e formulários para reserva de salas



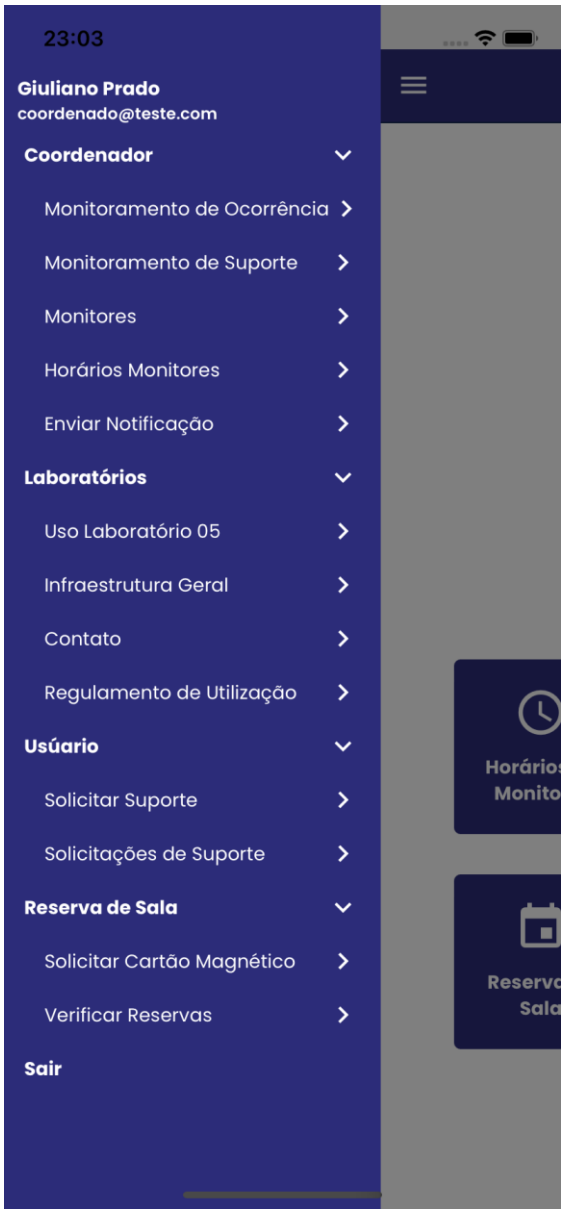
Fonte : Do autor.

Tela com os regulamentos de utilização dos laboratórios



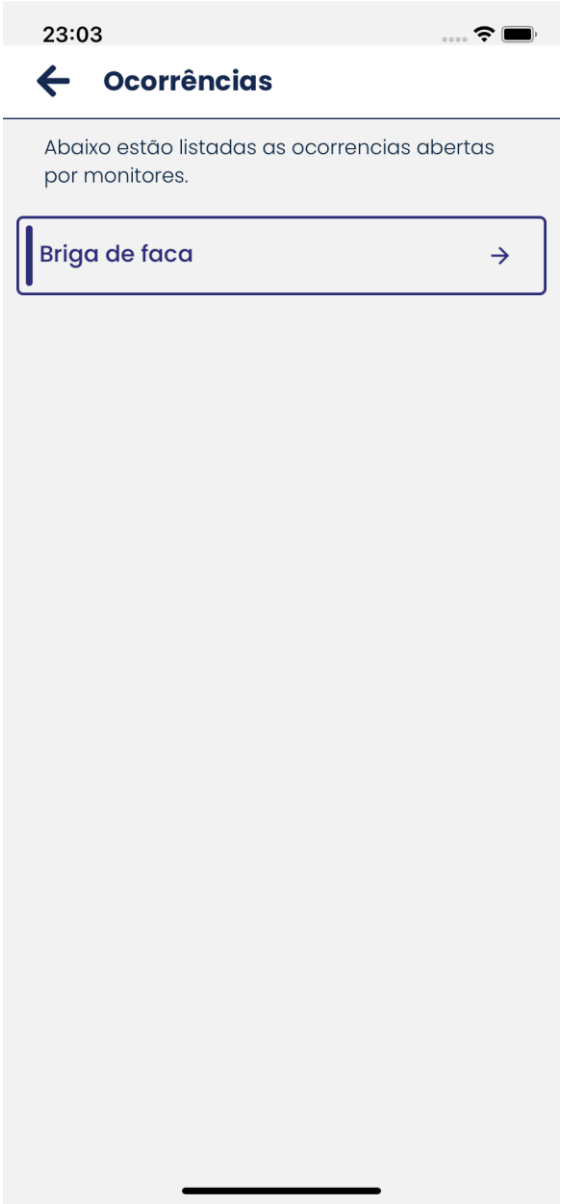
Fonte : Do autor.

Menu lateral que contém os menus para cada tipo de usuário.



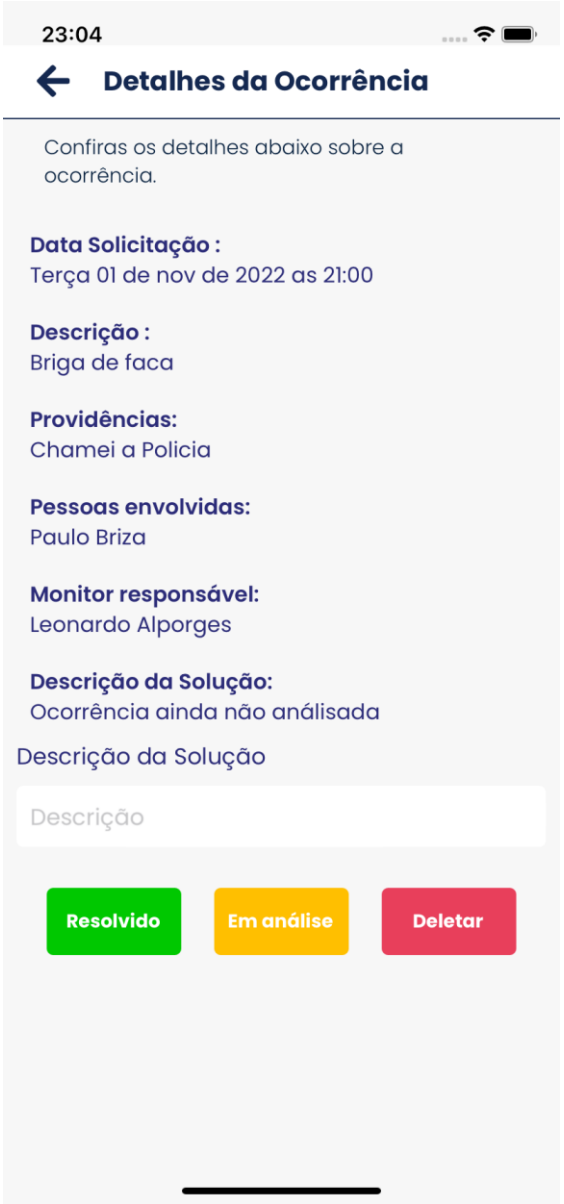
Fonte : Do autor.

Listagem de monitoramento de ocorrências registradas.



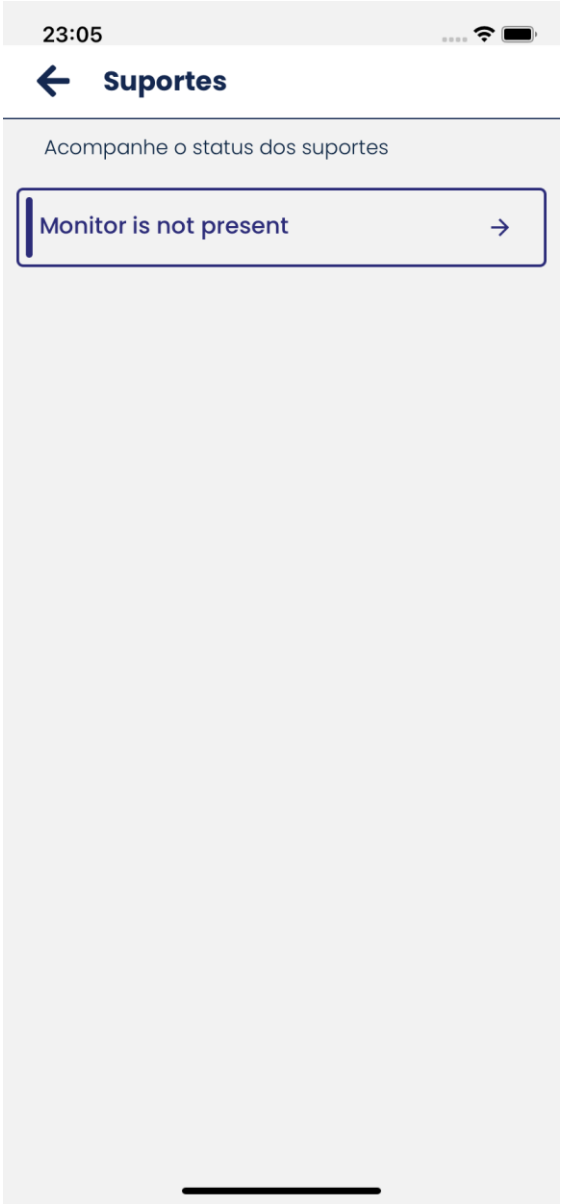
Fonte : Do autor.

Tela de detalhes do monitoramento de ocorrência



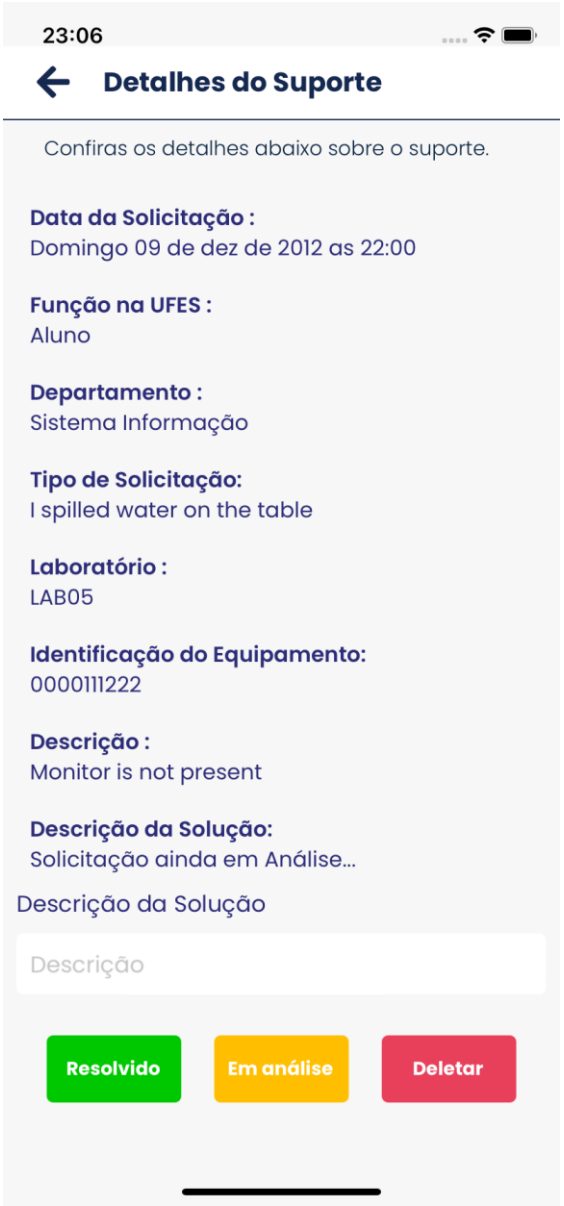
Fonte : Do autor.

Listagem de monitoramento de suportes solicitados.



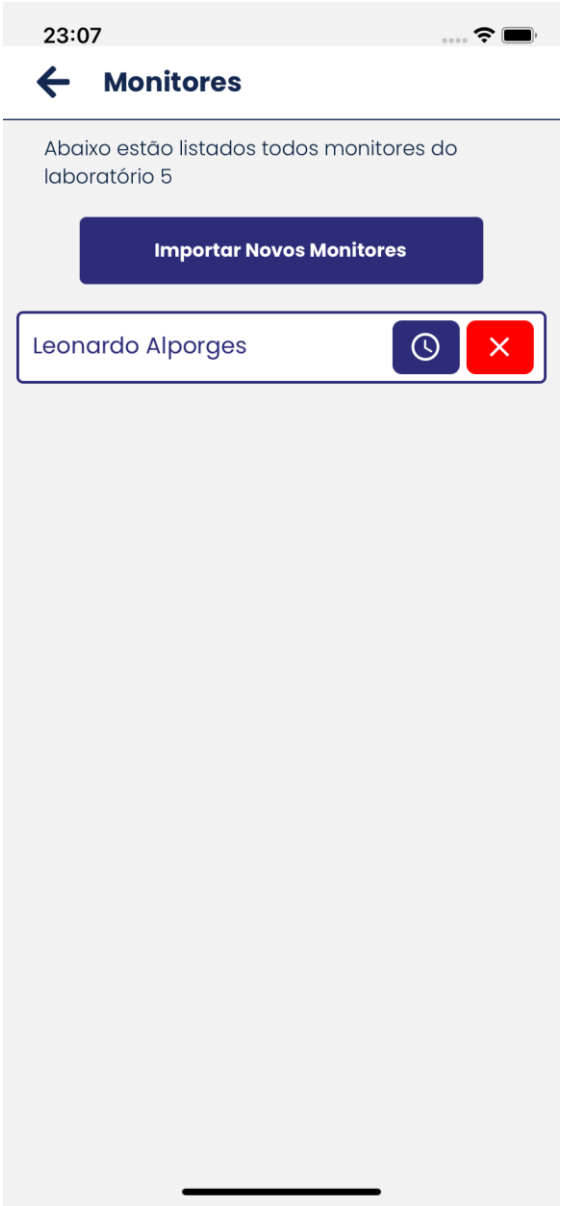
Fonte : Do autor.

Tela de detalhes do monitoramento de suporte



Fonte : Do autor.

Tela de listagem de monitores vigentes

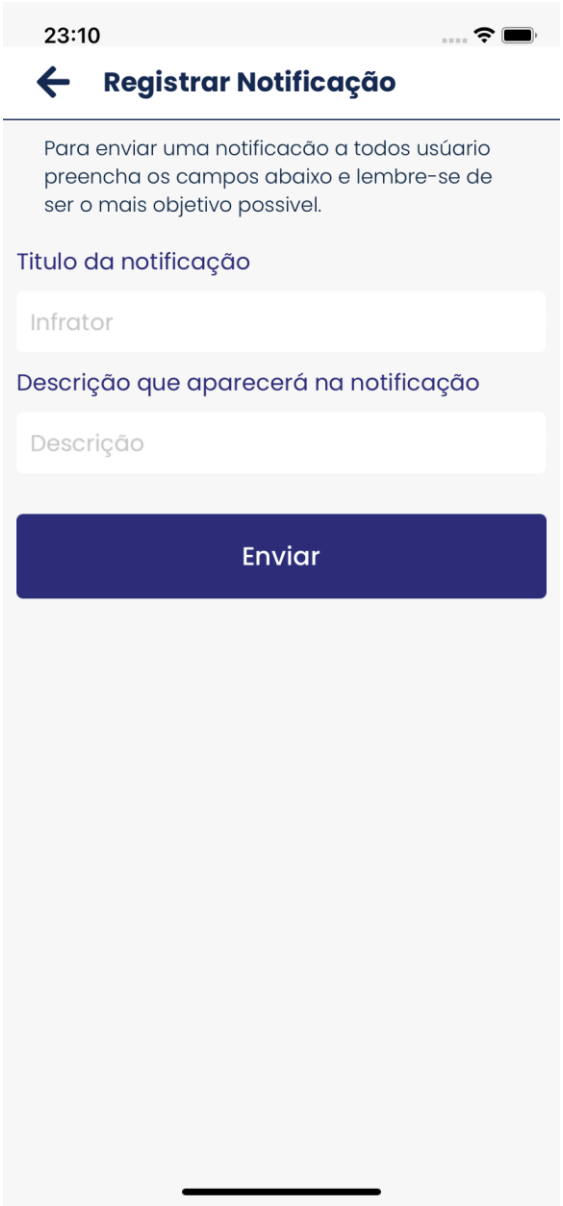


Fonte : Do autor.

Tela de alteração dos horários dos monitores

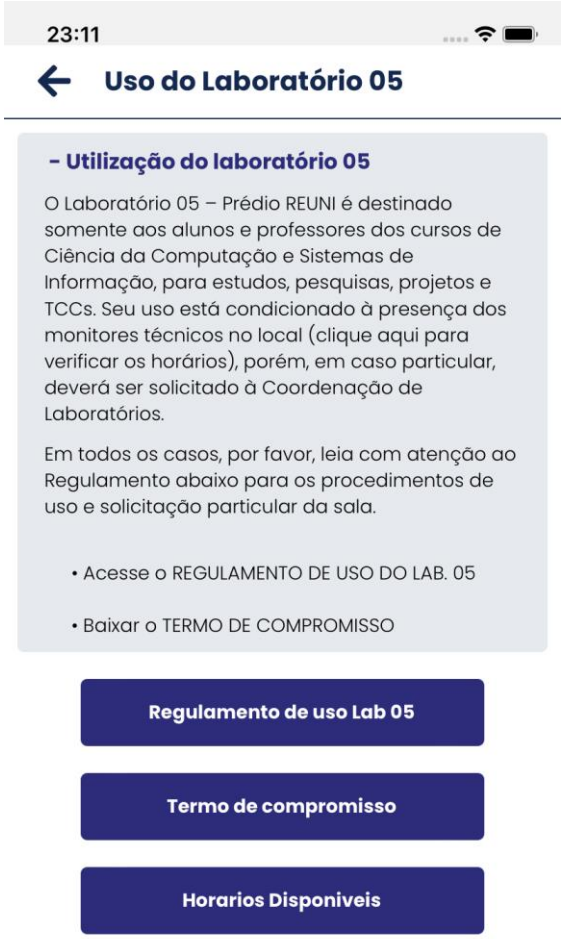
Fonte : Do autor.

Tela de registro e envio de notificação aos usuários



Fonte : Do autor.

Tela informativa e sobre a utilização do laboratório 05



Fonte : Do autor.

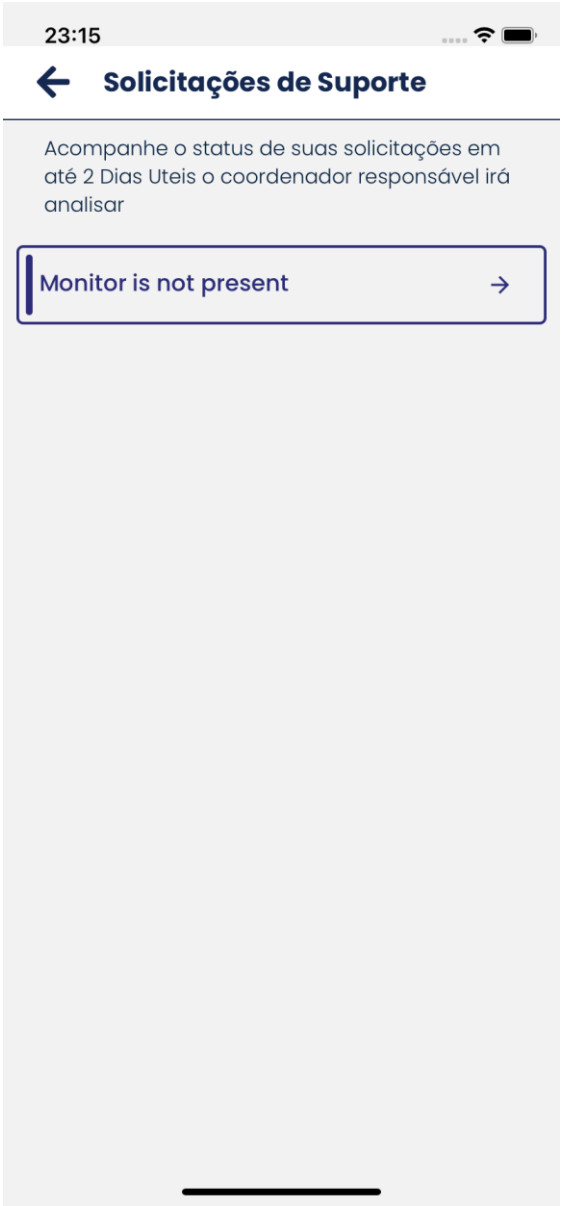
Tela com contato do coordenador vigente dos laboratórios

Fonte : Do autor.

Tela de solicitação de suporte para usuário

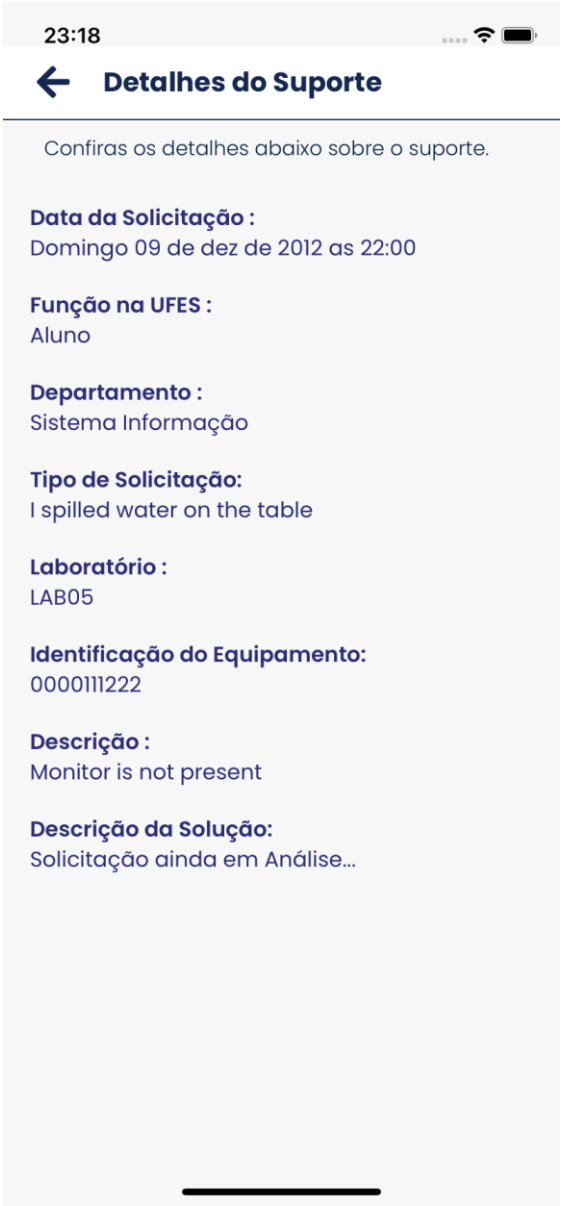
Fonte : Do autor.

Tela de listagem de acompanhamento de suporte solicitado



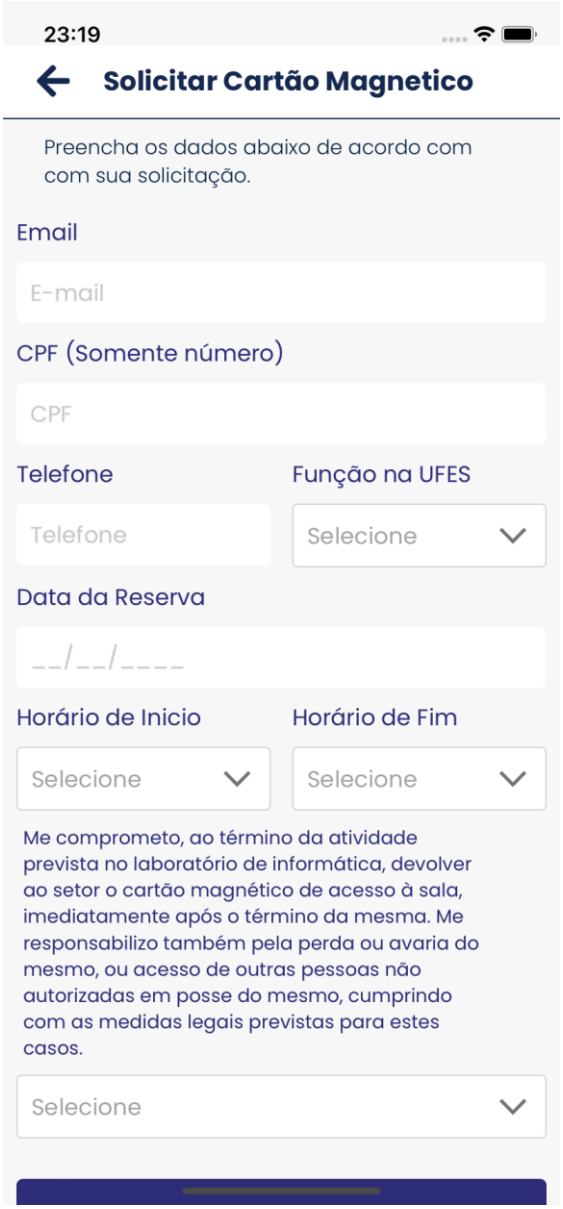
Fonte : Do autor.

Tela com detalhes sobre suporte solicitado



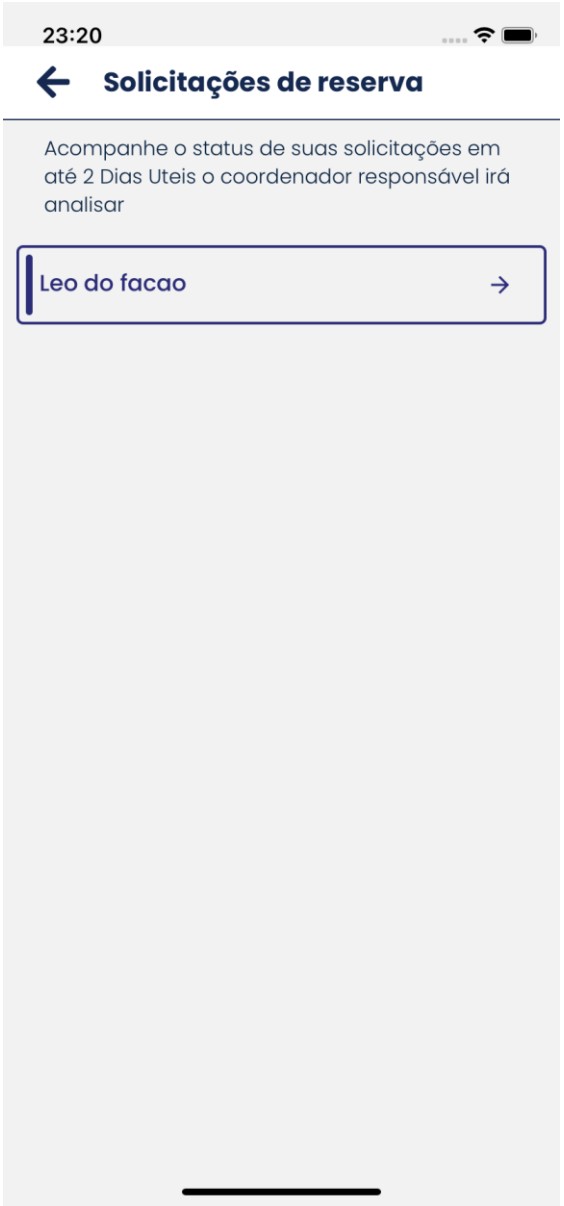
Fonte : Do autor.

Tela para solicitar reservar sala e solicitar o cartão magnético



Fonte : Do autor.

Tela de listagem de solicitação de cartão magnético

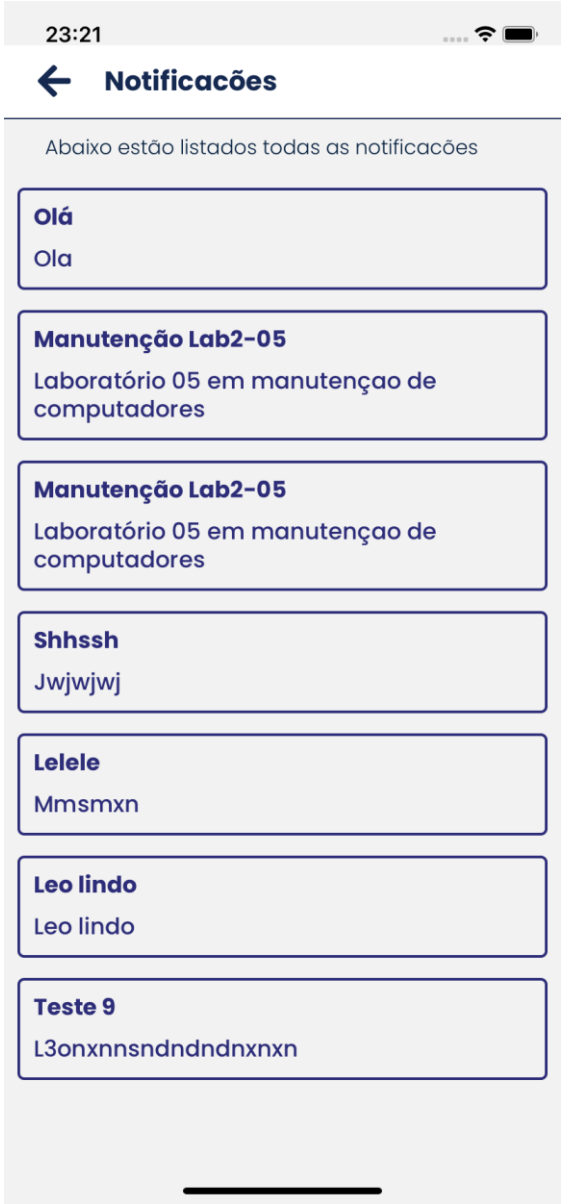


Fonte : Do autor.

Tela com detalhes sobre a solicitação do cartão magnético

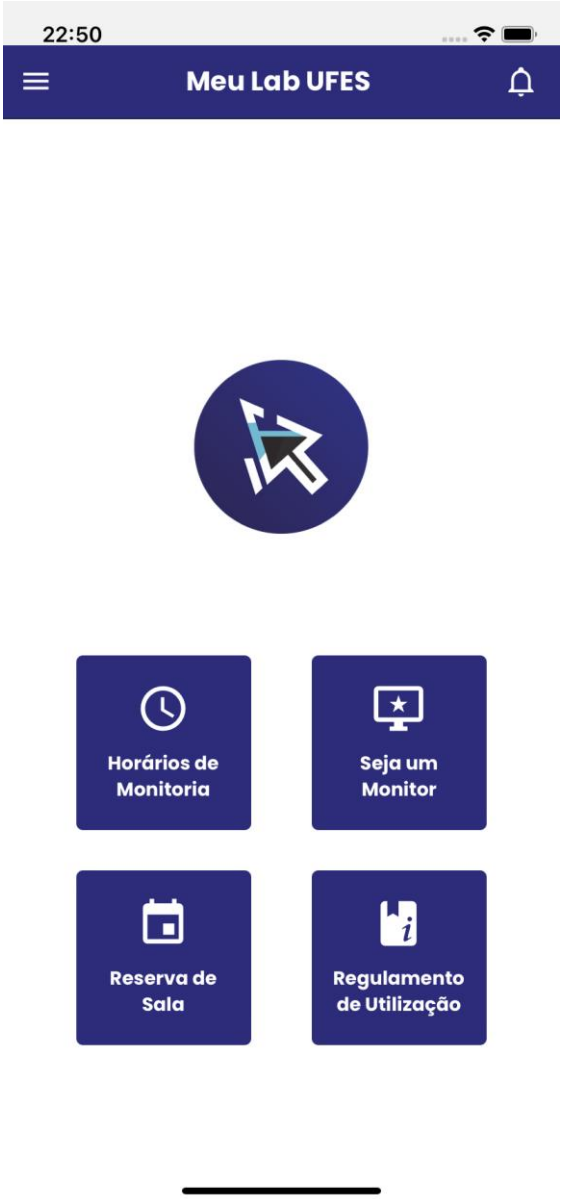
Fonte : Do autor.

Tela de listagem de notificações enviadas aos usuários



Fonte : Do autor.

Tela de início do sistema após logado



Fonte : Do autor.