**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**
**DEPARTAMENTO ACADÊMICO DE INFORMÁTICA**
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**CAIO VINICIUS SERPA**
**RAFAEL GUIMARÃES WELKE**

# APLICATIVO PARA ORGANIZAÇÃO DO LABORATÓRIO DE FERMENTAÇÃO DA UTFPR-PG

**TRABALHO DE CONCLUSÃO DE CURSO**

**PONTA GROSSA**
**2019**

**CAIO VINICIUS SERPA**

**RAFAEL GUIMARÃES WELKE**

# APLICATIVO PARA ORGANIZAÇÃO DO LABORATÓRIO DE FERMENTAÇÃO DA UTFPR-PG

Trabalho de Conclusão de Curso apresentada como requisito parcial à obtenção do título de Bacharel em Ciência da Computação, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Msc. Rogério Ranthum

**PONTA GROSSA**

**2019**

Ministério da Educação
**Universidade Tecnológica Federal do Paraná**
Câmpus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

# TERMO DE APROVAÇÃO

APLICATIVO PARA ORGANIZAÇÃO DO LABORATÓRIO DE FERMENTAÇÃO DA UTFPR-PG.

por

CAIO VINICIUS SERPA
RAFAEL GUIMARÃES WELKE

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 25 de novembro de 2019 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. MSc. Rogério Ranthum
Orientador

_____
Prof. MSc. Geraldo Ranthum
Membro titular

_____
Prof. Dr. Augusto Foronda
Membro titular

_____
Prof. MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Profª. Dra Mauren Sguario
Coordenador do curso

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

## AGRADECIMENTOS

Primeiramente gostaríamos de agradecer o nosso orientador, Rogério Ranthum, por abraçar o projeto e nos ajudar com sua experiência, ao professor Eduardo do laboratório e sua aluna Stefhani pelo convite. Agradecemos também a professora Simone Nasser, Simone Almeida e a equipe de desenvolvimento da COTED que mostraram total disposição nos momentos de dificuldades do nosso projeto.

# RESUMO

SERPA, Caio; GUIMARÃES, Rafael. **Aplicativo para organização do laboratório de fermentação da UTFPR-PG:** 2019. 59 páginas. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2019.

No ambiente da Universidade Tecnológica Federal do Paraná existe um laboratório de fermentação que é utilizado para realização de aulas práticas e outros experimentos manejados por alunos que estão sujeitos a reservas de equipamentos e inclusive o uso de reagentes, meios de cultivo e vidraria. No laboratório é necessário um controle de reservas de equipamentos, para realização das aulas praticas e outros projetos que podem vir a ser executados. Um aplicativo mobile, como resultado desse trabalho, busca fornecer apoio à organização do laboratório de fermentação, aonde são realizadas diversas aulas e experimentos com os recursos e equipamentos da universidade. A aplicação foi feita com o framework Ionic, o qual usa tecnologias web html, css e javascript, possibilitando o desenvolvimento multiplataforma com um único código e com o AdonisJS, o qual cria o web service para fazer a comunicação do aplicativo com o banco de dados. Para familiarização, entendimento das atividades e rotina do laboratório, foram aplicadas técnicas de levantamento de requisitos, como etnografia que consiste em conhecer o ambiente do projeto, acompanhando suas atividades em alguns dias e tentar filtrar as maiores necessidades do usuário e entrevista com o professor responsável pelo laboratório, o qual ressaltou as maiores dificuldades encontradas para manter a organização do laboratório. Buscando facilitar o gerenciamento de reservas, controle dos materiais utilizados nas atividades praticas e nas funções administrativas criou-se uma aplicação que atendesse as dificuldades encontradas.

**Palavras-chave:** Aplicativo. IONIC. Desenvolvimento Híbrido. Gestão de laboratório de química.

## ABSTRACT

SERPA, Caio; GUIMARÃES, Rafael. **Applicative for the organization of the fermentation laboratory of UTFPR-PG:** 2019. 59 páginas. Final paper (Bachelor in Computer Science) - Federal Technological University of Paraná. Ponta grossa, 2019.

In the environment of the Federal Technological University of Paraná there is a fermentation laboratory that is used for practical classes and other experiments managed by students who are subject to equipment reserves and even the use of reagents, culture media and glassware. In the laboratory it is necessary a control of equipment reserves, for the realization of practical classes and other projects that may be executed. A mobile application, as a result of this work, seeks to provide support to the organization of the fermentation laboratory, where several classes and experiments are carried out with university resources and equipment. The application was made with the Ionic framework, which uses html, css and javascript web technologies, enabling multiplatform development with a single code and AdonisJS, which creates the web service to communicate the application with the database. For familiarization, understanding of the activities and routine of the laboratory, requirements gathering techniques were applied, such as ethnography, which consists of knowing the project environment, monitoring its activities in a few days and trying to filter the user's greatest needs and interviewing the responsible teacher. by the laboratory, which highlighted the greatest difficulties encountered in maintaining the organization of the laboratory. Seeking to facilitate the management of reserves, control of materials used in practical activities and administrative functions, an application was created to meet the difficulties encountered.



**Keywords:** Applicative. IONIC. Hybrid Development. Chemistry Laboratory Management.

## LISTA DE ILUSTRAÇÕES

## LISTA DE QUADROS

## LISTA DE ABREVIATURAS

| | |
|---|---|
| APP | Aplicativo |
| LAB | Laboratório |
| SGBD | Sistemas de Gestão de Base de Dados |

## LISTA DE SIGLAS

| | |
|---|---|
| API | Application Programming Interface |
| CRUD | Conhecimento Organizacional |
| CSS | Create, Read, Update, Delete |
| HTML | Hyper Text Markup Language |
| JSON | JavaScript Object Notation |
| MVC | Model View Controller |
| ORM | Object Relational Mapping |
| SPA | Single-page application |
| S.O | Sistema Operacional |
| UTFPR | Universidade Tecnológica Federal do Paraná |
| XML | Extensible Markup Language |

**SUMÁRIO**

# 1 **INTRODUÇÃO**

O laboratório de fermentação da UTFPR, campus Ponta Grossa, tem suas atividades manejadas por um professor, por alunos da instituição e usuários externos do campus, que utilizam dos equipamentos que a UTFPR possui, por serem escassos na região.

Para realização das atividades no laboratório, é possível cada usuário realizar reserva de equipamentos para uso em experimentos, informar a quantidade de reagentes e meios de cultivo usados nas aulas ou fora delas, já que é possível o acesso ao laboratório por estagiários e usuários externos a UTFPR.

Os equipamentos presentes para reserva no laboratório são: Fluxo Laminal, Shaker e Centrifuga.

O laboratório mantém os registros e reservas de reagentes, equipamentos e meio de cultivo por manuscritos, cada equipamento possui uma folha de reserva para uso, reagentes e meios de cultivo também possuem folha de reserva para controlar a quantidade usada. Ocasionalmente, podem ocorrer danos físicos que comprometam as folhas de registros, perdendo a quantidade usada de reagente ou um equipamento com duas folhas de reservas, como resultado de uma perda momentânea de uma folha de registro, fato esse diagnosticado pelos usuários.

Esses problemas que implicam em dificuldades para manter a organização do laboratório serão abordados no estudo de caso presente no levantamento de requisitos.

## 1.1 OBJETIVO GERAL

Desenvolver um aplicativo móvel para gerenciamento do laboratório de fermentação da Universidade Tecnológica Federal do Paraná, campus Ponta Grossa.

## 1.2 OBJETIVOS ESPECÍFICOS

- Revisão de literatura;
- Levantamento e Análise de requisitos;

- Análise e seleção das ferramentas para desenvolvimento do sistema.
- Possibilitar a criação, atualização, consulta e remoção de equipamentos, reagentes e meios de cultivo;
- Liberar usuários somente com autorização do Administrador;
- Permitir cadastro de equipamentos por usuários externos;
- Controlar reservas de equipamentos;
- Controlar uso de reagentes e meios de cultivo;
- Possibilitar aos usuários relatarem possíveis danos/quebras de vidrarias.

## 1.3 JUSTIFICATIVA

Regularmente ocorre no laboratório de fermentação a duplicação de dados quanto a reserva, rasuras em folhas para atualizar quantidade de reagente e perca destas folhas, o que prejudica acentuadamente nas atividades diárias, ou até mesmo a não realização de uma aula programada, devido a falta de um recurso que foi atualizado de maneira equivocada, o que pode causar grandes perdas no desempenho acadêmico da turma. Fato relatado pelo professor. Justifica-se desta forma o desenvolvimento de um aplicativo de apoio a gerência e as atividades diárias do laboratório.

## 1.4 ORGANIZAÇÃO DO TRABALHO

No capítulo 1 é feita a introdução que visa contextualizar o que será abordado, objetivos gerais e específicos a serem concluídos. Já no capítulo 2 é onde está todo o referencial teórico, junto com as tecnologias utilizadas para o desenvolvimento da aplicação. Seguindo para o capítulo 3, tem-se o desenvolvimento prático do projeto, como o levantamento de requisitos, modelagem do banco de dados e a criação do aplicativo. Nos capítulos seguintes encontram-se a conclusão, trabalhos futuros e referências, respectivamente.

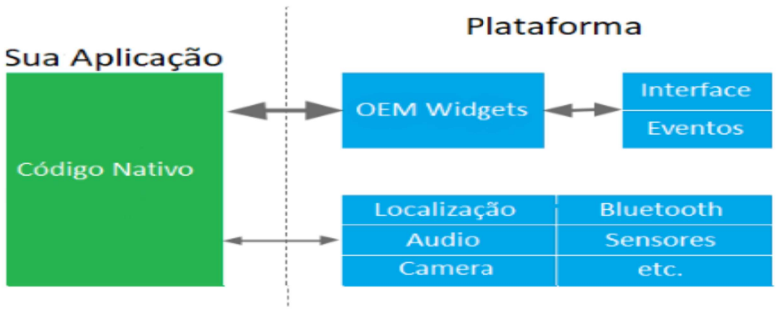
# 2 REFERENCIAL TEÓRICO

Neste capítulo abordaremos sobre duas metodologias de desenvolvimento, nativo e híbrido e as demais utilizadas para o desenvolvimento da aplicação.

## 2.1 METODOLOGIA NATIVA

Aplicação nativa/embarcada é um software desenvolvido para executar em uma plataforma específica (Miszura e Divino, 2012). A implementação de um aplicativo começa pelo código. Na implementação de um aplicativo nativo, é necessário uma linguagem nativa da plataforma, como Objective-C para iOS (http://www.apple.com/ios) da Apple ou Java para Android (http://www.android.com) do Google. A arquitetura para desenvolvimento é a mesma para as diferentes plataformas, porém cada uma possui seu kit de desenvolvimento, a Figura 1 demonstra essa arquitetura.

**Figura 1 - Comunicação de desenvolvimento nativo com a plataforma**




**Fonte: Alterado (Leler, 2017)**

Do lado esquerdo temos um código javascript e uma ponte, representando um código nativo que para executar na sua plataforma específica, faz a interação

com os eventos e interface através dos ***Widgets*** (ferramentas) e faz a comunicação direta com os recursos do aparelho, como a câmera.

A tabela abaixo apresenta algumas plataformas atuais do mercado e suas linguagens nativas.

**Quadro 1 - Plataforma e suas linguagens de desenvolvimento**

| Plataforma | Linguagem |
|---|---|
| Apple iOS | C, Objective C |
| Google Android | Java (Harmony flavored, Dalvik VM) |
| Rim BlackBerry | Java (J2ME flavored) |
| Symbian | C, C++, Phyton, HTML/CSS/JS |
| Windows Mobile | .NET |
| Windows 7 Phone | .NET |
| HP Palm webOS | HTML/CSS/JS |
| MeeGo | C, C++, HTML/CSS/JS |
| Samsung bada | C++ |

**Fonte: Autoria própria**

## 2.2 METODOLOGIA HIBRIDA

As tecnologias mais utilizadas para esse desenvolvimento híbrido são HTML, CSS e JavaScript (MADUREIRA, 2017, n.p.). Partindo desse conceito de junção dessas principais tecnologias, a utilização das linguagens de programação da web, resultará no funcionamento de um aplicativo em diferentes plataformas(VENTEU, 2018).

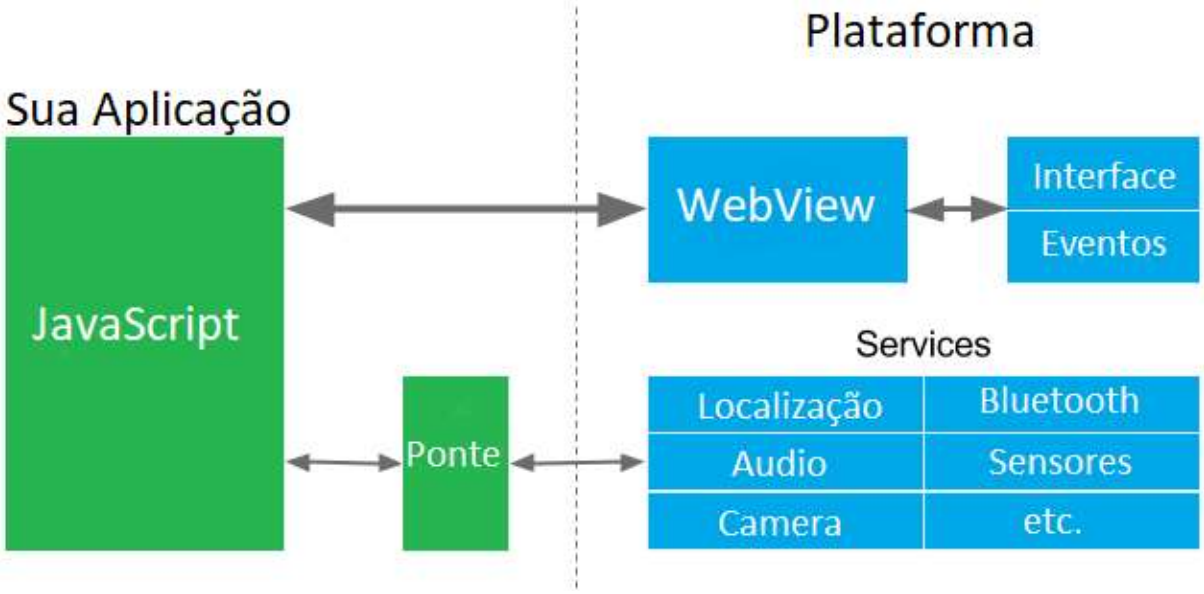
Nos últimos anos, a abordagem de aplicativos móveis híbridos cresceu e se tornou uma alternativa emergente e promissora para a abordagem de desenvolvimento de aplicativos nativos (Huynh, 2017).

A abordagem se trata de uma única codificação, que possa ser executada em diversas plataformas. O desenvolvimento híbrido ajuda na gestão de uma das

mais problemáticas métricas no desenvolvimento de aplicativos móveis: portabilidade (Malavolta e Ruberto, 2015), tradução nossa).

A Figura 2 apresenta a arquitetura, do lado da aplicação e do lado da plataforma onde a aplicação vai rodar, para desenvolvimento de um aplicativo híbrido, bastando apenas uma codificação para varias plataformas.

**Figura 2 - Comunicação de desenvolvimento híbrido com a plataforma**




**Fonte: Alterado (Leler, 2017)**

Do lado esquerdo um código ***HTML*/*CSS*/*JavaScript*** que utiliza ***webviews*** ( visualizações na web) para interação com interfaces e eventos e uma ponte, que neste trabalho será utilizado o Cordova, para fazer a comunicação do nosso código fonte com os serviços do aparelho.

## 2.3 METODOLOGIA NATIVA X HIBRIDA

É preciso dominar duas linguagens de programação totalmente diferentes para criar um ***app*** (aplicação) que rode, por exemplo, em Windows e iOS (Cronapp, 2018). Um ***app*** híbrido é mais simples e rápido de se desenvolver, porém, pode não tem a mesma rapidez de um ***app*** nativo (Guedes, 2017).

O desenvolvimento nativo faz a comunicação direta do aplicativo com os recursos do aparelho, proporcionando, em teoria, um melhor desempenho, mas, na

prática, isso é difícil de afirmar, já que as ferramentas de desenvolvimento otimizam essa performance ao máximo (Cronapp, 2018)

Sendo o custo uma métrica crucial no desenvolvimento de um software, o aplicativo híbrido leva vantagem por ser reconhecido por várias plataformas, evitando despesas elevadas com desenvolvedores específicos para cada plataforma. O ***app*** pro laboratório não requisitará recursos gráficos avançados, e como o esperado é atender a todos os usuários sem saber se existe uma homogeneidade dos aparelhos, por tempo de desenvolvimento e praticidade de programação o aplicativo será desenvolvido de modo híbrido.

## 2.4 TECNOLOGIAS UTILIZADAS

Tópico destinado a apresentar as tecnologias utilizadas no desenvolvimento do projeto.

Á tabela abaixo contem as tecnologias utilizadas para o desenvolvimento do projeto.

**Quadro 2-Tecnologias utilizadas**

| Ferramenta/ Tecnologia | Descrição |
|---|---|
| ADONISJS | ***Framework*** NodeJs que utiliza JavaScript como linguagem base e fornece uma estrutura MVC, que possibilita, de forma facilitada o desenvolvimento de aplicações ***Web*** do lado do servidor através da criação de API's |
| ANGULAR | Angular é uma plataforma e ***framework*** para criar aplicações em ***HTML*** e ***TypeScript*** (Angular, 2019). O Angular possibilita um desenvolvimento mais ágil por encapsular diversas funções utilizadas na criação da maioria das aplicações. |
| CORDOVA | Estrutura de desenvolvimento móvel ***open source***, que disponibiliza ***plugins*** para a comunicação de um ***App*** com componentes nativos do celular, essa comunicação é feita através de API's (Cordova, 2019). |
| CSS | ***CSS*** (***Cascading Style Sheets***) é uma linguagem de estilo utilizada informar como os elementos, criados anteriormente pelo |

| | |
|---|---|
| | ***HTML***, serão mostrados na tela. Em outras palavras o ***CSS*** possibilita estilizar os componentes contidos na página web, adicionando cores, fontes, alinhamento, margem, sombra, etc.., o que torna a visualização de um web site muito mais agradável (MDN 2019). |
| HTML | O ***HTML*** (***Hyper Text Markup Language***) é uma linguagem de marcação de texto utilizada para informar ao navegador como a estruturação da sua página deve ser feita e exibida aos usuários. Essas marcações consistem em envolver conteúdos (normalmente texto) em ***tags*** (palavra-chave) para que se comportem de uma maneira específica em ***web sites***. |
| IONIC | ***Kit*** de ferramentas de interface do usuário de código aberto para a criação de aplicativos móveis e de ***desktop*** de alto desempenho e de alta qualidade, usando tecnologias da ***Web*** (***HTML, CSS e JavaScript***) (Ionic, 2019) ou seja, ele é responsável por criar toda a interface de interação com o usuário. |
| JAVASCRIPT | Segundo Rodrigues (2019), ***JavaScript*** é uma linguagem de programação interpretada, inicialmente criada para que ***scripts*** (instruções) pudessem ser executados diretamente no ***browser*** sem a necessidade de passar por um servidor. Ela é responsável por realizar toda a interação com usuários e também realizar a manipulação da árvore de elementos do ***HTML*** também chamada de ***DOM*** (***Document Object Model***). |
| LUCID | O Lucid é um ***ORM***[1] criado pelo AdonisJS, para facilitar a consulta no banco de dados, pois não utiliza SQL para essas funcionalidades, ao invés disso as consultas são feitas através de funções ***JavaScript***, tornando muito mais simples e intuitivas tais tarefas (AdonisJS, 2019). |
| NODEJS | O Node é um interpretador de ***JavaScript open source***, que trabalha de forma assíncrona e com orientação a eventos. Ele permite escrever código em ***JavaScript*** do lado do servidor, possibilitando a criação de aplicações de alta escalabilidades |

[1]Técnica que visa mapear os objetos relacionais a fim de criar uma camada entre nosso modelo de objetos, referente a aplicação e o modelo relacional.

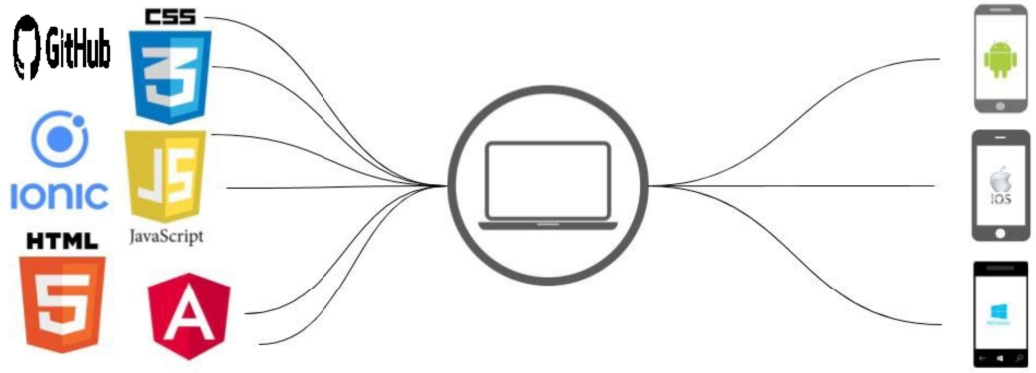
| | |
|---|---|
| | capazes de manipular milhares de requisições simultâneas (NodeJs, 2019). |
| TYPESCRIPT | ***TypeScript*** é um superconjunto de ***JavaScript*** que é compilado em JavaScript simples. Código aberto (***Typescript***, 2019). Como o ***TypeScript*** tenta não alterar o ***JavaScript*** mas sim se adaptar aos seus recursos, os desenvolvedores que trabalham com ele podem utilizar dos recursos do JavaScript além de ser uma linguagem de forte tipagem para escrever códigos mais organizados. |

**Fonte: Autoria própria**

# 3 DESENVOLVIMENTO

A tecnologia utilizada para o desenvolvimento do aplicativo será o Ionic, além de ser um ***framework open source*** (código aberto) ele possibilita criação de aplicações multiplataforma, ou seja, um único código engloba usuários com sistemas operacionais diferentes, como Android e IOS. O Ionic proporciona um desenvolvimento ágil e simples, pois utiliza tecnologias web, como ***HTML5***, ***CSS*** e ***JavaScript***. Além disso o ***framework*** adota como base o Cordova, que disponibiliza ***plugins*** para a comunicação com os recursos do sistema operacional do celular, assim como câmera, GPS, lanterna, etc. Tornando o desenvolvimento mais dinâmico. A figura a baixo mostra como funciona a integração das tecnologias junto de um ***web service*** para atingir diferentes plataformas.

**Figura 3-Exemplificação das tecnologias para desenvolvimento hibrido**




**Fonte: Autoria Própria**

## 3.1 REQUISITOS

A descoberta de requisitos (às vezes, chamada elicitação de requisitos) é o processo de reunir informações sobre o sistema requerido e os sistemas existentes e separar dessas informações os requisitos de usuário e de sistema (SOMMERVILLE, 2011).

Foram usadas as técnicas de entrevista, etnografia e ***brainstorming*** (tempestade de ideias) para criar um levantamento de requisitos, o que gerou o estudo de caso a seguir:

“O laboratório de fermentação da UTFPR-PG está com um problema de organização no uso dos materiais e equipamentos pelos alunos. Todas as informações são feitas por manuscrito, causando uma desorganização.

O aplicativo contém dois tipos de acesso, um administrador o qual ficará responsável por gerenciar o aplicativo (professor) e seus usuários (alunos e pessoas externas), que podem efetuar cadastro (autorizado pelo administrador), fazer reservas dos equipamentos, informar uso de reagentes e meio de cultivo.

O administrador deve ser capaz de manter:

- Os equipamentos, os quais serão feitas reservas;
- Reagentes e meio de cultivo para uso;
- Autorizar cadastro de usuários.

Para um usuário poder realizar reservas de equipamentos, informar uso de reagente e meio de cultivo, um cadastro prévio (pois necessita autorização do ADM) deve ser realizado no aplicativo informando seu Nome, Senha, E-mail, RA, Instituição, Nome do aluno, Orientador.

Todo usuário ao fazer uma reserva de equipamento, deve informar:

- Nome;
- Nome orientador;
- Curso;
- Instituição;
- Data;
- Horário de início e término.

Para informar o uso de reagentes e meios de cultivo é necessário:

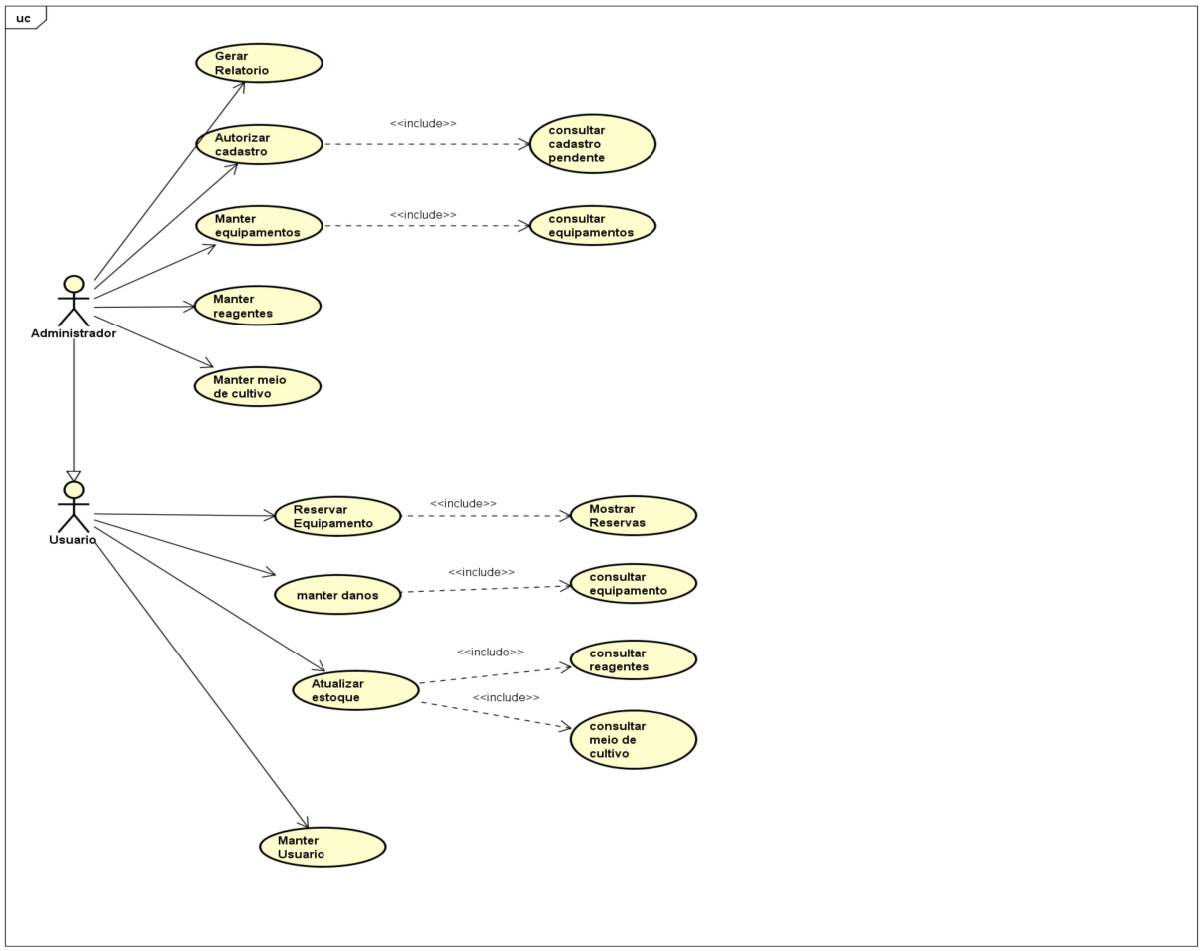
- Numeração (Código);
- Nome do (reagente ou meio);
- Quantidade.

Quando o usuário informa o uso de reagentes e meios de cultivo, deve ser dado baixa no estoque para controle. Quando atingir um limite mínimo do material X, deve ser informado na lista de materiais um aviso ou alerta dizendo que tal material precisa ser reposto. O aplicativo deve gerar para o professor um relatório com o uso dos reagentes, meios de cultivo e vidrarias (quantas tem de cada vidraria e se houve

alguma perda por dano). O usuário deve ter um espaço para informar se houve alguma vidraria quebrada. Feitos os levantamentos foi criado o Diagrama de casos de uso, apresentado na Figura 4. Existem dois atores no diagrama, o ator administrador e o ator usuário e entre eles existem uma relação de generalização, onde o administrador herda os casos de uso do usuário, então todo administrador é um usuário. Os casos de uso do usuário são reservar equipamentos onde é necessário mostrar as reservas já feitas para não haver conflitos, manter danos onde é feito o relato de danos as vidrarias, atualizar estoque onde é informado uma quantidade de reagente ou meio de cultivo utilizada e o manter usuário, onde o usuário pode se cadastrar, atualizar suas informações e excluir a conta se necessário. Do ponto de vista do ator administrador, tem-se o caso de uso gerar relatório, onde o administrador pode gerar relatórios de entrada e saída de reagentes, meios de cultivo e vidrarias.

No caso de uso autorizar cadastro possui um include (informando que quando o autorizar cadastro for chamado sempre será chamado o caso de uso consultar cadastro pendentes). Manter equipamentos é o caso de uso responsável pelo cadastro de novos equipamentos, atualizações nos equipamentos já cadastrados e exclusão dos mesmos se necessário.

**Figura 4 - Diagrama de Caso de uso**




**Fonte: Autoria própria**

Os casos de uso Manter reagentes e Manter meios de cultivo possuem semelhanças, sendo os dois para atualizar, criar e excluir reagentes e meios de cultivo, respectivamente.

## 3.2 BACK-END

Para a criação do ***back-end***, que compõe o banco de dados, todos os registros que são essenciais para o funcionamento do aplicativo, acompanhado da regra de negócio, foi utilizado o ***framework*** AdonisJs no desenvolvimento de um ***Web Service*** (Serviço de internet) que disponibiliza uma ***API*** para a troca de informações entre servidor e aplicação.

## 3.3 WEB SERVICE

O ***Web Service*** é uma camada localizada entre o cliente e o servidor. Ela é responsável por receber uma requisição do cliente, repassar essa requisição ao servidor, que devolve uma resposta ao ***Web Service*** e então essa resposta é repassada ao cliente.

Essa requisição é realizada através de um protocolo de comunicação chamado ***HTTP*** (Protocolo de Transferência de Hipertexto Seguro).

Segundo (Vieira, 2007) o protocolo ***HTTP*** é um texto estruturado utilizado para enviar e receber informações na web, esse texto é composto por diversos campos contendo as informações sobre a requisição realizada.

A principal vantagem na utilização de ***Web Services*** para a construção da base de dados, é que, por fornecer um ***JSON*** como resposta, qualquer aplicação pode consumir tal serviço, então o mesmo ***Web Service*** que alimenta uma aplicação Web também poderá alimentar um Smartphone ou um Tablet por exemplo, tornando a aplicação altamente escalável para os usuários, que podem acessar o aplicativo de diferentes plataformas físicas e diferentes sistemas operacionais.

Na Figura 5 é mostrado a arquitetura de funcionamento do ***Web Service***. No corpo dessa requisição normalmente são enviados ***JSON's*** (notação de objeto javascript) ou um ***XML*** (linguagem de marcação extensível) com os dados enviados por uma requisição feita na aplicação para ser interpretada e processada pelo banco de dados.

**Figura 5 - Arquitetura Web Service**


**Fonte: (PhpFlow, 2017)**

## 3.4 MODELAGEM DO BANCO DE DADOS

Após o levantamento de requisitos do sistema e a elaboração do diagrama de caso de uso foi possível o desenvolvimento da modelagem do banco de dados, ilustrado na Figura 6, através do Modelo Relacional.

Onde são identificados as tabelas que compõe o banco de dados bem como seus atributos e relacionamentos, essenciais para a construção e funcionamento do sistema. Cada tabela contem um nome, identificado em negrito no top. Em seu interior são armazenados os nomes dos atributos que a pertencem, seguido do tipo (valor inteiro, texto, número decimal) que esse atributo pode assumir.

As linhas que ligam uma tabela a outra são chamadas de relacionamentos, onde o símbolo contido no final de cada linha representa a cardinalidade entre as tabelas, podendo ter relacionamentos 1:1 (dois tracos de um lado da linha e dois traços do outro), 1:n (dois traços de um lado e “pé- de galinha” do outro), ou relacionamentos n:n, onde é gerado uma terceira tabela (ilustrado pelo relacionamento entre Usuário e Recursos, que gerou a tabela “Usuario_Recurso”) e um relacionamento 1:n é criado com essa nova tabela gerada. Um relacionamento 1:1 (lê um pra um) é feito quando uma tabela só pode se relacionar de forma unitária com outra, um exemplo seria uma tabela Pessoa com outra tabela CPF, então uma pessoa só pode ter um CPF e um CPF só pode pertencer a uma pessoa.

Um relacionamento 1:n (lê um pra n), é quando uma tabela pode se relacionar de forma múltipla com outra tabela, porém essa segunda se relaciona de forma unitária com a primeira, um exemplo seria a tabela Pessoa e a Tabela telefone (celular), onde um telefone só pode pertencer a uma pessoa porém a pessoa pode ter mais de um telefone.

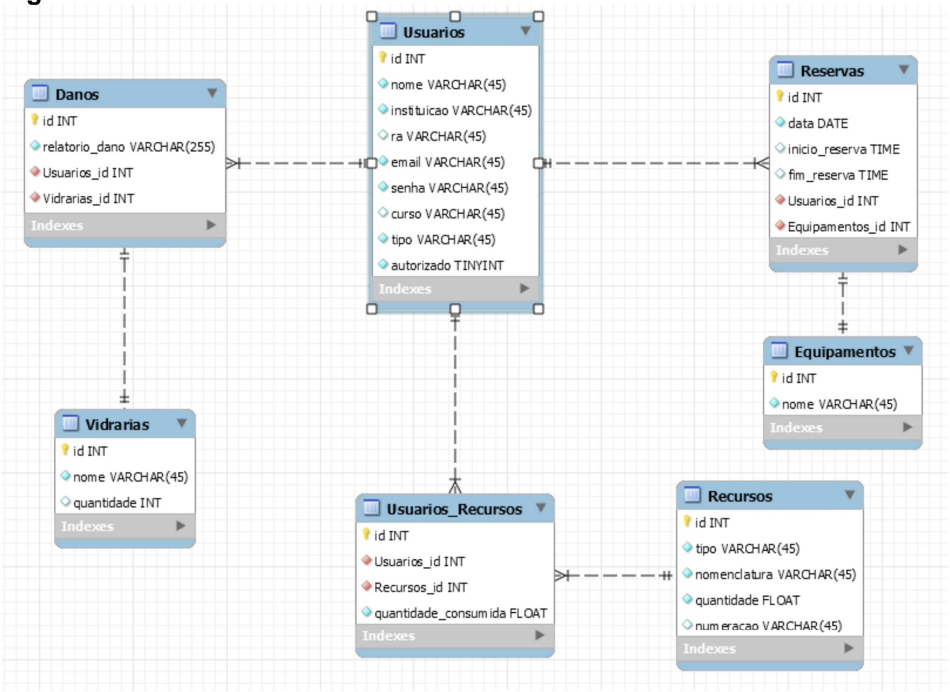
E o último caso é onde ocorre o relacionamento n:n (lê n para n ou muitos para muitos), onde uma tabela pode se relacionar de foma múltipla com outra tabela, o que acaba gerando uma terceira tabela mantendo o modelo dentro das formas normais.

Essa terceira tabela, como citado anteriormente, vai possuir dois relacionamentos de 1:n com as duas tabelas inciais que a geraram.

Na Figura a baixo, é representado o modelo relacional do nosso projeto, feito com base no estudo de caso levantando com o cliente, mostrado nos requisitos da aplicação.

Como já destacado, a tabela Usuarios_Recursos foi gerada a partir de um relacionamento n:n entre as tabelas usuários e recursos.

**Figura 6-Modelo Relacional**




**Fonte: Autoria Própria**

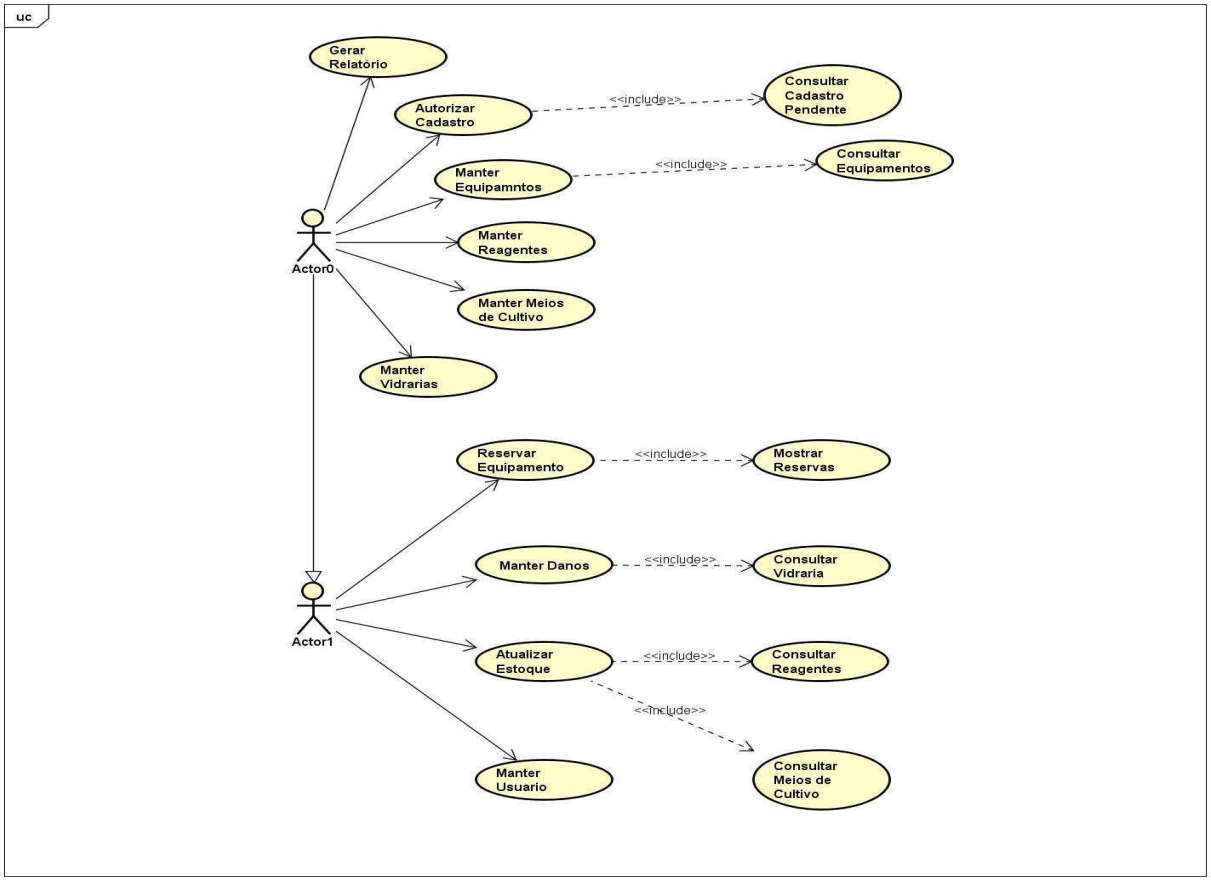
## 3.5 DESENVOLVENDO A APLICAÇÃO

Do estudo de caso presente no levantamento dos requisitos, item 3.1 desse documento, foram atualizadas as seguintes informações, após reuniões e discussões com cliente:

- Criação de controle de vidraria, possuindo cadastro com nome e quantidade.
- Disponibilizar criação de relatório para vidrarias.
- Professor não fará cadastro de alunos ficando apenas com a função de autorizar os pré-cadastrados.

Foram feitas atualizações no banco de dados onde surgiram atualizações no modelo relacional e no diagrama de caso de uso.

**Figura 7 - Diagrama de caso de uso após as atualizações**




**Fonte: Autoria própria**

No diagrama de caso de uso, onde pode-se notar a criação de um caso de uso chamado manter vidraria, para fazer a realização da criação, atualização, acesso e exclusão as vidrarias do sistema.

A atualização do diagrama de caso de uso, onde o actor0 é o Administrador e o actor1 é o Usuário.

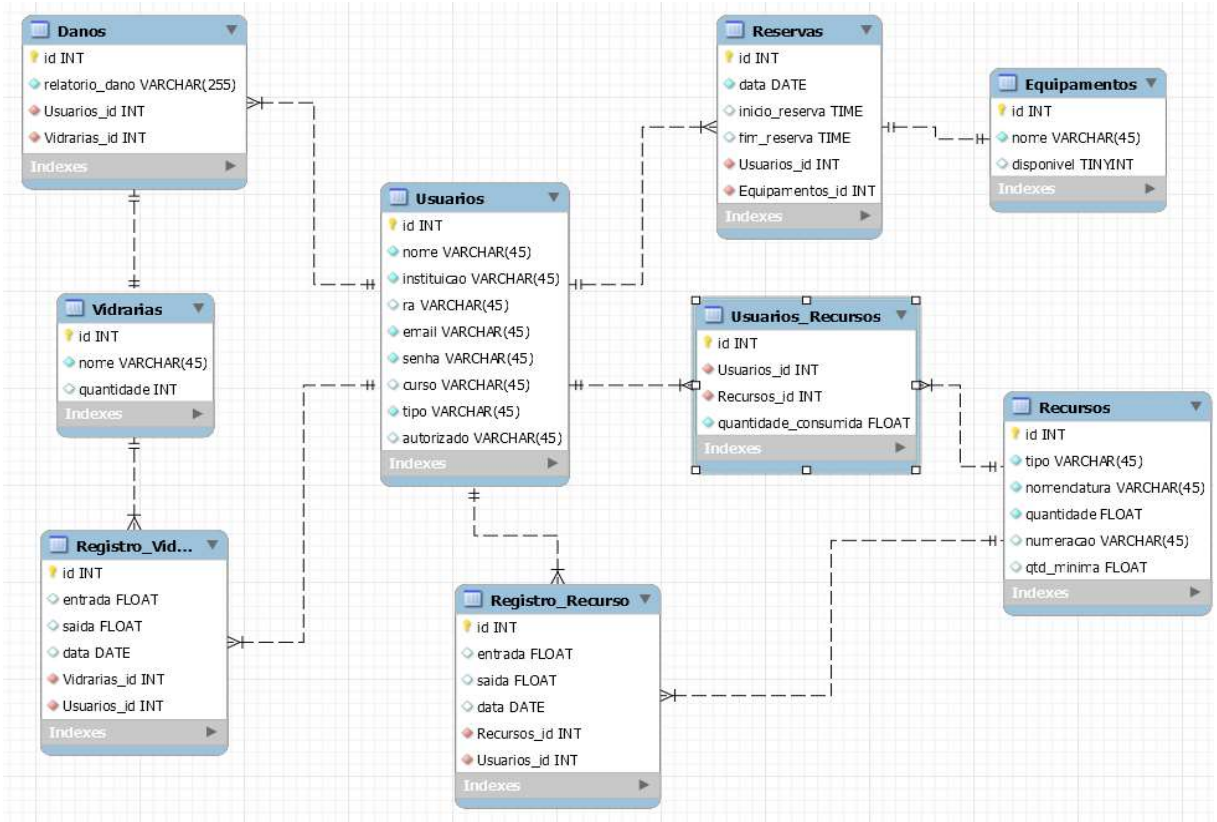
Sendo assim, pela relação deles de generalização representado pela seta saindo do actor0 para o actor1 o administrador possui os casos de uso do usuário também.

Das atualizações no modelo relacional, a criação de duas novas tabelas foram necessárias para manter o modelo dentro das formas normais.

A tabela Registro_vidraria, que foi gerada a partir de uma relação n:n entre as tabelas vidrarias e usuários e também foi gerada a tabela Registro_Recursos, tal

tabela que foi gerada a partir de uma ligação n:n entre as tabelas usuários e recursos.

**Figura 8 - Modelo relacional após as atualizações**




**Fonte: Autoria própria**

Ambas as tabelas foram geradas pelo mesmo motivo da geração da tabela Usuarios_Recursos, que já estava presente do modelo, a fim de manter funcional o modelo relacional, usando o conceito de normalização introduzido por Edgar F. Codd em 1970.

Normalização de dados é o processo formal e passo a passo que examina os atributos de uma entidade, com o objetivo de evitar anomalias observadas na inclusão, exclusão e alteração de registros (Kelyn Schenatto, 2015).

### 3.5.1 INSTALAÇÃO E CONFIGURAÇÃO DO ADONISJS

Para desenvolver a ***API*** que alimentará a aplicação será feita a instalação do Adonis, porém sendo um ***framework*** Node, será necessário ter instalado

anteriormente o NodeJs e o npm (gerenciador de pacotes que já vem junto com o Node quando é instalado). Caso o Node não esteja instalado é possível obter o mesmo indo até o site “nodejs.org” e realizar o download. Com posse do Node, o Adonis pode ser instalado em poucos passos.

No prompt de comando insira o seguinte comando:

**npm i -g @adonisjs/cli**

Esse comando instalará de forma global (parâmetro -g) o Adonis, com a interface de linha de comando (cli). Agora basta ir até um diretório onde deseja criar seu projeto e inserir o comando:

**adonis new nomedoprojeto**

Isso criará toda a estrutura de pastas que o Adonis disponibiliza para a iniciação do projeto. Adicionando o comando a baixo o servidor já está disponível em “127.0.0.1:3333”.

**adonis serve –dev**

**Figura 9 - SGBD’s suportados pelo Adonis**

| Database | NPM Driver |
|---|---|
| MariaDB | `npm i mysql` or `npm i mysql2` |
| MSSQL | `npm i mssql` |
| MySQL | `npm i mysql` or `npm i mysql2` |
| Oracle | `npm i oracledb` |
| PostgreSQL | `npm i pg` |
| SQLite3 | `npm i sqlite3` |

**Fonte: (Adonis, 2019)**

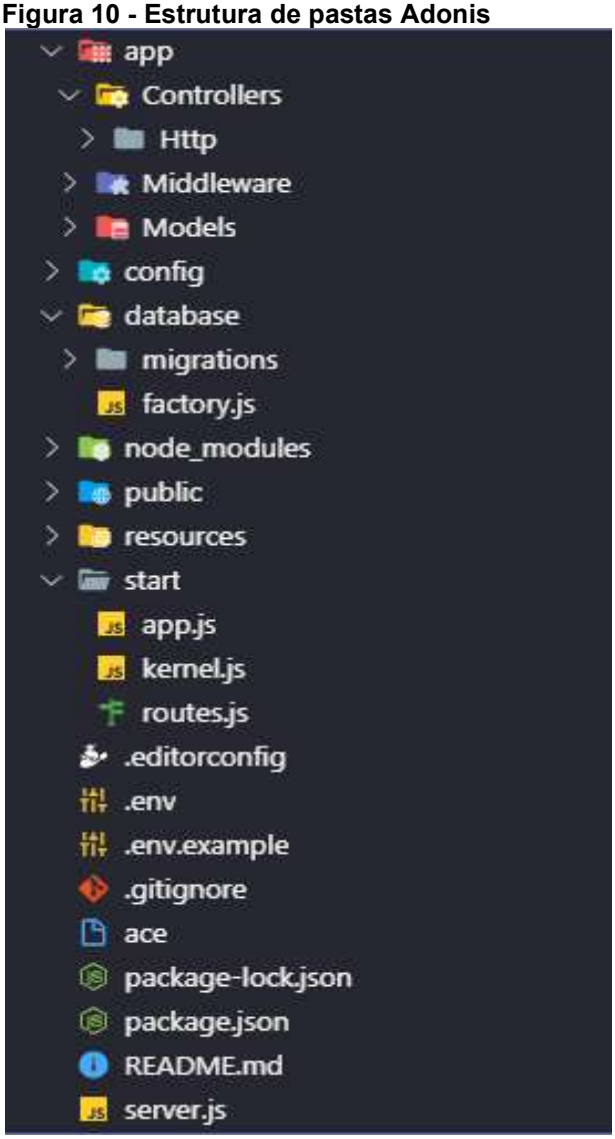
Agora basta escolher um dos SGBD’s da Figura 9 e disparar o comando correspondente a ele.

No arquivo “.env” localizado no diretório raiz do projeto deve ser inserido os dados de conexão com o banco, tais como usuário, senha, host, porta, database. Agora a conexão com o banco pode ser feita com sucesso.

### *3.5.1.1* ESTRUTURA DE PASTAS DO ADONISJS

Estrutura de pastas geradas pelo Adonis na inicialização do projeto. Apenas será abordado os diretórios relevantes ao projeto neste tópico.

**Figura 10 - Estrutura de pastas Adonis**




**Fonte: Autoria própria**

No diretório raiz temos alguns arquivos importantes. O primeiro é o ".env", citado anteriormente, onde ficam armazenados as variáveis globais responsáveis pelo funcionamento do projeto, assim como o tipo de conexão, usuário e senha do bando de dados, porta, host. Ainda no mesmo diretório temos os arquivos "package.json" e "package-lock.json" onde são armazenadas todas as dependências do projeto com as versões utilizadas no mesmo. Outros diretórios que foram utilizados dentro do projeto são:

• Diretório “./Start”, mais especificamente no arquivo “routes.js”, onde estarão armazenadas todas as nossas rotas. As rotas são responsáveis por determinar a comunicação com o servidor, onde cada rota busca uma informação específica e distinta;

• No diretório “./Database/Migrations”, onde está armazenadas todas as ***Migrations*** (migrações), que nada mais são do que o esquema do banco de dados, em outras palavras, cada ***Migrations*** representam uma tabela no banco de dados, contendo seus atributos e os tipos de dados de cada atributo possa assumir;

• No diretório “./App/Models”, responsável por guardar nosso ***Models*** (modelos) que determinam as relações entre as tabelas;

• E por último, no diretório “./App/Controllers/Http” estarão os ***Controllers*** (Controles) da nossa ***API***, que conterá todas as funções com as regras de negócio.

#### *3.5.1.2* CRIAÇÃO DA ESTRUTURA DO BANCO

Após as configurações iniciais e o entendimento da estrutura do projeto, pode ser realizado o desenvolvimento da ***API***. Inicialmente foram criadas as tabelas no banco de dados, para isso o seguinte comando deve ser efetuado:

**adonis make:model nometabela -c -m**

Este comando é responsável por criar o ***Model*** com o nome passado, porém os parâmetros “-c” e “-m” já criam o ***Controller*** e a ***Migration*** correspondente a esse ***Model***. Para todas as tabelas do banco temos que executar o comando anterior, só então definir o “corpo” de cada componente.

### 3.5.2 INSTALAÇÃO E CONFIGURAÇÃO DO IONIC

Primeiramente é necessária a instalação do Ionic, para isso foi dado o comando no prompt de comando:

**npm install -g ionic**

Após instalação do Ionic, pode ser criado o projeto ionic com o comando base ionic ***start*** [nomeProjeto] [modelo]. Nosso projeto foi iniciado com modelo ***blank*** (vazio) com o comando:

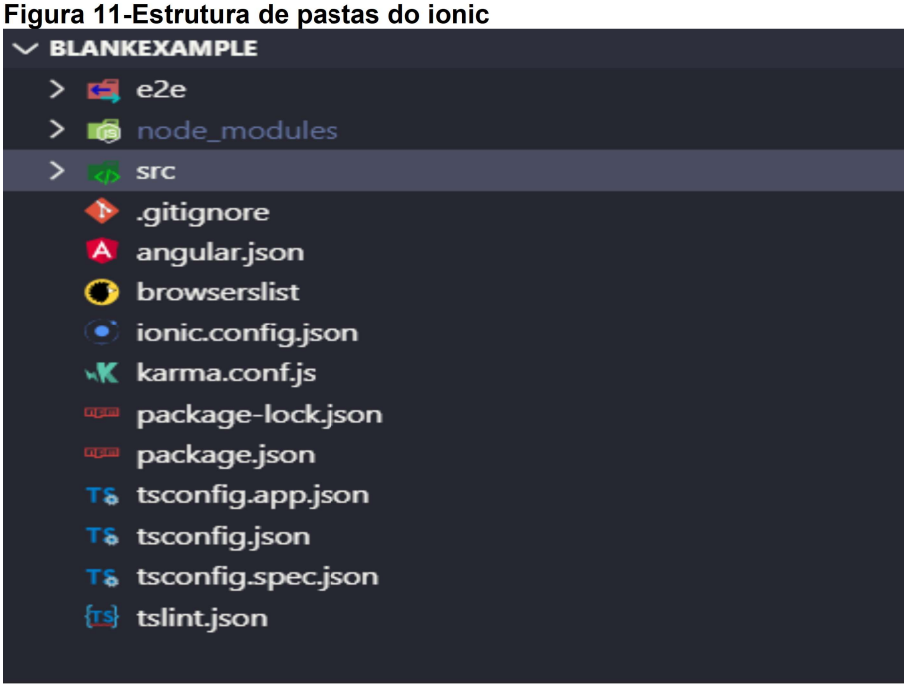
**ionic start MypApp blank**

O Ionic disponibiliza alguns componentes nativos, incluindo ***cards*** (cartas), listas, formulários e guias oferecendo total recurso para o desenvolvimento da aplicação.

### *3.5.2.1* ESTRUTURAS DE PASTAS DO IONIC

Árvore de pastas do projeto Ionic criado:

**Figura 11-Estrutura de pastas do ionic**




**Fonte: Autoria própria**

Essa é a estrutura de pastas fornecida pelo Ionic. A pasta node_modules contem todas as dependências listadas no arquivo package.json.

Foi dado maior foco a pasta src, a qual contem os arquivos ***HTML***, ***CSS*** e ***TypeScript*** do desenvolvimento do aplicativo.

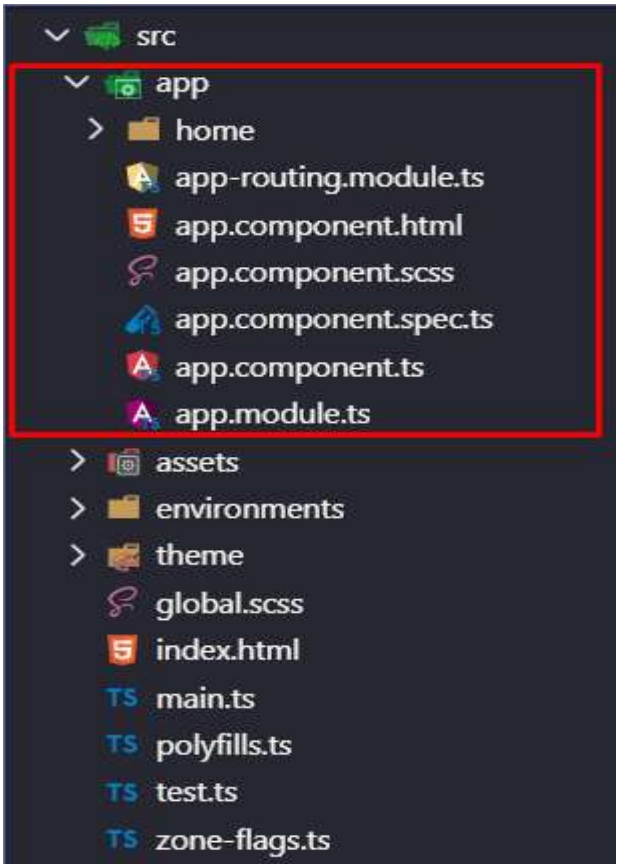
Os arquivos são fundamentais no funcionamento do aplicativo, já que são responsáveis por moldar as características visuais das telas com os arquivos ***html.***

Possibilitam também adicionar efeitos visuais com os arquivos ***css's,*** *deixando assim o aplicativo mais atrativo e confortável para o usuário utilizar.*

*E por fim, os arquivos* ***typescript,*** *que irão d*efinir como a aplicação vai reagir à iteração de um usuário com as telas e seus componentes, como por

exemplo, um botão de confirmar reserva o qual precisa enviar uma requisição para a API e fazer uma criação no banco:

**Figura 12 - Arvore de arquivos pasta src**




**Fonte: Autoria própria**

A pasta ***App*** armazena além das páginas que compõe o aplicativo, os controles, ***HTML*** e ***TypeScript*** do aplicativo na inicialização.

A pasta ***assets*** (ativos) é responsável por armazenar as imagens que serão utilizadas no projeto, como ícone do aplicativo que aparecera nos dispositivos dos usuários e as imagens de fundo da aplicação.

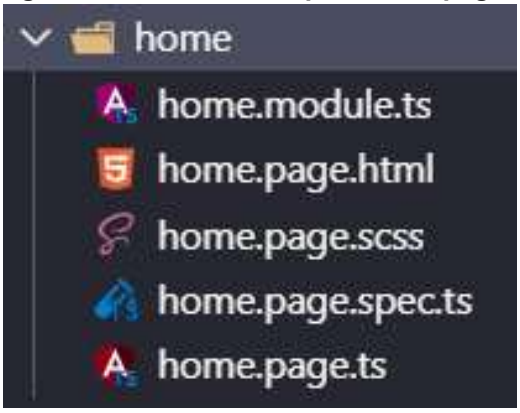
*A pasta* ***Theme*** (tema) carrega com ela configurações pré definidas de ***CSS***, o que ajuda na estilização do ***App***.

A pasta ***environment*** (meio ambiente) armazena as variáveis globais do aplicativo.

Por último, a pasta ***home*** (inicio) é criada junto com o comando de ***start***, sendo a primeira página do projeto, junto dos arquivos ***App*** responsáveis pelo controle de inicialização do aplicativo e rotas.

Estrutura da pasta ***home***, a qual será a basicamente a mesma para todas as outras páginas do projeto:

**Figura 13 - Arvore de arquivos das páginas**




**Fonte: Centro Social de Azurva (2009)**

O arquivo home.module.ts faz o controle dos ***imports*** (importações) dá página, junto com a rota de acesso para a mesma. Home.page.spec.ts é um arquivo que pode ser utilizado para alguns testes usando outras ferramentas de apoio, não foram realizados testes com esses arquivos. Home.page.sccs armazena as configurações locais de ***CSS*** para a página ***home***. Home.page.ts armazena as funções da página que podem vir a ser necessárias para responder a uma requisição do usuário. E por último a home.page.html, que armazena todo conteúdo visual da página, composta por código ***HTML*** junto com as configurações ***CSS***.

## 3.6 ATENDENDO AOS REQUISITOS

Este capítulo mostrará as funcionalidades do aplicativo que atende aos requisitos levantados no início do trabalho.

### 3.6.1 DANO

É importante que os danos às vidrarias sejam relatados pelos usuários para que o administrador do laboratório tome ciência da perca de uma unidade de vidraria para trabalhos no laboratório e solicite a compra de novas vidrarias caso seja necessário, a fim de não faltar material aos alunos durante as aulas e ter um controle maior de tudo que é danificado durante alguma atividade, já que alguns

pedidos de vidrarias demoram um tempo elevado e a reposição deve ser feita com maior antecedência possível.

**Figura 14 - Tela para relatar danos a vidraria**

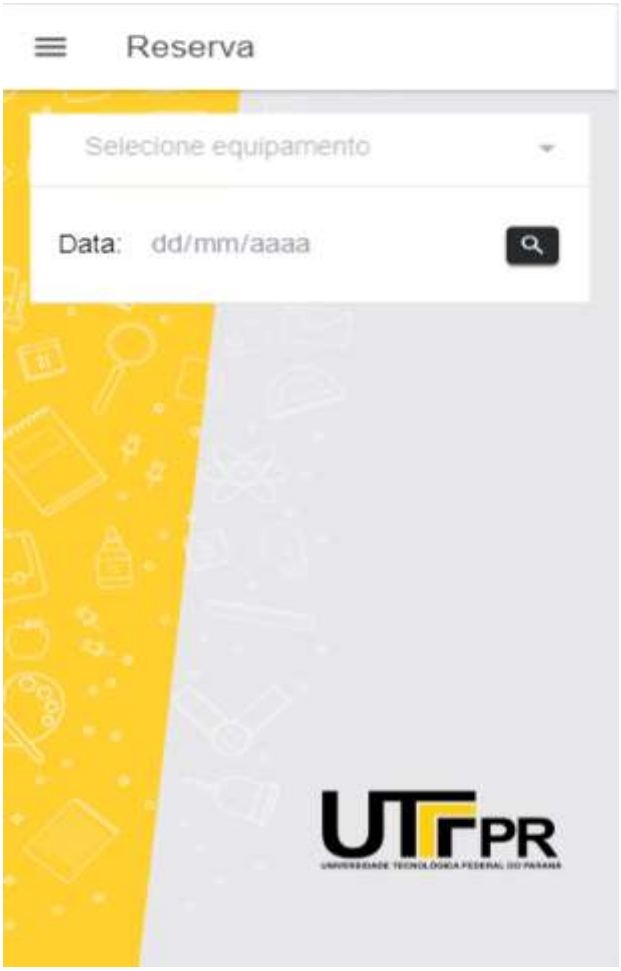

**Fonte: Autoria própria**

Na Figura 14 é ilustrada a tela para informar os danos. Logo no topo encontra-se um "***select***" com a lista de todas as vidrarias cadastradas no aplicativo. O usuário deve selecionar a vidraria em que ocorreu o dano, descrever detalhadamente o ocorrido no campo a baixo e então acionar o botão "Registrar dano" para que as informações sobre a vidraria danificada seja salva no sistema.

### 3.6.2 RESERVAS DE EQUIPAMENTOS

As reservas são destinadas apenas aos equipamentos, podendo ser reservadas por alunos da UTFPR e por alunos externos a universidade, contanto que os mesmos tenham sidos previamente cadastrados, e autorizados pelo administrador do sistema.

**Figura 15 - Tela para reserva de equipamentos**



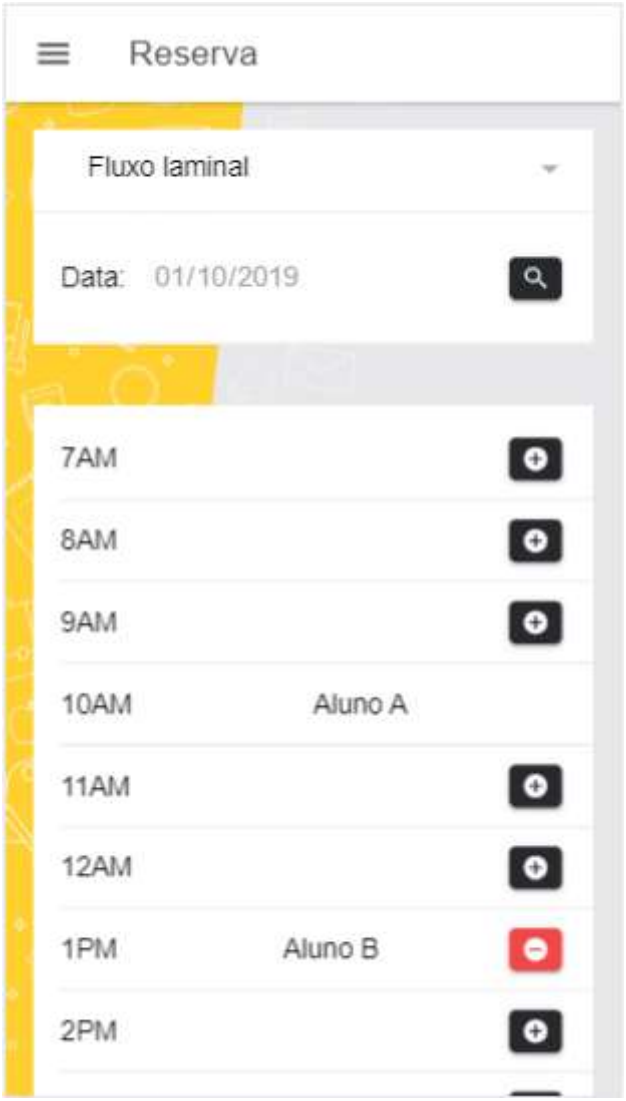

**Fonte: Autoria própria**

O primeiro campo encontra-se um "***select***" contendo a lista de equipamentos disponíveis para serem reservados.

Após escolher o equipamento desejado para uso, deve ser inserido a data em que deseja-se reservar, para que seja carregada em uma lista as reservas já realizadas e os horários disponíveis naquela data que podem vir a receber uma

reserva, mantendo mais organizada e clara a reserva de equipamentos não só para o administrador como para todos os usuários.

**Figura 16 - Tela para reserva de equipamentos**




**Fonte: Autoria própria**

O início das reservas começam as 07h00min da manhã e se enceram as 22h00min. Sendo todas as reservas de apenas uma hora. O aluno que deseja utilizar os equipamentos por mais de uma hora deverá fazer duas ou mais reservas. As reservas são feitas clicando no botão com um sinal de mais, que se encontra na mesma linha do horário em que deseja escolher.

Os horários que não estão mais disponível contém o nome da pessoa que reservou e o botão de adicionar reserva fica ocultado.

O administrador do sistema poderá clicar no nome e obter mais informações (nome completo, orientador, RA, etc) sobre o aluno que reservou o equipamento naquele horário.

O aluno que realizou a reserva terá a opção de cancelá-la assim que desejar, acionando o botão com um sinal de menos que fica visível somente para ele.

### 3.6.3 LOGIN E CADASTRO

O ***App*** tem acesso de dois tipos de usuário, aluno e professor (administrador) e cada um possui seu menu, diferenciando a opção administração, visível apenas para o professor. Ao iniciar o aplicativo o usuário é direcionado à tela de login para ter acesso ao sistema:

**Figura 17 - Tela Login**


**Fonte: Autoria própria**

Para fazer login o usuário deve informar seu e-mail e senha cadastrados que serão validados no banco de dados. Caso ainda não possua cadastro deve realizar o mesmo utilizando o botão cadastrar:

**Figura 18 - Cadastro**


**Fonte: Autoria própria**

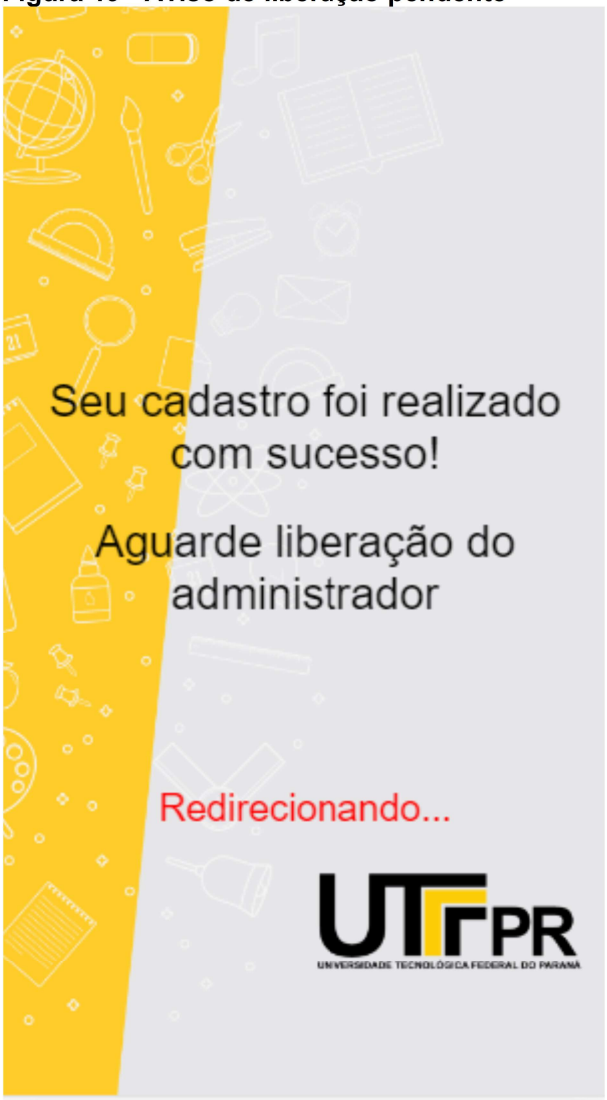
Para o cadastro foram solicitados Nome, RA, Orientador, Instituição, Curso, email e senha para o novo usuário.

Após preencher e realizar o cadastro no aplicativo, um aviso será mostrado na tela para o usuario confirmando seu cadastro no sistema, alertando da aprovação do administrador para ter o acesso as funcionalidades do aplicativo.

Sendo assim o usuário será redirecionado para a tela login novamente, ficando no aguardo da liberação do usuário administrador para concretizar o

cadastro no sistema e poder começar a usufruir das funcionalidades implantadas na aplicação.

**Figura 19 - Aviso de liberação pendente**



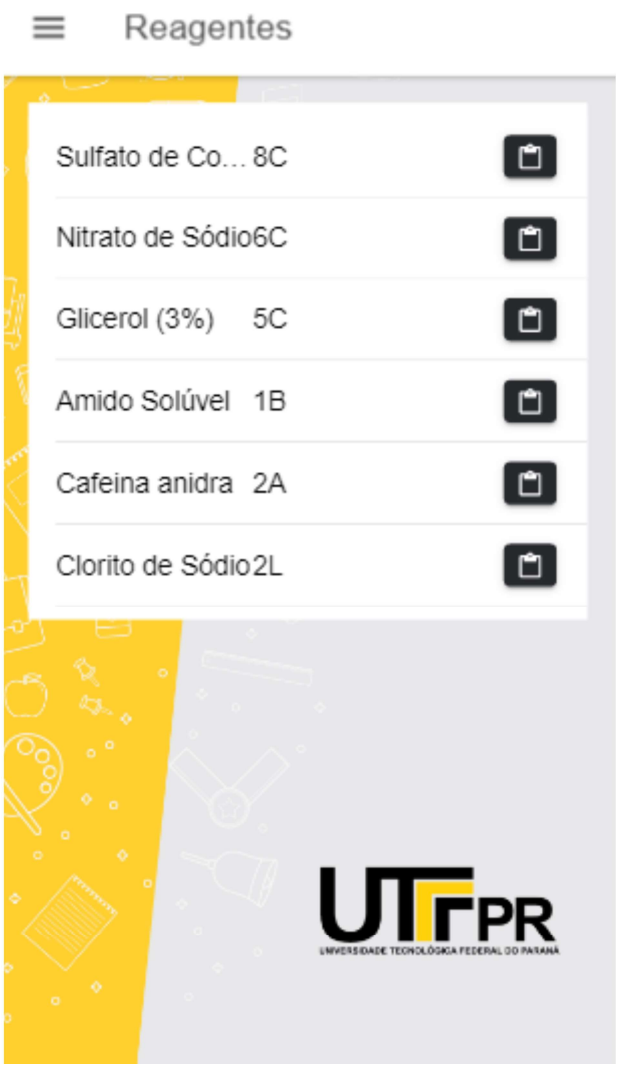

**Fonte: Autoria própria**

Ao efetuar login, o id e o tipo do usuário logado é salvo na aplicação, para ter o controle de acesso as funcionalidades restritas aquele tipo de usuário e também para recuperar, de forma facilitada, as informações do usuário quando o mesmo utilizar algum recurso do laboratório. Após isso o usuário é redirecionado para a tela de reagentes, funcionalidade mais utilizada pelos usuários segundo o cliente.

### 3.6.4 REAGENTES E MEIOS DE CULTIVO

Para informar o consumo de um reagente (ou meio de cultivo) o usuário deve selecioná-lo, clicando no ícone “prancheta” de seu respectivo item:

**Figura 20 - Lista de reagentes**




**Fonte: Autoria própria**

Após a seleção o usuário é redirecionado para a tela de recursos (Figura 23). Na tela de recursos aparecerá informações mais detalhadas sobre o item escolhido, seguido de um campo para adicionar a quantidade de recursos que foi consumida.

Após o preenchimento do campo, basta clicar no botão informar uso para que as informações sejam salvas no sistema, porém, antes de salvar tais informações, uma janela, contendo os dados a serem salvos é exibida para que o

usuário cheque se as informações inseridas batem com o que ele realmente deseja informar, a fim de reduzir erros de submissão de informações.

**Figura 21 - Informar Uso**


**Fonte: Autoria própria**

### 3.6.5 MENU DE ADMINISTRADOR

Para a gerência do laboratório, foi criado um menu administrador, possuindo acesso apenas os usuários do tipo administrador, contendo as funcionalidades pedidas pelo cliente, como gerar relatório, criar, atualizar, excluir e consultar os equipamentos, reagentes e meios de cultivo.

Nesse menu está presente o controle de liberação de novos usuarios pelo administrador.

Menu administrador, onde mais abaixo serão explicadas suas funcionalidades:

**Figura 22 - Menu Administrador**


**Fonte: Autoria própria**

### *3.6.5.1* NOVOS ITENS

Para a criação de novos reagentes, meios de cultivo, vidrarias ou equipamentos, foi criada a tela Novo Item, onde o administrador seleciona qual tipo de item deseja dar a entrada e um layout de cadastro é aberto especificamente para cada escolha selecionada.

Mesmo os itens possuindo atributos diferentes, a cada seleção feita por um usuário será atualizado o formulário de criação de novo item, se adaptando ao tipo de item selecionado.

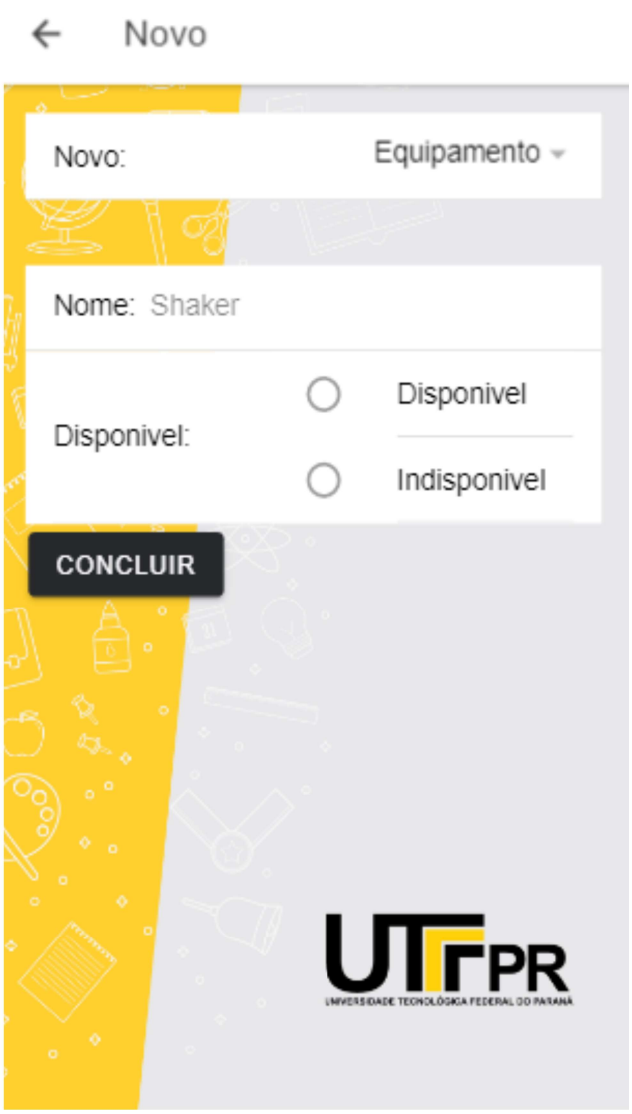
**Figura 23 - Novo reagente ou meio de cultivo**


**Fonte: Autoria própria**

Como para reagentes e meios de cultivo os atributos pedidos pelo cliente para serem salvos no banco de dados são os mesmos, os campos solicitados para o administrador no momento da criação de um novo reagente ou meio de cultivo também são os mesmos, a nomenclatura do reagente ou meio de cultivo a ser cadastrado, acompanhado de sua numeração para identificação e a quantidade que será cadastrada inicialmente.

Caso o administrador queira realizar o cadastro de um novo equipamento para o laboratório será direcionado a tela da Figura 24.

**Figura 24 - Novo equipamento**




**Fonte: Autoria própria**

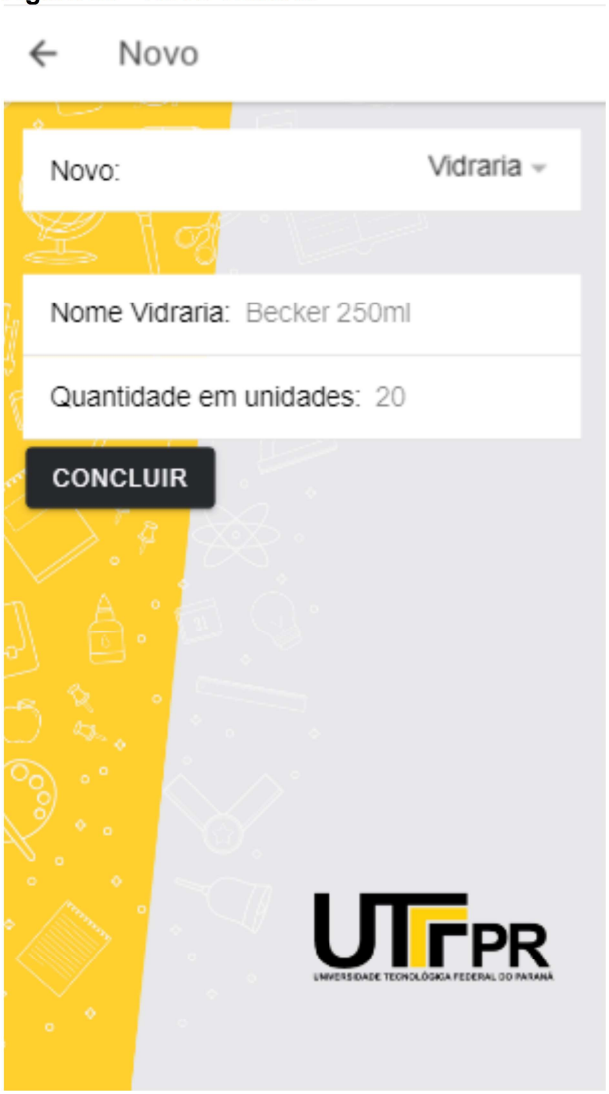
Para o equipamento foi solicitado apenas o nome do próprio e foi adicionada uma opção para dizer se o equipamento está disponível para uso ou não, para evitar reservas com equipamentos danificados.

Se a opção selecionada pelo administrador para criação for uma nova vidraria, foi solicitado pelo cliente apenas o nome e a quantidade da nova vidraria.

Assim pode-se manter um maior controle do que vai entrar no laboratório para uso nas atividades, podendo assim manter um controle de estoque das

vidrarias e evitando a nova ocorrência de faltar determinada vidraria para alguma aula ou experimento.

**Figura 25 - Nova Vidraria**




**Fonte: Autoria própria**

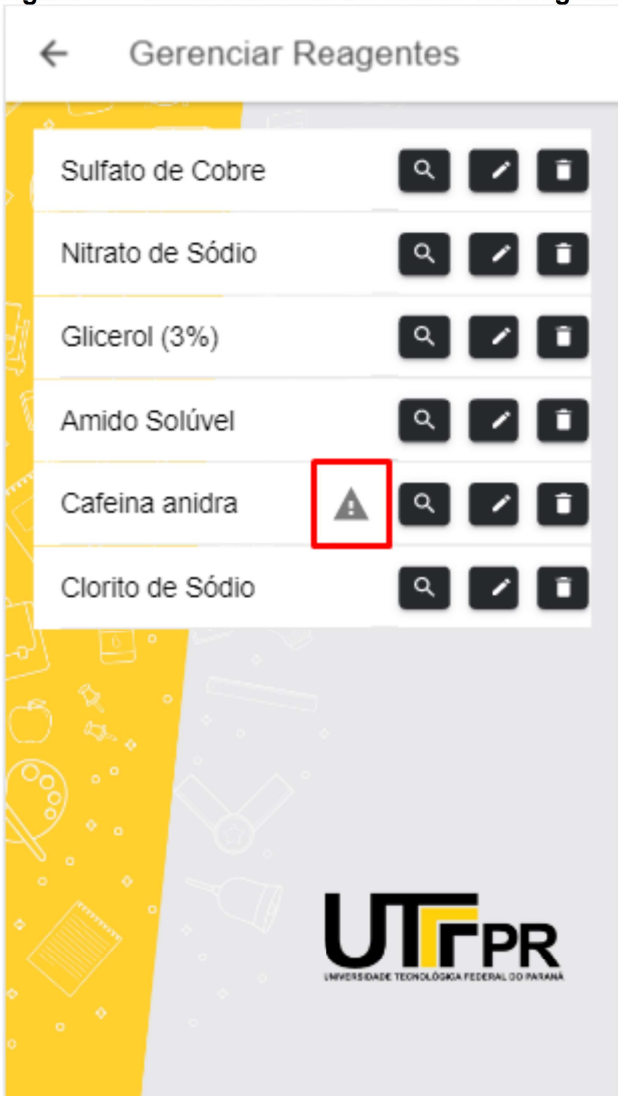
Se a opção for nova vidraria, foi solicitado pelo cliente apenas o nome e a quantidade da nova vidraria que vai entrar no laboratório para uso nas atividades.

### 3.6.6 OPÇÕES DE GERENCIAMENTO

Como mostrado na Figura 24, o usuário tipo professor tem as opções de gerência de equipamentos, reagentes, meio de cultivo ou vidraria, onde em cada um deles é possível fazer a consulta dos itens já cadastrados em cada tópico, editar os

itens caso precise, por exemplo, adicionar uma quantidade nova de um reagente ou meio de cultivo e a exclusão se necessário.

**Figura 26 - Gerenciar meio de cultivo ou reagente**




**Fonte: Autoria própria**

É listado todos os reagentes ou meio de cultivo para o administrador. Um aviso, demarcado em vermelho na figura, é mostrado caso o reagente esteja com sua quantidade menor do que a quantidade estabelecida pelo administrador na hora do cadastro e 3 botões.

O botão com uma lupa o administrador pode verificar os dados do item, o botão com ícone lixeira é utilizado para excluir um item, aparecendo um aviso se o administrador tem certeza da ação que está tomando.

O botão com lápis leva o administrador para uma tela onde ele possa fazer edição dos dados do reagente ou dar entrada de uma quantidade do mesmo item:

**Figura 27 - Editar meio de cultivo ou reagente**

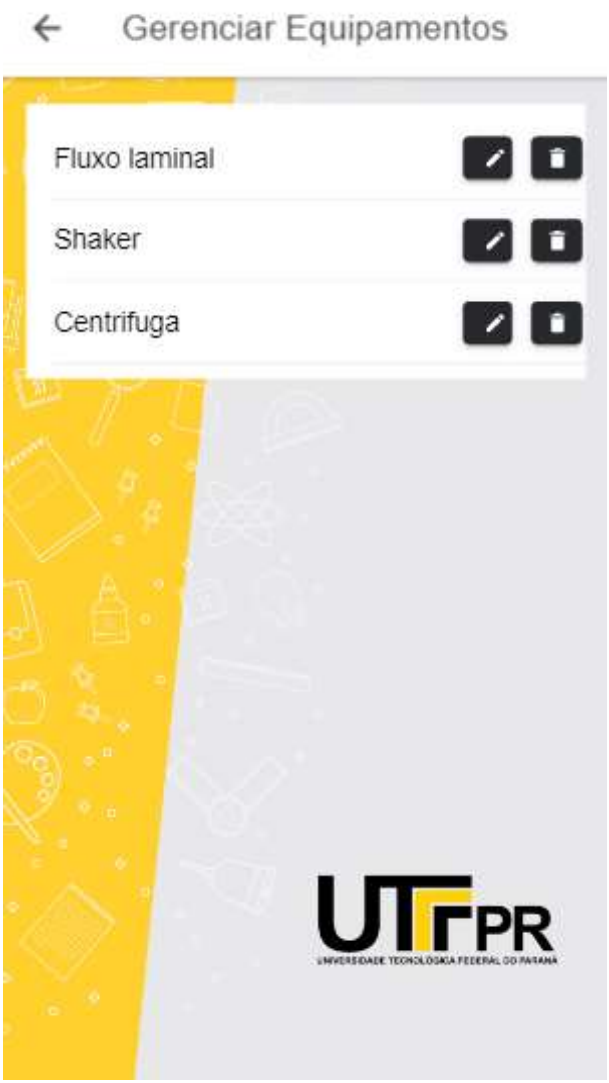

**Fonte: Autoria própria**

Na Figura 27 é detalhada a edição do Sulfato de Cobre, o qual possui sua numeração 8C, com quantidade para uso de 5.00 ml. Foi definida pelo administrador na hora do cadastro a quantidade mínima de 6.20 ml para o Sulfato de Cobre e o último campo é para dar a entrada de nova quantidade do reagente.

O menu de gerência de equipamentos ficou semelhante ao de reagentes e meios de cultivo.Porém como o único atributo do equipamento é o nome, não possuindo quantidade, quantidade mínima ou numeração.

Então não foi criada uma lupa para que pudesse ser visualizado os dados dos equipamentos, já que a única informação a ser exibida para os usuários administradores é exatamente o nome do equipamento.

**Figura 28 - Gerenciar equipamentos**




**Fonte: Autoria própria**

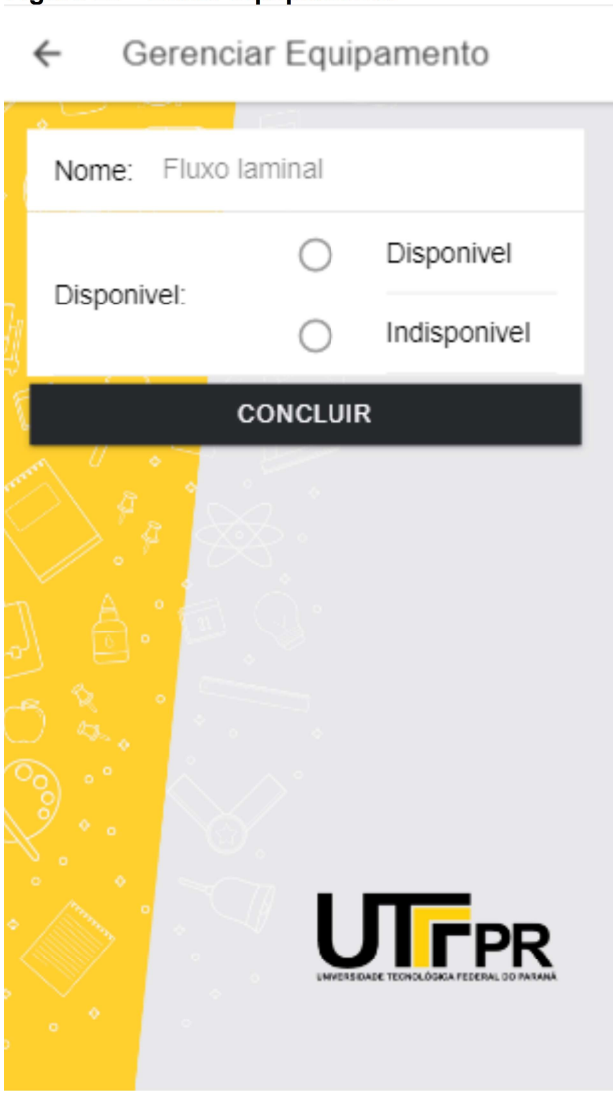
Mostrado o nome do equipamento para o administrador, como exemplo a centrífuga e dois botões de opções de editar e excluir.

Dos botões da tela, o botão com ícone de lápis o usuário administrador pode editar o equipamento, isto é, definir se ele está disponível ou não, podendo passar por manutenção ou simplesmente quebrado e também é possível editar o nome caso necessário.

O botão com ícone de lixeira é onde o usuário administrador pode excluir o equipamento caso haja necessidade da exclusão do mesmo, tratando que a

universidade deixou de contar com aquele equipamento no laboratório e manter seus registros não são mais necessarios.

**Figura 29 - Editar equipamento**



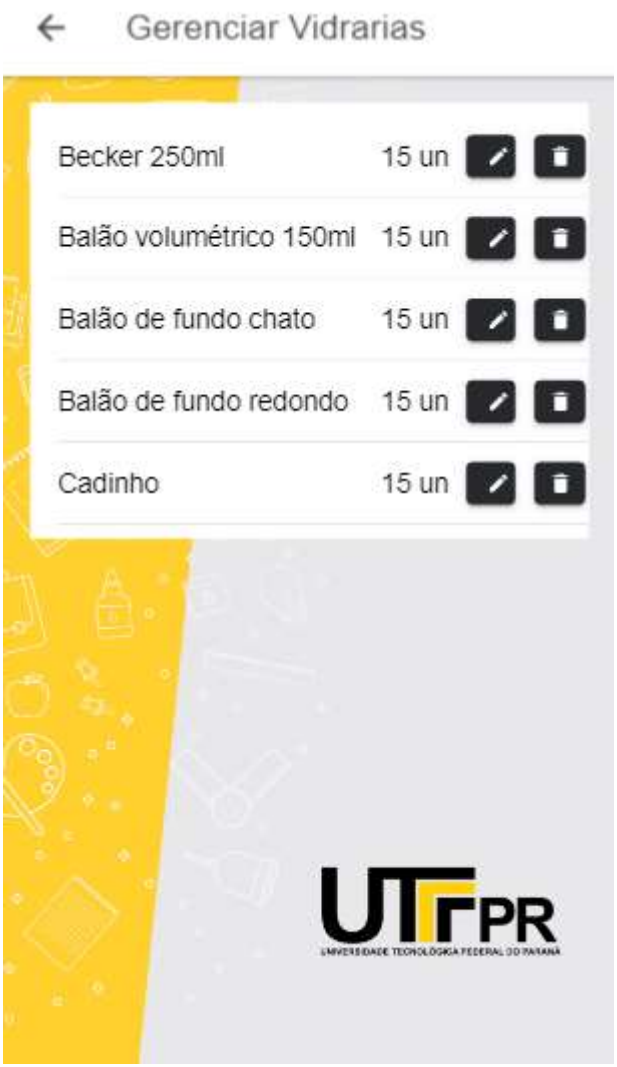

**Fonte: Autoria própria**

Uma tela simples, se tratando de um item que só possui um atributo que atinge aos usuários, no caso o atributo nome, e o atributo disponível com valor oculto aos usuários do tipo alunos, porém caso algum equipamento seja definido como indisponível, o mesmo é removido da lista de equipamentos para reservas de usuários.

Para a gerência das vidrarias, uma tela relativamente simples, muito semelhante à do equipamento, porém a tela de vidrarias conta com um campo onde, para cada item, é mostrado a quantidade em unidades da vidraria presentes no lab.

Se ocorrer algum dano a vidraria, é descontando nessa quantidade. Caso uma vidraria já cadastrada tenha atualização de estoque, na opção de editar é possível alterar essa quantidade, somando a nova a atual, semelhante ao adicionar reagente ou meio de cultivo.

**Figura 30 - Editar Vidraria**




**Fonte: Autoria própria**

### *3.6.6.1* RELATÓRIO

O relatório consiste em registrar os danos notificados pelos usuários e tudo que entra e sai do laboratório (como reagentes, meios de cultivo e vidrarias) e exibir esses registros para serem analisados posteriormente.

Essa funcionalidade tem o intuito de obter um controle maior, e ciência, de como está sendo utilizado os recursos por um determinado período de tempo.

Acompanhando os relatórios fica facilitado para o administrador acompanhar quando é necessário a reposição de um reagente, meio de cultivo ou vidraria.

**Figura 31 - Tela inicial do relatório**

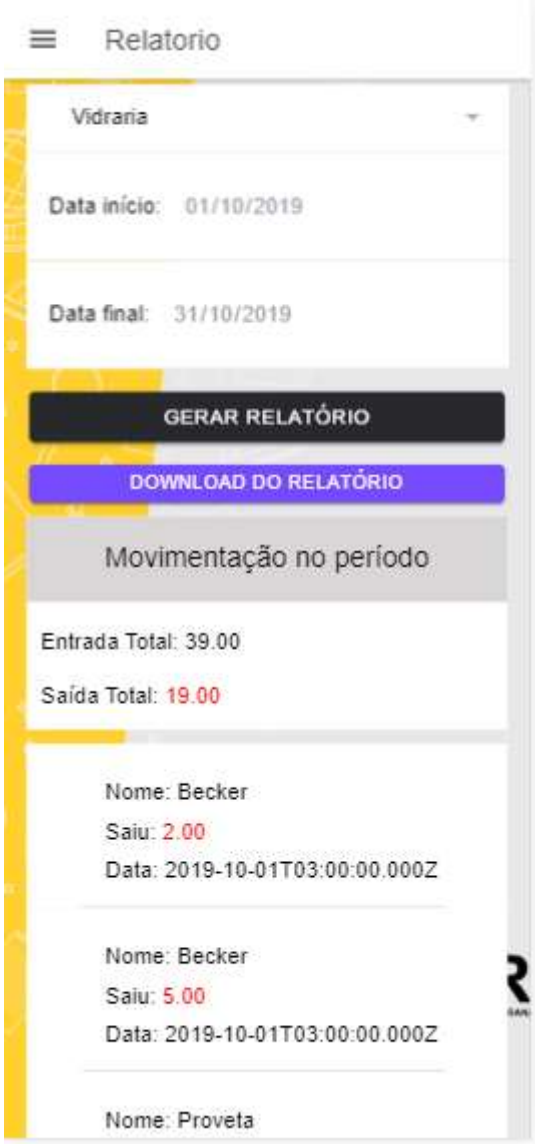

**Fonte: Autoria própria**

Na Figura 31 é mostrada a tela inicial da sessão relatório, onde o primeiro campo contem um ***select*** (seleção), contendo as vidrarias, reagentes e meios de cultivo, o qual o usuário administrador seleciona para gerar o relatório.

Abaixo do ***select*** é pedido para o usuário administrador a data inicial do relatório e a data final, respectivamente, do período em que deseja-se obter as informações sobre o item selecionado.

Após o preenchimento dos campos solicitados pela aplicação e o acionamento do botão “Gerar relatório”, as informações são mostradas na tela para o administrador:

**Figura 32 - Tela de pesquisa do relatório**




**Fonte: Autoria própria**

Pode-se notar que logo abaixo do botão é apresentado o quadro “movimentação no período”, que informa o total de entradas e saídas do item, no período selecionado. Logo após esse quadro, é exibida uma lista com todos os registros realizados neste período. Se a opção selecionada for vidrarias, é possível clicar em um item e visualizar a descrição feita pelo usuário que relatou o dano. Devido ao fato de essa funcionalidade gerar muitos dados (registros), o que pode

sobrecarregar visualmente a tela, um botão para gerar PDF é disponibilizado, possibilitando também à análise em outros dispositivos, como notebooks.

# 4 CONCLUSÃO

O trabalho foi realizado com intuito de apoiar na gestão do laboratório de fermentação da Universidade Tecnológica Federal do Paraná campus Ponta Grossa, o qual apresenta dificuldades em manter seus registros íntegros e organizados. No laboratório são feitos variados experimentos, que podem ser manejados por alunos em aula e por outros usuários do laboratório, sendo acessível também para usuários externos ao campus. Como levantado pelo professor do laboratório, pela não clareza dos dados nas reservas, já ocorreram percas de aula programada, por não possuir a quantidade indicada na folha de registro. Essa e outras situações que podem vir a ocorrer pelo controle inadequado de registros têm sido prejudiciais ao aprendizado, as atividades e experimentos realizados.

Para solucionar esse problema, o professor requisitou um aplicativo mobile, com sistema de login para usuários, os quais podem reservas os equipamentos para uso, notificar uso de reagentes e meios de cultivo e informar possíveis danos a vidrarias. Um aplicativo mobile facilita o acesso aos recursos do laboratório e um maior controle de organização e gerência para o professor/administrador. Como se trata de um aplicativo de metodologia híbrida, pode-se atingir a todos os usuários que possuírem um smartphone, atendendo a usuários do iPhone e Android.

A modelagem e desenvolvimento de um aplicativo móvel para gerenciamento do laboratório de fermentação da UTFPR-PG foram realizadas com sucesso. Para esse projeto foram escolhidas ferramentas que melhor se adaptassem ao escopo do projeto.

Os objetivos levantados para o aplicativo foram cumpridos, o sistema é capaz de cadastrar, atualizar, editar e excluir reagentes, meios de cultivo, vidrarias, equipamentos e usuários. O sistema também possui a funcionalidade de reservar equipamentos e reportar eventuais danos que podem vir a acontecer com as vidrarias do laboratório.

Uma sessão de administração foi criada para a gerência, permitindo a criação de relatórios de entrada e saída de reagentes, meios de cultivo e vidrarias. Foi solicitado um controle de cadastro, sendo o mesmo possível apenas com autorização do administrador.

Para manter um bom controle de versões durante o desenvolvimento da aplicação, uma ferramenta que auxilia os desenvolvedores e que nós utilizamos foi o GitHub, o qual classificamos como indispensável quando se trabalha em equipe.

Enquanto no desenvolvimento do aplicativo, concluímos que separar o sistema ou aplicação a ser desenvolvida em módulos pode ser interessante na hora de juntar o ***front-end*** com o ***back-end*** e manter a organização.

# 5 TRABALHOS FUTUROS

Para trabalhos futuros deixamos os seguintes itens:

-Um histórico de movimentações, reservas e danos de cada usuário;

-Um menu “Grupo de Estudos”, onde os usuários com as mesmas aulas possam se comunicar através do aplicativo, visto que hoje é comum os estudantes criarem grupos de WhatsApp para estudo;

-Melhorar o estoque, criando um controle de lote de reagentes, meios e vidrarias;

-Implantar o aplicativo no laboratório.

# REFERÊNCIAS

ADONIS. **Adonis docs**. Disponível em: <https://adonisjs.com/docs/4.1>. Acesso em: 09. Out. 2019.

ADONIS. **Lucid models**. Disponível em: <https://adonisjs.com/docs/4.0/lucid>. Acesso em: 09. Out. 2019.

ANGULAR. **Architecture overview**. Disponível em: <https://angular.io/guide/architecture>. Acesso em: 08 jun. 2019.

CORDOVA. **Overview**. Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>. Acesso em: 05. Jun. 2019.

CRONAPP. **Compreenda as diferenças entre o desenvolvimento nativo e híbrido.** Disponível em: <https://www.cronapp.io/pt-br/compreenda-as-diferencas-entre-o-desenvolvimento-nativo-e-hibrido/>. Acesso em: 17 mai. 2019.

DEVMEDIA. **ORM – obejct relational mapping**. Disponível em: <https://www.devmedia.com.br/orm-object-relational-mapping-revista-easy-net-magazine-28/27158>. Acesso em: 09. Out. 2019.

I. MALAVOLTA; S. RUBERTO. **End users' perception of hybrid mobile apps in the google play store.** Disponível em: <http://www.ivanomalavolta.com/files/papers/MS_2015.pdf>. Acesso em: 17 mai. 2019.

IONIC. **Ionic framework docs**. Disponível em: <https://ionicframework.com/docs>. Acesso em: 05. Jun. 2019.

J. MISZURA, T; G. DIVINO, D. **Desenvolvimento em Smartphones - Aplicativos Nativos e Web**. Disponível em: <http://www.cpgls.pucgoias.edu.br/7mostra/Artigos/agrarias.pdf>.Acesso em: 11 mai. 2019.

J. RODRIGUES. **Javascript tutorial**. Disponível em: <https://www.devmedia.com.br/javascript-tutorial/37257>. Acesso em: 08 mai. 2019.

K. SCHENATTO. **Normalização de Dados**. Disponível em: <http://paginapessoal.utfpr.edu.br/kschenatto/fundamentos-de-banco-de-dados/aulas-2-2015/Normalizacao%20de%20dados.pdf/at_download/file>. Acesso em: 25 Out. 2019.

KELLY C. VENTEU. D**esenvolvimento móvel híbrido.** Disponível em: <https://revista.fatectq.edu.br/index.php/interfacetecnologica/article/download/337/215/>. Acesso em: 17 mai. 2019.

M. GUEDES. **App nativo x app híbrido: existe o melhor?**. Disponível em: <https://www.cronapp.io/pt-br/compreenda-as-diferencas-entre-o-desenvolvimento-nativo-e-hibrido/>. Acesso em: 18 mai. 2019.

M. HUYNH. **Hybrid app approach: could it mark the end of native app domination?.** Disponível em: <http://iisit.org/Vol14/IISITv14p049-065Huynh3472.pdf>. Acesso em: 13 mai. 2019.

MADUREIRA, Daniel. **Aplicativo nativo, web App ou aplicativo híbrido**?. Disponível em: <https://usemobile.com.br/aplicativo-nativo-web-hibrido/>. Acesso em: 12 mai, 2019.

MDN. **CSS básico**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Aprender/Getting_started_with_the_web/CSS_basico>. Acesso em: 04. Jun. 2019.

MDN. **Learning html: Guides and tutorials.** Disponível em: <https://developer.mozilla.org/en-US/docs/Learn/HTML>. Acesso em: 08. Jun. 2019.

NANDO VIEIRA. **Entendendo um pouco mais sobre o protocolo http**. Disponível em: <https://nandovieira.com.br/entendendo-um-pouco-mais-sobre-o-protocolo-http>. Acesso em: 09. Out. 2019.

NODEJS. **Node about**. Disponível em: <https://nodejs.org/en/about/>. Acesso em: 07. Out. 2019.

PHPFLOW. **Types of web services soap, xml-rpc and restfull**. Disponível em: <https://www.phpflow.com/php/web-service-types-soapxml-rpcrestful/>. Acesso em: 09. Out. 2019.

SOMMERVILLE, Ian. Engenharia de Software. 9. Ed. São Paulo. 2011.

TYPESCRIPT. **Typescript website**. Disponível em: <https://www.typescriptlang.org>. Acesso em: 25. Out. 2019.

W. LELER. **What's revolutionary about flutter.** Disponível em: <https://hackernoon.com/whats-revolutionary-about-flutter-946915b09514>. Acesso em: 11 jun. 2019.