CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE FRANCA

“Dr. THOMAZ NOVELINO”

**TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

iGOR PRATES GONÇALVES

VINICIUS TEIXEIRA DO CARMO

achados, perdidos & roubados

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof.Drª Jaqueline Brigladori Pugliesi

FRANCA/SP

2020

**Resumo**

O Software de Achados, perdidos e roubados irá atender todo o tipo de usuário que necessita encontrar um item perdido ou entregar um item achado à outra pessoa. Com base na necessidade de resolver situações inesperadas que ocorrem no nosso dia-a-dia foi desenvolvido um software para aproximar estes usuários. Ele conta com listas e datas que ajudam o usuário a ter uma fácil compreensão de como usá-lo. Sua interface é agradável e harmoniosa, e não requer conhecimentos avançados para sua utilização, o usuário deverá informar os dados que serão solicitados e posteriormente analisados, ou irá procurar em uma das listas para que assim possam conseguir reaver itens perdidos, de forma rápida e fácil, economizando tempo de procura, reduzindo espaço utilizados dos locais que possuem achados e perdidos, pois com o software as pessoas irão devolver os itens encontrados diretamente aos respectivos donos, evitando assim acumulo de itens nesses locais. O projeto tenta também incentivar as pessoas a devolverem itens achados através da facilidade e rapidez com que o software proporciona, pois, as pessoas só irão cadastrar os itens e depois é só esperar receber notificação de alguém falando que achou ou perdeu aquele respectivo item.

**Palavras-chave:** Achados, Perdidos, Roubados.

***Abstract***

*The Lost, Stolen and Found Software will serve all types of users who need to find a lost item or deliver a found item to someone else. Based on the need to solve unexpected situations that occur in our daily lives, software was developed to bring these users closer. It has lists and dates that help the user to have an easy understanding of how to use it. Its interface is pleasant and harmonious, and does not require advanced knowledge for its use, the user must inform the data that will be requested and subsequently analyzed, or will search in one of the lists so that they can get lost items, quickly and easily , saving search time, reducing space used in places that have found and lost, because with the software people will return the items found directly to their respective owners, thus avoiding accumulation of items in those places. The project also tries to encourage people to return items found through the ease and speed with which the software provides, because people will only register the items and then just wait to receive notification from someone saying that they found or lost that respective item.*

***Keywords:*** *Found, Lost, Stolen.*

1 Introdução

As ações que executamos ao longo do nosso dia a dia passam, inevitavelmente, de forma direta ou indireta, pela utilização de aplicações de software. Neste contexto que resolvemos criar uma ferramenta de software que auxiliasse em um problema que as cidades do mundo inteiro têm: itens que são achados, perdidos ou roubados. Muitas vezes pessoas que acham um item não sabem, ou não tem as instruções necessárias para fazer a devolução do mesmo, o mesmo acontece com pessoas que perdem algo, e no caso das pessoas que tiveram seus itens furtados, o software tentará auxiliar o seu encontro através de pessoas que possam ter visto os criminosos ou ter visto o próprio objeto.

**2 Viabilidade do projeto**

O projeto funcionaria através de parcerias com locais que costumam ter um fluxo de pessoas elevados, pois são esses lugares que costumam ocorrer o maior número casos de itens achados, perdidos ou roubados. A atividade chave do projeto é conseguir fazer com que as pessoas possam recuperar itens perdidos de maneira simples e rápida, economizando tempo e espaços. Os recursos utilizados no projeto foram os mínimos possíveis, como computadores próprio dos autores, celulares e principalmente tempo.

A proposta do projeto é tecnologia da Informação como ferramenta para devolução e cadastramento de itens achados, perdidos e roubados. E o software sera disponibilizado na rede totalmente gratuito para que qualquer usuario possa ter acesso através de smartphones ou computadores, pois o publico alvo são as pessoas que acham, perdem ou tem o seus items roubados,sendo qualquer pessoa a cima de dezoito anos.

**3 Levantamento de Requisitos**

3.1 Elicitações especificação dos Requisitos.

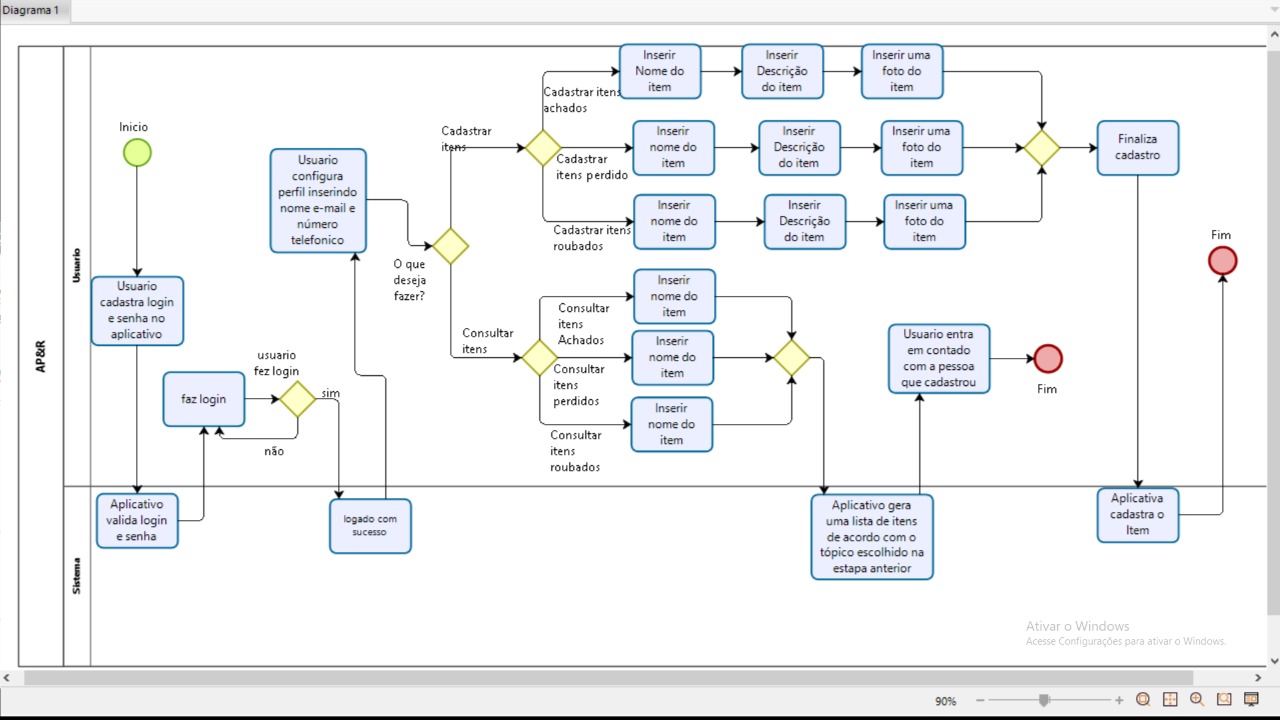
O levantamento de requisitos foi feito através de algumas visitas feitas em locais aonde costumam ter um fluxo grande de pessoas, como terminal de ônibus, rodoviária, Fatec e algumas escolas, sendo observado como esses locais lidam com as pessoas que tem os objetos achados perdidos e roubados, com os objetos e quais objetos aparecem nesses locais.

Durante essas visitas, conversamos com pessoas que trabalham nos locais para que pudéssemos melhor compreender sobre a frequência que as pessoas perdem objetos, sobre a frequência que as pessoas acham os itens que elas haviam perdido.

E assim foram observados pontos positivos e negativos ajudando a organizar, estruturar e por fim desenvolver o software da melhor maneira possível, para que ele pudesse cumprir o objetivo para o qual ele foi desenvolvido de maneira mais simples possível.

3.2 A Figura 1 apresenta O Business Process Model and Notation (BPMN) que é uma notação da metodologia de gerenciamento de processos de negócio que se trata de uma série de ícones padrões para o desenho de processos, o que facilita o entendimento do usuário.

**Figura 1** - Business Process Model and Notation (BPMN).



**Fonte:** Autores.

3.3 O Quadro 1 representa o que o software faz, em termos de tarefas e serviços. Uma função é descrita como um conjunto de entradas, seu comportamento e as saídas.

**Quadro 1 –** Requisitos Funcionais do sistema.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RF 001 – Achados** | Categoria: ( ) Oculto  (X) Evidente | | Prioridade: ( ) Altíssima  (X) Alta  ( ) Média  ( ) Baixa | | |
| **Descrição**: Na tela de Achados irá conter a lista com os itens que foram cadastrados e que se enquadram nesse tipo, a lista deverá ter alguns campos obrigatórios, como localização e data de onde foi achado, o tipo do item e o nome do item. Também irá conter o botão “É meu” que irá dar o “match” e levar para o chat. | | | | | |
| **RF 002 – Perdidos** | Categoria: ( ) Oculto  (X) Evidente | | Prioridade: ( ) Altíssima  (X) Alta  ( ) Média  ( ) Baixa | | |
| **Descrição**: Na tela de Perdidos irá conter a lista com os itens que foram cadastrados e que se enquadram nesse tipo, a lista deverá ter alguns campos obrigatórios, como localização e data de onde foi achado, o tipo do item, o nome do item e a descrição.. Também irá conter o botão “Eu perdi” que irá dar o “match” e levar para o chat. | | | | | |
| **RF 003 – Roubados** | Categoria: ( ) Oculto  (X) Evidente | | Prioridade: ( ) Altíssima  (X) Alta  ( ) Média  ( ) Baixa | | |
| **Descrição**: Na tela de Roubados irá conter a lista com os itens que foram cadastrados e que se enquadram nesse tipo, a lista deverá ter alguns campos obrigatórios, como localização e data de onde foi achado, o tipo do item, o nome do item e a descrição. Também irá conter o botão “Ajuda” que irá dar o “match” e levar para o chat. | | | | | |
| **RF 004 – Cadastrar Achado** | Categoria: ( ) Oculto  (X) Evidente | | Prioridade: (X) Altíssima  ( ) Alta  ( ) Média  ( ) Baixa | | |
| **Descrição**: Na tela de cadastrar um item achado irá conter um formulário com os campos que serão precisos para cadastrar o item, deverá conter alguns campos obrigatórios, como localização e data de onde foi achado, o tipo do item e o nome do item, os campos descrição e imagens não serão obrigatórios. Também contará com os botões de enviar e cancelar, que redireciona para lista referente ao tipo do item que foi cadastrado e o outro voltará para página principal, respectivamente. | | | | | |
| **RF 005 – Cadastrar Perdido** | Categoria: ( ) Oculto  (X) Evidente | | Prioridade: (X) Altíssima  ( ) Alta  ( ) Média  ( ) Baixa | | |
| **Descrição**: Na tela de cadastrar um item perdido irá conter um formulário com os campos que serão precisos para cadastrar o item, deverá conter alguns campos obrigatórios, como localização e data de onde foi achado, o tipo do item ,o nome do item e a descrição, o campo imagens não será obrigatório. Também contará com os botões de enviar e cancelar, que redireciona para lista referente ao tipo do item que foi cadastrado e o outro voltará para página principal, respectivamente. | | | | | |
| **RF 006 – Cadastrar Roubado** | | Categoria: ( ) Oculto  (X) Evidente | | Prioridade: (X) Altíssima  ( ) Alta  ( ) Média  ( ) Baixa | |
| **Descrição**: Na tela de cadastrar um item roubado irá conter um formulário com os campos que serão precisos para cadastrar o item, deverá conter alguns campos obrigatórios, como localização e data de onde foi achado, o tipo do item ,o nome do item e a descrição, o campo imagens não será obrigatório. Também contará com os botões de enviar e cancelar, que redireciona para lista referente ao tipo do item que foi cadastrado e o outro voltará para página principal, respectivamente. | | | | | |
| **RF 007 – Tipo do Item** | | Categoria: ( ) Oculto  (X) Evidente | | Prioridade: ( ) Altíssima  (X) Alta  ( ) Média  ( ) Baixa | |
| **Descrição**: O tipo do item vai ser um input que o usuário poderá digitar o tipo do item, mas ele irá contar com um auto completar que irá facilitar caso ele não saiba o que colocar nesse campo que é obrigatório. | | | | | |
| **RF 008 – Formulário** | | Categoria: ( ) Oculto  (X) Evidente | | Prioridade: ( ) Altíssima  (X) Alta  ( ) Média  ( ) Baixa | |
| **Descrição**: Em cada página de cadastro de item deverá conter um formulário com os campos específicos de cada tipo, que o usuário deverá preencher os campos obrigatórios para conseguir concluir e ir para a listagem, senão irá dar uma mensagem de erro. | | | | | |
| **RF 009 – Enviar** | | Categoria: ( ) Oculto  (X) Evidente | | | Prioridade: ( ) Altíssima  (X) Alta  ( ) Média  ( ) Baixa |
| **Descrição**: Botão de enviar que ficará nas telas de cadastro de itens que será responsável por enviar as informações do formulário e se estiver tudo certo irá para a página de listagem de itens. | | | | | |
| **RF 0010 – Termos de responsabilidade** | | Categoria: ( ) Oculto  (X) Evidente | | | Prioridade: ( ) Altíssima  (X) Alta  ( ) Média  ( ) Baixa |
| **Descrição**: Link na página de cadastro de usuário que irá abrir uma página mostrando os termos de uso do software e a política de privacidade. | | | | | |

3.4 O quadro 2 apresenta os requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas. Estes requisitos dizem respeito a como as funcionalidades serão entregues ao usuário do software.

**Quadro 2** – Requisitos Não Funcionais do sistema.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requisitos não funcionais** | | | | |
| Nome | Restrição | Categoria | Obrigatoriedade | Permanência |
| RNF001 - Cadastro | O sistema deverá exigir um cadastro de usuário | (X) Oculto  ( ) Evidente | ( ) Desejável  (X) Obrigatório | (X) Permanente  ( ) Transitório |
| RNF002 - Login | O sistema deverá exigir o login do usuário já cadastrado | (X) Oculto  ( ) Evidente | ( ) Desejável  (X) Obrigatório | (X) Permanente  ( ) Transitório |
| RNF003 – Verificação de e-mail | O sistema deverá exigir a verificação do e-mail para efetuar o cadastro | (X) Oculto  ( ) Evidente | ( ) Desejável  (X) Obrigatório | (X) Permanente  ( ) Transitório |
| RNF004 – Termo de Responsabilidade | O usuário deverá aceitar os Termos de Responsabilidade | (X) Oculto  ( ) Evidente | (X ) Desejável  ( ) Obrigatório | (X) Permanente  ( ) Transitório |

3.5 O quadro 3 define a forma de fazer o negócio, refletindo a política interna, o processo definido e/ou as regras básicas de conduta. Ou seja, é um conjunto de instruções que os usuários já seguem e que o sistema a ser desenvolvido deve contemplar.

**Quadro 3** – Regras de Negócio do sistema.

|  |  |
| --- | --- |
| RN001 | O sistema não se responsabilizará por qualquer item que for cadastrado. |
| RN002 | As listas de Achados, perdidos e roubados serão ordenadas por nome e pela data da postagem. |
| RN003 | O usuário deve utilizar um e-mail valido. |

3.6 Casos de Uso.

Índice de casos de uso, diagrama de casos de uso e especificações de casos de uso.

**3.6.1 Índices de Casos de Uso**

UC 001 – Cadastrar-se no sistema.

UC 002 – Definir qual lista deseja ter acesso (lista de achados, perdidos ou roubados).

UC 003 – Visualizar lista de Achados.

UC 004 – Visualizar lista de Perdidos.

UC 005 - Visualizar lista de Roubados.

UC 006 - Cadastrar item na lista de Achados, perdidos ou roubados.

**3.6.2 Diagrama de casos de uso**

Na Figura 2 está representando uma unidade funcional coerente provida pelo sistema, subsistema, ou classe manifestada por sequências de mensagens intercambiáveis entre os sistemas e um ou mais atores.

**Figura 2 –** Casos de uso.

Mapa com linhas pretas em fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

**3.6.3 Especificações dos casos de uso.**

No quadro 4 explica como deve funcionar o cadastro de usuários, seus cenários principais e cenários alternativos.

**Quadro 4** – Use Case Cadastrar Usuários

|  |  |
| --- | --- |
| **Caso de Uso – Cadastrar Usuário** | |
| **ID** | UC 001 – Cadastrar Usuário |
| **Descrição** | Este caso de uso tem por objetivo realizar o cadastro do usuário. |
| **Ator Primário** | Usuário |
| **Pré-condição** | Não possui |
| **Cenário Principal** | 1. O use case inicia quando o usuário seleciona a opção de Cadastro  2. O sistema mostra um formulário  3. O usuário digita seus dados  4. O usuário é obrigado a validar o e-mail |
| **Pós-condição** | Ter validado o e-mail |
| **Cenário Alternativo** | \*a – Em qualquer momento o usuário pode sair do sistema  3a – Dados inválidos  3a.1 O sistema mostra mensagem informando que os dados são inválidos |
| **Inclusão** |  |
| **Extensão** |  |

No quadro 5 explica como o usuário irá visualizar as listas.

**Quadro 5** – Use Case definir lista

|  |  |
| --- | --- |
| **Caso de Uso – Definir lista** | |
| **ID** | UC 002 – Definir qual lista deseja ter acesso (lista de achados, perdidos ou roubados) |
| **Descrição** | Este caso de uso tem por objetivo definir qual lista o usuário quer visualizar. |
| **Ator Primário** | Usuário |
| **Pré-condição** | Estar logado no sistema |
| **Cenário Principal** | 1. O use case inicia quando o usuário faz login  2. O sistema mostra duas 3 opções  3. O usuário seleciona a que deseja |
| **Pós-condição** | Não possui |
| **Cenário Alternativo** | \*a – Em qualquer momento o usuário pode sair do sistema |
| **Inclusão** |  |
| **Extensão** |  |

No quadro 6 explica como deve funcionar a lista de itens achados, onde é mostrado o cenário principal, e possíveis cenários alternativos.

**Quadro 6** – Use Case Visualizar Achados

|  |  |
| --- | --- |
| **Caso de Uso – Visualizar Achados** | |
| **ID** | UC 003 - Visualizar Achados |
| **Descrição** | Este caso de uso tem por objetivo visualizar a lista de Achados |
| **Ator Primário** | Usuário |
| **Pré-condição** | Efetuar login |
| **Cenário Principal** | 1. O use case inicia quando o usuário já inseriu efetuou o login  2. O sistema carrega a lista online  3. O usuário visualiza a lista  4. O usuário encontra o que procura |
| **Pós-condição** | Não possui |
| **Cenário Alternativo** | \*a – Em qualquer momento o usuário pode sair do sistema  3a – Usuário sem internet  3a.1 – O sistema carregará a lista que foi atualizada da última vez que a internet foi acessada  4b – Usuário não encontrou  4b.1 – O usuário fará o cadastro do item então |
| **Inclusão** |  |
| **Extensão** |  |

No quadro 7 explica como deve funcionar a lista de itens perdidos, onde é mostrado o cenário principal, e seus possíveis cenários alternativos.

**Quadro 7** – Use Case Visualizar Perdidos

|  |  |
| --- | --- |
| **Caso de Uso – Visualizar Perdidos** | |
| **ID** | UC 004 - Visualizar Perdidos |
| **Descrição** | Este caso de uso tem por objetivo visualizar a lista dos Perdidos |
| **Ator Primário** | Usuário |
| **Pré-condição** | Efetuar login |
| **Cenário Principal** | 1. O use case inicia quando o usuário já inseriu efetuou o login  2. O sistema carrega a lista online  3. O usuário visualiza a lista  4. O usuário encontra o que procura |
| **Pós-condição** | Não possui |
| **Cenário Alternativo** | \*a – Em qualquer momento o usuário pode sair do sistema  3a – Usuário sem internet  3a.1 – O sistema carregará a lista que foi atualizada da última vez que a internet foi acessada  4b – Usuário não encontrou  4b.1 – O usuário fará o cadastro do item então |
| **Inclusão** |  |
| **Extensão** |  |

No quadro 8 explica como deve funcionar a lista de itens perdidos, onde é mostrado o cenário principal, e seus possíveis cenários alternativos.

**Quadro 8** – Use Case Visualizar Roubados

|  |  |
| --- | --- |
| **Caso de Uso – Visualizar Roubados** | |
| **ID** | UC 005 – Visualizar Roubados |
| **Descrição** | Este caso de uso tem por objetivo visualizar a lista dos Roubados |
| **Ator Primário** | Usuário |
| **Pré-condição** | Efetuar login |
| **Cenário Principal** | 1. O use case inicia quando o usuário já inseriu efetuou o login  2. O sistema carrega a lista online  3. O usuário visualiza a lista  4. O usuário encontra o que procura |
| **Pós-condição** | Não possui |
| **Cenário Alternativo** | \*a – Em qualquer momento o usuário pode sair do sistema  3a – Usuário sem internet  3a.1 – O sistema carregará a lista que foi atualizada da última vez que a internet foi acessada  4b – Usuário não encontrou  4b.1 – O usuário fará o cadastro do item então |
| **Inclusão** |  |
| **Extensão** |  |

No quadro 9 explica como deve funcionar o cadastramento de itens achados perdidos ou roubados.

**Quadro 9** – Use Case cadastro de item

|  |  |
| --- | --- |
| **Caso de Uso – Cadastro de item** | |
| **ID** | UC 006 – cadastrar item |
| **Descrição** | Este caso de uso tem por objetivo cadastrar itens perdidos, achados ou roubados |
| **Ator Primário** | Usuário |
| **Pré-condição** | Efetuar login |
| **Cenário Principal** | 1. O use case inicia quando o usuário já inseriu efetuou o login  2. O sistema seleciona a opção de cadastrar item  3. O usuário preenche formulário com descrições do item, local e data.  4. O usuário finaliza o cadastro. |
| **Pós-condição** | Não possui |
| **Cenário Alternativo** | \*a – Em qualquer momento o usuário pode sair do sistema  3a – Usuário pode cancelar o cadastramento do item  4b – Usuário não encontrou  4b.1 – O usuário fará o cadastro do item então |
| **Inclusão** |  |
| **Extensão** |  |

3.7 Na Figura 3 há uma representação estática utilizada na área da programação para descrever a estrutura de um sistema, apresentando suas classes, atributos, operações e as relações entre os objetos.

**Figura 3 –** diagrama de classe.

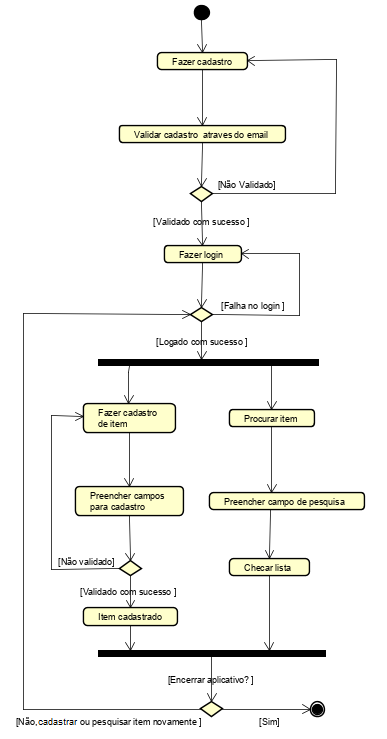
Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

3.8 Na Figura 4 mostra um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra e serão empregados para fazer a modelagem de aspectos dinâmicos do sistema.

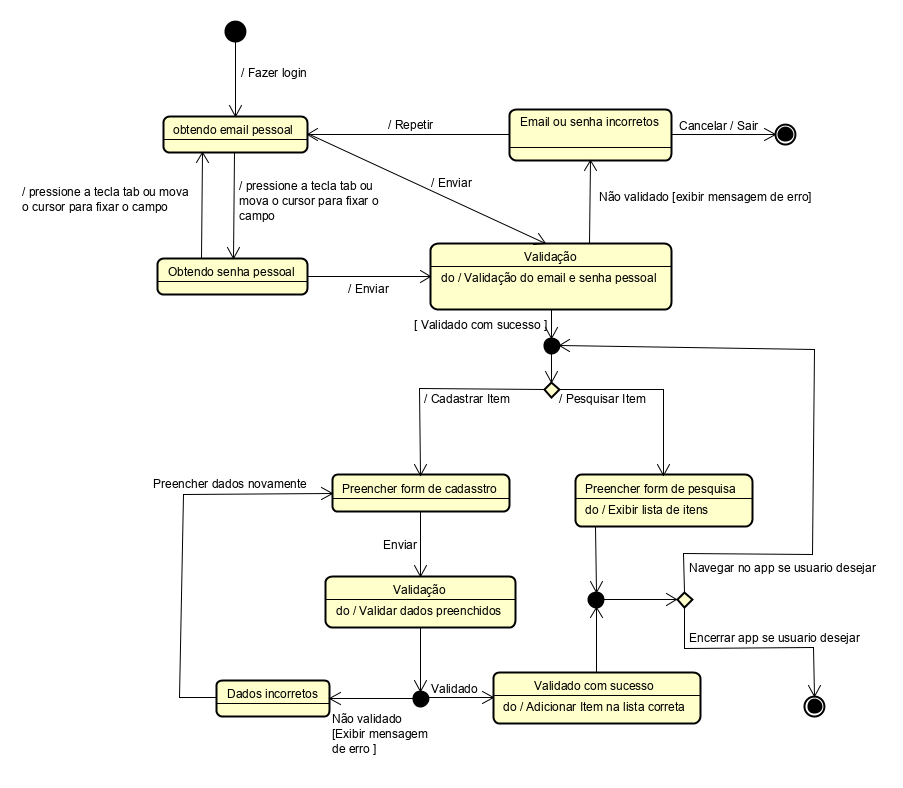
**Figura 4 –** diagrama de atividades.



**Fonte:** Autores.

3.9 Na Figura 5 mostra os possíveis estados de um objeto e as transações responsáveis pelas suas mudanças de estado.

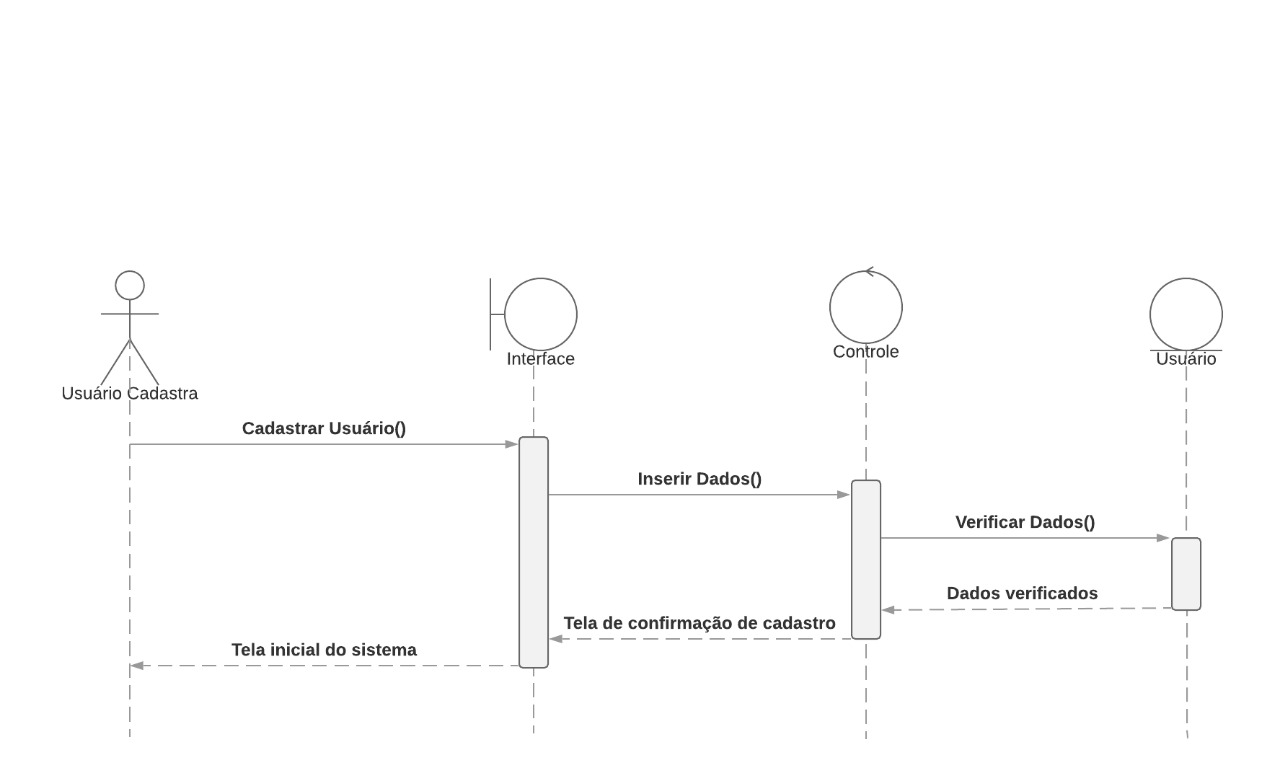
**Figura 5 –** diagrama de estado.



**Fonte:** Autores.

3.10 A Figura 6 representa a sequência de processos num programa de computador.

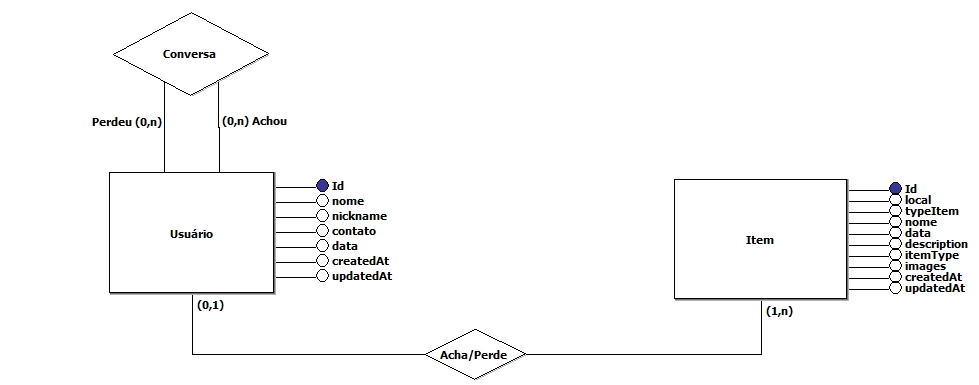
**Figura 6 –** diagrama de sequência.



**Fonte:** Autores.

3.11 A Figura 7 é um modelo conceitual utilizado na Engenharia de Software para descrever os objetos (entidades) envolvidos em um domínio de negócios, com suas características (atributos) e como elas se relacionam entre si (relacionamentos). Em geral, este modelo representa de forma abstrata a estrutura que possuirá o banco de dados da aplicação.

**Figura 7 –** diagrama de Modelo Entidade Relacionamento.



**Fonte:** Autores.

**4 Ferramentas e Métodos ou Desenvolvimento**

4.1 Ferramentas

**Visual Studio Code**

O Visual Studio Code é um editor de código-fonte leve, porém poderoso, que roda na sua área de trabalho e está disponível para Windows, macOS e Linux. Ele vem com suporte interno para JavaScript, TypeScript e Node.js e possui um rico ecossistema de extensões para outras linguagens (como C ++, C #, Java, Python, PHP, Go).

A versão utilizada foi a: 1.44.0.

Licença: Código-fonte: Licença MIT; Binários: Freeware.

O motivo para a utilização do vscode se deve pelo fato de ele ser simples de usar, extensível, e permite que você personalize da sua forma o que faz com que não ocorra grande perda de tempo procurando e instalando extensões e mexendo em configurações.

**Axios**

Axios é um cliente HTTP, que funciona tanto no browser quanto em node.js. A biblioteca é basicamente uma API que sabe interagir tanto com XMLHttpRequest quanto com a interface http do node. Isso significa que o mesmo código utilizado para fazer requisições ajax no browser também funciona no servidor. Além disso, as requisições feitas através da biblioteca retornam uma promise, compatível com a nova versão do JavaScript - ES6.

A versão utilizada foi a: 0.19.0

Licença: MIT.

O uso do Axios se deve pelo fato de ele conseguir fazer as requisições no banco de dados.

**React**

React é “uma biblioteca JavaScript declarativa, eficiente e flexível para a criação de interfaces de usuário (UI)”. Essa biblioteca surgiu em 2011, no Facebook, e passou a ser utilizada na interface do mural de notícias da rede social. No ano seguinte, passou a integrar também a área de tecnologia do Instagram e de várias outras ferramentas da empresa. Em 2013, o código foi aberto para a comunidade, o que colaborou para sua grande popularização.

A versão utilizada foi a: 16.10.1

Licença: MIT.

O uso do react se deve ao fato de ela proporcionar uma facilidade maior em relação a criação de interfaces e por já existir uma familiaridade com a ferramenta por parte dos autores do projeto.

**Node**

Node.js é uma plataforma construída sobre o motor JavaScript do Google Chrome para facilmente construir aplicações de rede rápidas e escaláveis. Node.js usa um modelo de I/O direcionada a evento não bloqueante que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos.

A versão utilizada foi a: 13.8.0

Licença: MIT.

O uso do Node se deve ao fato de que os desenvolvedores do projeto já tinham conhecimento sobre a ferramenta fazendo assim com que economizasse tempo para aprender sobre outras ferramentas necessárias.

**Express**

O Express é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel. Com uma miríade de métodos utilitários HTTP e middleware a seu dispor, criar uma API robusta é rápido e fácil. O Express fornece uma camada fina de recursos fundamentais para aplicativos da web, sem obscurecer os recursos do Node.js.

A versão utilizada foi a: 4.17.1

Licença: MIT.

O uso se deve ao fato dele trabalhar em conjunto com node.

**sequelize**

O Sequelize é um ORM (Object-Relational Mapper) para Node.js, que tem suporte aos bancos de dados PostgreSQL, MariaDB, MySQL, SQLite e MSSQL, como ORM ele faz o mapeamento de dados relacionais (tabelas, colunas e linhas) para objetos Javascript. Ele permite criar, buscar, alterar e remover dados do banco de dados utilizando métodos JS, além de permitir a modificação da estrutura das tabelas, com isso temos muita facilidade tanto na criação, população e migração de banco de dados.

A versão utilizada foi a: 5.21.4

Licença: Domínio público.

O uso se dá pela facilidade que a ferramenta te proporciona na criação de tabelas do banco de dados, você só escreve o comando no terminal passando qual são os campos que deseja criar e ele cria as tabelas pra você no banco, e cria também a model, e a migration, por exemplo.

**Bcrypt**

O BCrypt foi desenvolvido por Niels Provos e David Mazières (1999) com o propósito de esconder senhas criadas pelos usuários em forma de texto “puro” em dados indecifráveis, utilizando o algoritmo hash. Essa é uma ferramenta segura para armazenar senhas no banco de dados e pode ser utilizada por qualquer linguagem (C, C ++, C #, Go, Java, JavaScript, Elixir, Perl, PHP, Python, Ruby e outros).

A versão utilizada foi a: 3.0.8

Licença: MIT.

O uso se dá pela necessidade de criptografar senha dos usuários.

**Tedious**

Tedious é um pacote Node que fornece uma implementação do protocolo TDS (Tabular Data Sream), usado para interagir com instâncias do SQL Server da Microsoft.

A versão utilizada foi a: 8.0.1

Licença: MIT.

O uso delese deve ao fato de ele ser compatível com todas as versões do SQL Server de 2000 a 2017.

**Cross-Origin Resource Sharing(CORS)**

Cross-Origin Resource Sharing (Compartilhamento de recursos com origens diferentes) é um mecanismo que usa cabeçalhos adicionais HTTP para informar a um navegador que permita que um aplicativo Web seja executado em uma origem (domínio) com permissão para acessar recursos selecionados de um servidor em uma origem distinta. Um aplicativo Web executa uma requisição cross-origin HTTP ao solicitar um recurso que tenha uma origem diferente (domínio, protocolo e porta) da sua própria origem.

A versão utilizada foi a: 2.8.5

Licença: none.

Por motivos de segurança, navegadores restringem requisições cross-origin HTTP a partir de scripts. Por exemplo, XMLHttpRequest e Fetch API seguem a política de mesma origem (same-origin policy). Assim, um aplicativo web que faz uso dessas APIs só poderá fazer requisições HTTP da mesma origem da qual o aplicativo foi carregado.

**Mysql**

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface.

A versão utilizada foi a: 6.0.0

Licença: GPL e Licença comercial.

O uso do Mysql se deve ao fato de que os desenvolvedores do projeto já tinham conhecimento sobre a ferramenta fazendo assim com que economizasse tempo para aprender sobre outras ferramentas necessárias.

**Nodemon**

O Nodemon é um utilitário que monitora qualquer alteração na sua fonte e reinicia automaticamente o servidor.

A versão utilizada foi a: 1.19.3

Licença: MIT.

O uso da ferramenta se deve pela ajuda no desenvolvimento pois não precisa reiniciar o servidor manualmente toda vez que fizesse uma alteração no back-end.

**react-dom**

O pacote react-dom provê métodos específicos para o DOM que podem ser usados no nível superior de sua aplicação como uma válvula de escape para sair do modelo do React se você precisar.

A versão utilizada foi a: 16.10.1

Licença: MIT.

A utilização do react-dom se deve ao fato da ferramenta facilitar na manipulação de tags html.

**React-router-dom**

Esse componente server para que possamos fazer a navegação entre componetes que vão funcionar como se fossem páginas.

A versão utilizada foi a: 5.1.2

Licença: MIT.

A decisão de usar esse componente pariu pela facilidade que ele da para criar rotas poupando assim tempo para que os autores pudessem se dedicar a outras partes do projeto.

**React-scripts**

O pacote react-scripts contém tudo o que você precisa como uma sub-dependencia do seu projeto.

A versão utilizada foi a: 3.0.1

Licença: MIT.

O uso do react-scripts se deve ao fato de que os desenvolvedores do projeto já tinham conhecimento sobre a ferramenta fazendo com que economizasse tempo para aprender sobre outras ferramentas necessárias.

**Git**

O Git é um software para gerar projetos onde vários desenvolvedores podem trabalhar ao mesmo tempo. Desenvolvido em 2005 por Linus Torvalds precisamente para a criação do Kernel do Linux, hoje ele é utilizado para elaborar qualquer site, software ou código de forma rápida e inteligente. Mas o recurso que faz essa ferramenta ser tão útil e prática é o sistema de controle de alterações. Isso quer dizer que cada vez que alguém muda o código é gerada uma nova versão para o projeto. Assim, não se corre o risco de perder nenhuma informação. No Git, também não há problema com códigos sobrescritos e perda de informações, já que as versões são salvas no repositório — diretório onde são armazenados todos os arquivos do seu projeto. É possível guardar esse repositório no seu computador ou, se você utilizar uma plataforma online como GitHub, também pode ser armazenado lá.

A versão utilizada foi a: 2.17.1

Licença: GNU GPLv2.

O uso se deve ao fato do git facilitar muito o controle de alterações feitas durante o desenvolvimento do projeto.

**DBeaver**

O DBeaver é uma ferramenta universal para banco de dados. Isso significa que ele nos dá a possibilidade de manipular vários SGBDs, como Oracle, Postgres, MySQL entre outros.

A versão utilizada foi a: 6.3.4

Licença: Apache license.

O uso se deve pelo fato de ele facilitar na visualização das tabelas, suas estruturas e etc.

**Ubuntu**

Ubuntu é o nome do sistema operacional construído a partir do núcleo Linux (Linux Kernel). É um sistema de código aberto baseado nas normas do software livre.

A versão utilizada foi a: 8.04.4 LTS.

Licença: Free software + some proprietary device drivers.

O ubuntu porque Linux é mais fácil de instalar e de usar as ferramentas de programação e demanda menos processamento do hardware do computador.

**Postman**

O Postman é um aplicativo com a função de testar e desenvolver APIs em uma interface bastante simples e intuitiva. Ele nos permite simular requisições HTTP de forma rápida, armazenando-as para que possamos usá-las posteriormente.

A versão utilizada foi a: 7.22.1

Licença: none.

O uso do postman se deve ao fato de ele simular requisições de uma forma simples e rápida fazendo assim com que tempo.

4.2 Métodos ou Desenvolvimento

A Figura 8 Mostra uma função que está no front-end e é usada pra controlar as rotas que o acesso é restrito, por exemplo: só pode ter acesso a página principal se o usuário tiver feito o login, portanto mesmo que ele coloque no navegador a rota certa, por exemplo: “/principal”, se o usuário não tiver logado em sua conta, a função ira redireciona-lo pra página de login.

**Figura 8 –** Controle de login do usuário.

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

A Figura 9 é uma função chamada handleSubmit(), ela é usada no login e fica no front-end, ela é uma função assíncrona, porque ela vai lá no back-end no banco de dados buscar as algumas informações do usuário quando ele tenta fazer o login, e verifica se ele já possui um cadastro, e então ela retorna o e-mail e a senha do usuário (criptografada) e assim vai fazer o login dando acesso ao usuário a página principal, caso as informações do usuário não forem validadas, a função retorna uma mensagem de erro.

**Figura 9 –** Controle de conta existente do usuário.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

A Figura 10 esse parte do código está localizada back-end, esse código é o que conclui a criação da conta do usuário logo após ele preencher o formulário de criação de conta, esse código é usado para salvar os dados do usuario no banco de dados e aqui se faz o uso da ferramenta sequelize pois é ela quem verifica se o e-mail para cadastro do usuário já foi utilizado tornando-o assim invalido para o uso e por fim faz o cadastramento no banco de dados (linha 20), também se faz o uso do bcrypt que é usado para criptografar a senha do usuário.

**Figura 10 –** Cadastramento do usuário no banco de dados.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

A figura 11 está localizado no back-end, a onde é montada as configurações da criptografia da senha, e aqui se utiliza a biblioteca “jsonwebtoken”, pois é ela quem pega a senha original que o usuário colocou na hora em que fez o cadastro e vai criptografa-la para assim retornar o token para se possa grava-la no banco de dados, e quando necessário vai descriptografar a mesma.

**Figura 11 –** faz a criptografia da senha do usuário.

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

A figura 12 essa função está localiza no front-end, beforeLogin() é uma requisição do axios que irá fazer antes da página principal ser montada, por isso é usada na função do react componentDidMount(), porque a página não foi renderizada ainda e ele irá verificar se o token gerado no login já existe, se existir significa que o usuário está logado e aquela página pode ser rende rizada.

**Figura 12 –** Função permite a renderização do menu.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

**5 Resultados e Discussão**

Na figura 13 é apresentado a tela inicial do software, aonde possui dois campos que precisam ser preenchidos para que se possa fazer login no sistema, o primeiro campo solicita e-mail já cadastrado no sistema caso o usuário já possuir conta, e o segundo campo solicita senha também já cadastrado no sistema caso o usuário possuir conta, logo a baixo tem um check-box para que fique salvo os dados digitados e assim quando acessar a página futuramente não será necessário digitar tudo novamente, na linha a baixo possui um link para usuários que tenham perdido ou esquecido a senha assim conseguindo recupera-la, logo a baixo possui o botão “Entrar” para que os usuários consigam concluir o login, e caso o usuário não possua uma conta já cadastrada no sistema logo a baixo possui um link para que o usuário possa ser redirecionado a página correta e assim fazer o cadastro de uma conta válida.

**Figura 13 –** Tela inicial do software

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

Na figura 14 é mostrada a tela de formulário para a criação de conta dos usuários, o primeiro campo solicita para digitar o nome pessoal do usuário, no campo seguinte é solicitado para que o usuário preencha com um “nick” que é o nome que irá aparecer pros outros usuário no chat por exemplo, no campo de baixo é solicitado para que digite o endereço um e-mail valido ou um número de telefone valido que serão usados para que possa fazer login futuramente, logo a baixo é solicitado para que usuário coloque a sua data de nascimento, no campo logo a baixo, é solicitado que o usuário digite uma senha que será utilizada para fazer login futuramente, no campo de baixo é solicitado para que o usuário confirme a senha como uma medida de segurança para que não haja engano na hora de digitar a senha, e logo a baixo há um check-box que é necessário o usuário ler e aceitar os termos para que possa concluir o cadastro, e o último item da tela, é um botão que ao clicar se todos os campos tiverem sido preenchidos corretamente ele finaliza o cadastro do usuário.

**Figura 14 –** Tela de formulário para cadastro de usuários.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

Na figura 15 é visualizada a tela que aparece logo a pós o usuário fazer login, nessa tela o usuário se depara com dois botões que definem qual é a sua intenção, caso a intenção do usuário seja cadastrar um item achado ele irá clica no botão a esquerda “Cadastrar um Item”, caso o intuito do usuário seja procurar um item que perdeu, ele irá clicar no botão a direita “Ir para listagem de itens”.

**Figura 15 –** Menu, aonde usuário define qual é a intenção dele.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

A figura 16 mostra a tela de cadastro de item, aonde logo de início tem um check-box para o usuário selecionar em que categoria o cadastro dele se encontra, se ele quer cadastrar um item achado, um item perdido ou um item roubado, o próximo campo solicita ao usuário que digite o local em que o item foi achado, perdido ou roubado, sem seguida devera pôr o tipo de item, se é um smartphone, notebook e etc. No campo logo a baixo o usuário precisa colocar o nome do tipo, por exemplo: o item é um smartphone portanto ira colocar “Samsung galaxy A50”, em seguida tem o campo data que o usuário devera informa a data em que o item foi achado/perdido ou roubado, no próximo campo o usuário ira informar a descrição do item, como marcas de uso no item, cor e etc. No próximo campo o usuário tem a opção de colocar uma foto do objeto para que assim seja de maior facilidade o reconhecimento do item, e logo a baixo aparecem dois botões, o vermelho com a palavra Cancelar caso o usuário queira cancelar o cadastramento do item que estava fazendo, e o verde com a palavra Enviar caso o usuário queira Enviar os dados para serem gravados no banco de dados e assim os outros usuários terem acesso as informações preenchidas.

**Figura 16 –** Tela de cadastro de item.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:**  Autores.

A figura 17 mostra a tela aonde os usuários têm acesso aos formulários preenchidos anteriormente por usuários que acharam/ perderam ou teve seu item roubado, a tela possui um botão em cada formulário preenchido para que o usuário que está lendo as listas consiga entrar em contato com o usuário que preencheu a lista.

**Figura 17 –** Tela de visualização de itens.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fonte:** Autores.

**Considerações finais**

O objetivo inicial era criar um software que pudesse ajudar as pessoas, porque muitas passam por isso todos os dias e através do software muitos dos casos poderiam ser resolvidos de maneira rápida e simples, acreditamos que o que foi desenvolvido bate com a proposta inicial pois sempre mantivemos a proposta inicial, e os planos futuros são relacionados a fazer melhorias com base em feedbacks dos usuários.

**Referências**

Microsoft. **Visual Studio Code,** Disponível em: <httpscode.visualstudio.com>. Acesso em: 19. abr.2020.

Bernardes Marlon. **Como usar Axios como cliente HTTP**, 16/07/2015. Disponível em: <http://codeheaven.iohow-to-use-axios-as-your-http-client-pt>. Acesso em 01.Jun.2020.

Udacity Brasil. **React: o que é e como funciona essa ferramenta?,** 13/09/2018. Disponível em: <http://stableless.com.brreact-o-que-e-e-como-funciona-essa-ferramenta>. Acesso em 01. Jun. 2020.

NodeBr. **O que é Node.js?,** 14/11/2016. Disponível em: < http://nodebr.como-que-e-node-js>. Acesso em 01. Jun. 2020.

Orlandi Claudio. **Sequelize no NodeJS com ExpressJS,** Disponível em: <https://blog.rocketseat.com.br/nodejs-express-sequelize/>. Acesso em 01. Jun. 2020.

Kelyn. **Uma breve introdução sobre BCrypt,** 10/12/2019. Disponível em: <https://medium.com/reprogramabr/uma-breve-introdu%C3%A7%C3%A3o-sobre-bcrypt-f2fad91a7420>. Acesso em 01. Jun. 2020.

MDN web docs. **Cross-Origin Resource Sharing (CORS)**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Controle\_Acesso\_CORS>. Acesso em 01. Jun. 2020.

Oficina da net. **O que é MySQL?,** 06/01/2010. Disponível em: <https://www.oficinadanet.com.br/artigo/2227/mysql\_-\_o\_que\_e>. Acesso em 01. Jun. 2020.

Nodemon. **nodemon recarregar automaticamente.** Disponível em: <https://nodemon.io/>. Acesso em 01. Jun. 2020.

React. **ReactDOM**.Disponível em: <https://pt-br.reactjs.org/docs/react-dom.html>. Acesso em 01. Jun. 2020.

Felipe Davi. **React router dom**. Disponível em: <http://www.davifelipe.com.br/react-router-dom>. Acesso em 02. Jun. 2020.

Rafael. **Entendendo react-Scripts**, 07/12/2016. Disponível em <https://rafaell-lycan.com/2016/entendendo-create-react-app/>. Acesso em Acesso em 02. Jun. 2020.

Sousa Ivan. **Entenda de uma vez o que é Github e a importância dele num negócio,** 11/02/2020. Disponível em <https://rockcontent.com/blog/o-que-e-github/>. Acesso em 02. Jun. 2020.

Cuenca Maria. **Conhecendo o DBeaver**, 24/01/2020. Disponível em <https://dev.to/mgabrielacuenca/pt-br-conhecendo-o-dbeaver-2ka8>. Acesso em 02. Jun. 2020.

Significados. **Significado de Ubuntu**, 28/05/2018. Disponível em <https://www.significados.com.br/ubuntu/>. Acesso em 02. Jun. 2020.

DevMedia. **Testando APIs Web com o Postman**. Disponível em <https://www.devmedia.com.br/testando-apis-web-com-o-postman/37264>. Acesso em 02. Jun. 2020.