

**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: Profa. Dra. Jaqueline Brigladori  
Pugliesi

**FRANCA/SP  
2020**

# ACHADOS, PERDIDOS & ROUBADOS

Igor Prates Gonçalves<sup>1</sup>

Vinicius Teixeira do Carmo<sup>2</sup>

## 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. Podendo facilitar também o encontro de itens roubados através de imagens que podem ser colocadas na listagem de itens roubados para que caso alguma pessoa tenha visto consiga identificá-lo e assim passar informações como onde viu e quando viu, para a pessoa que tenha sido roubada. 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 utilizado dos locais que possuem achados e perdidos, pois com o software as pessoas irão devolver os itens encontrados diretamente aos respectivos donos, evitando assim acúmulo de itens nesses locais. O projeto visa também incentivar as pessoas a devolverem itens achados com a facilidade e rapidez que o software proporciona, pois as pessoas só irão cadastrar os itens e depois é esperar receber notificação de alguém falando que achou ou perdeu aquele respectivo item.

**Palavras-chave:** Achados. Perdidos. Roubados. Objetos. Devolução.

## Abstract

*The Lost, Stolen and Found software will receive all types of users who need to find a lost item or deliver a found item to someone else. It can also facilitate or find stolen items through images that can be placed on the list of stolen items for cases where someone has seen an account identified it and thus obtain information as always, when they saw someone who has been stolen. Based on the need to resolve unexpected situations that occurred in our daily lives, software was developed to bring these users closer. It contains lists and data that can be used by the user as an easy to use and how to use. Its interface is pleasant and harmonious, and does not require advanced knowledge for its use, the user requests information about the data requested and subsequently analyzed, or searches in a list of lists so that he can recover lost items, quickly and easily, saving search time, space available for use in places that have found and lost items, as the software is like people who use to return items found directly to users, thus avoiding local items. The project also aims to encourage people to return items found with ease and speed that the soft-*

---

<sup>1</sup> Graduando em Análise e desenvolvimento de sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: [igoorprateshotmail.com.br](mailto:igoorprateshotmail.com.br).

<sup>2</sup> Graduando em Análise e desenvolvimento de sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: [vinicius.teixeira1104@gmail.com](mailto:vinicius.teixeira1104@gmail.com).

*ware provides, as people only register the items and then receive notifications from someone who spoke or lost what they were using.*

**Keywords:** *Found. Lost. Stolen. Objects. Devolution.*

## **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, ou no caso das pessoas que tiveram seus itens furtados.

A proposta do projeto é que através do software a devolução de itens achados e perdidos fique mais fácil de ocorrer, pois a única coisa que o usuário que encontrou um determinado objeto terá que fazer é cadastrar o objeto no software e esperar a pessoa que perdeu o item mandar mensagem, e quem sabe assim influenciar as pessoas a terem mais atitudes de devolver objetos que não pertençam a elas.

O projeto funciona por meio de parcerias com locais que costumam ter um fluxo de pessoas elevado, pois são nesses lugares que geralmente ocorre 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 normalmente utilizados para armazenar os itens perdidos ou achados. Os recursos utilizados no projeto foram, computadores próprios dos autores, celulares e principalmente tempo.

A proposta do projeto é a Tecnologia da Informação como ferramenta para devolução e cadastro de itens achados, perdidos e roubados. O software será disponibilizado na internet totalmente gratuito para que qualquer usuário possa ter acesso por meio de *smartphones* ou computadores, pois o público alvo são as pessoas que acham, perdem ou tem os seus itens roubados, sendo qualquer pessoa acima de dezoito anos.

## 2 Levantamento de Requisitos

### 2.1 Elicitação e especificação dos Requisitos

O levantamento de requisitos foi feito por meio de visitas feitas em locais que 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 objetos achados, perdidos e roubados, e como esses locais também lidam com os objetos achados e quais objetos costumam aparecer nesses locais.

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

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.

### 2.2 BPMN

A Figura 1 apresenta o *Business Process Model and Notation* (BPMN) do software de Achados, perdidos e roubados, BPMN estabelece um padrão para representar os processos graficamente, por meio de diagramas. Esse padrão possui um conjunto de símbolos e regras que permite modelar diferentes fluxos de processos, com vários níveis de detalhamento (NÓBILE, 2017).

```

graph TD
    Inicio((Início)) --> CadastroLogin[Usuário cadastra login e senha no aplicativo]
    CadastroLogin --> ValidaLogin[Aplicativo valida login e senha]
    ValidaLogin --> FazLogin{faz login}
    FazLogin -- não --> CadastroLogin
    FazLogin -- usuário fez login --> Logado[logado com sucesso]
    Logado --> GeraLista[Aplicativo gera uma lista de itens de acordo com o tópico escolhido na etapa anterior]
    GeraLista --> ConsultaItens{Consulta itens}
    ConsultaItens --> ConsultaAchados[Consultar itens Achados]
    ConsultaItens --> ConsultaPerdidos[Consultar itens perdidos]
    ConsultaItens --> ConsultaRoubados[Consultar itens roubados]
    ConsultaAchados --> InsereNomeAchados[Inserir nome do item]
    ConsultaPerdidos --> InsereNomePerdidos[Inserir nome do item]
    ConsultaRoubados --> InsereNomeRoubados[Inserir nome do item]
    InsereNomeAchados --> InsereDescricaoAchados[Inserir Descrição do item]
    InsereNomePerdidos --> InsereDescricaoPerdidos[Inserir Descrição do item]
    InsereNomeRoubados --> InsereDescricaoRoubados[Inserir Descrição do item]
    InsereDescricaoAchados --> InsereFotoAchados[Inserir uma foto do item]
    InsereDescricaoPerdidos --> InsereFotoPerdidos[Inserir uma foto do item]
    InsereDescricaoRoubados --> InsereFotoRoubados[Inserir uma foto do item]
    InsereFotoAchados --> FinalizaCadastro[Finaliza cadastro]
    InsereFotoPerdidos --> FinalizaCadastro
    InsereFotoRoubados --> FinalizaCadastro
    FinalizaCadastro --> AplicaCadastro[Aplicativo cadastra o item]
    AplicaCadastro --> GeraLista
    CadastroLogin --> ConfiguraPerfil[Usuário configura perfil inserindo nome e-mail e número telefônico]
    ConfiguraPerfil --> OQueDeixaFazer{O que deseja fazer?}
    OQueDeixaFazer --> CadastroItens[Cadastrar item]
    OQueDeixaFazer --> ConsultaItens
    CadastroItens --> CadastroAchados{Cadastrar item achados}
    CadastroItens --> CadastroPerdidos{Cadastrar item perdidos}
    CadastroItens --> CadastroRoubados{Cadastrar item roubados}
    CadastroAchados --> InsereNomeAchados
    CadastroPerdidos --> InsereNomePerdidos
    CadastroRoubados --> InsereNomeRoubados
    
```

O diagrama de fluxo de dados (DFD) para o Sistema de Cadastro e Login de Usuários é composto por os seguintes elementos:

- Atores:**
  - Usuário:** Representado por um círculo verde no topo esquerdo, iniciando o processo.
  - Sistema:** Representado por um círculo vermelho no topo direito, terminando o processo.
- Processos (Retângulos azuis):**
  - Usuário cadastra login e senha no aplicativo:** O primeiro processo principal.
  - Aplicativo valida login e senha:** Recebe dados do usuário e retorna uma resposta.
  - faz login:** Um processo de decisão que verifica se o usuário fez login.
  - logado com sucesso:** Um processo que indica o sucesso do login.
  - Aplicativo gera uma lista de itens de acordo com o tópico escolhido na etapa anterior:** Gera uma lista de itens baseada no tópico selecionado.
  - Consulta itens:** Um processo de decisão que direciona o fluxo para a consulta de itens achados, perdidos ou roubados.
  - Consultar itens Achados, Consultar itens perdidos, Consultar itens roubados:** Processos que permitem ao usuário consultar itens específicos.
  - Inserir nome do item, Inserir Descrição do item, Inserir uma foto do item:** Processos de entrada de dados para cadastrar um novo item.
  - Finaliza cadastro:** O processo que finaliza o cadastro de um item.
  - Aplicativo cadastra o item:** O processo que registra o item no sistema.
- Decisões (Losângulos amarelos):**
  - O que deseja fazer?:** Decide se o usuário deseja cadastrar um item ou consultar itens.
  - faz login:** Decide se o usuário fez login com sucesso.
  - Consulta itens:** Decide qual tipo de item o usuário deseja consultar (achados, perdidos ou roubados).
  - Cadastrar item:** Decide se o usuário deseja cadastrar um item achado, perdido ou roubado.
- Fluxo de Dados:**
  - O processo começa com o **Início** e segue para **Usuário cadastra login e senha no aplicativo**.
  - Os dados do login são enviados para **Aplicativo valida login e senha**, que retorna uma resposta.
  - Se o usuário não fez login, o fluxo retorna para **Usuário cadastra login e senha no aplicativo**.
  - Se o usuário fez login com sucesso, o fluxo segue para **logado com sucesso**.
  - Depois de logado, o aplicativo gera uma lista de itens com base no tópico escolhido.
  - O usuário pode então consultar itens (achados, perdidos ou roubados) ou cadastrar um novo item.
  - Para cadastrar um item, o usuário insere o nome, a descrição e a foto, finalizando o cadastro.
  - O item é então cadastrado no sistema e a lista de itens é atualizada.
  - O processo termina com o **Fim**.

## 2.3 Requisitos Funcionais

## 2.3 Requisitos Funcionais

O Quadro 1 representa o que o software faz, em termos de tarefas e serviços. No quadro está mostrando um conjunto de funções que são explicados, descritos e seus respectivos comportamentos e as saídas.

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

<b>RF 001 – Achados</b>	Categoria: <input type="radio"/> Oculto <input checked="" type="radio"/> Evidente	Prioridade: <input type="radio"/> Altíssima <input checked="" type="radio"/> Alta <input type="radio"/> Média <input type="radio"/> Baixa
<b>Descrição:</b> 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.		
<b>RF 002 – Perdidos</b>	Categoria: <input type="radio"/> Oculto <input checked="" type="radio"/> Evidente	Prioridade: <input type="radio"/> Altíssima <input checked="" type="radio"/> Alta <input type="radio"/> Média <input type="radio"/> Baixa
<b>Descrição:</b> 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.		
<b>RF 003 – Roubados</b>	Categoria: <input type="radio"/> Oculto <input checked="" type="radio"/> Evidente	Prioridade: <input type="radio"/> Altíssima <input checked="" type="radio"/> Alta <input type="radio"/> Média <input type="radio"/> Baixa
<b>Descrição:</b> Na tela de Roubados irá conter a lista com os itens que foram cadastrados e que se enquadra 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.		
<b>RF 004 – Cadastrar Achado</b>	Categoria: <input type="radio"/> Oculto <input checked="" type="radio"/> Evidente	Prioridade: <input checked="" type="radio"/> Altíssima <input type="radio"/> Alta <input type="radio"/> Média <input type="radio"/> Baixa
<b>Descrição:</b> 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.		
<b>RF 005 – Cadastrar Perdido</b>	Categoria: <input type="radio"/> Oculto <input checked="" type="radio"/> Evidente	Prioridade: <input checked="" type="radio"/> Altíssima <input type="radio"/> Alta <input type="radio"/> Média <input type="radio"/> Baixa
<b>Descrição:</b> 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.		
<b>RF 006 – Cadastrar Roubado</b>	Categoria: <input type="radio"/> Oculto	Prioridade: <input checked="" type="radio"/> Altíssima

	(X) Evidente	( ) Alta ( ) Média ( ) Baixa
<b>Descrição:</b> 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.		
<b>RF 007 – Tipo do Item</b>	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima (X) Alta ( ) Média ( ) Baixa
<b>Descrição:</b> 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.		
<b>RF 008 – Formulário</b>	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima (X) Alta ( ) Média ( ) Baixa
<b>Descrição:</b> 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.		
<b>RF 009 – Enviar</b>	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima (X) Alta ( ) Média ( ) Baixa
<b>Descrição:</b> 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.		
<b>RF 0010 – Termos de responsabilidade</b>	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima (X) Alta ( ) Média ( ) Baixa
<b>Descrição:</b> Link na página de cadastro de usuário que irá abrir uma página mostrando os termos de uso do software e a política de privacidade.		

## 2.4 Requisitos Não Funcionais

O Quadro 2 apresenta os requisitos não funcionais, que são os 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.**

Nome	Restrição	Categoria	Obrigatoriedade	Permanência
<b>RNF001 - Cadastro</b>	O sistema deverá exigir um cadastro de usuário.	(X) Oculto ( ) Evidente	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório

<b>RNF002</b> - Login	O sistema deverá exigir o login do usuário já cadastrado.	(X) Oculto ( ) Evidente	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
<b>RNF003</b> – 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
<b>RNF004</b> – Termo de Responsabilidade	O usuário deverá aceitar os Termos de Responsabilidade.	(X) Oculto ( ) Evidente	(X ) Desejável ( ) Obrigatório	(X) Permanente ( ) Transitório

## 2.5 Regras de Negócio

O Quadro 3 mostra as regras de negócio do sistema, que definem 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 válido.

## 2.6 Casos de Uso

Caso de Uso demonstra como uma funcionalidade é utilizada, como ela se comportará diante de eventos, inputs, exceções etc.

Os casos de uso definidos para este sistema são:

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.

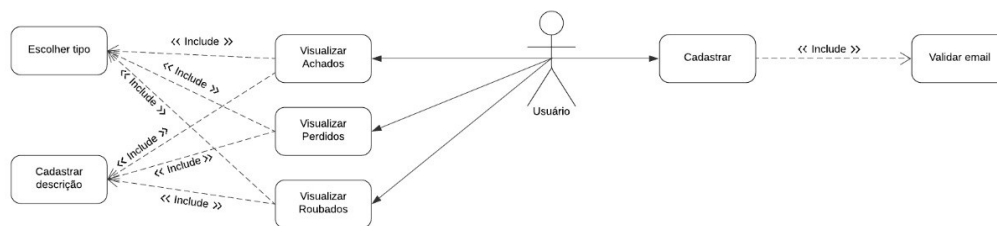
UC 005 – Visualizar lista de Roubados.

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

Na Figura 2 está representada 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.**



**Fonte:** Autores.

O Quadro 4 apresenta como deve funcionar o cadastro de usuários, seus cenários principais e cenários alternativos.

**Quadro 4 – Use Case Cadastrar Usuários**

<b>Caso de Uso – Cadastrar Usuário</b>	
<b>ID</b>	UC 001 – Cadastrar Usuário
<b>Descrição</b>	Este caso de uso tem por objetivo realizar o cadastro do usuário.
<b>Ator Primário</b>	Usuário
<b>Pré-condição</b>	Não possui
<b>Cenário Principal</b>	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
<b>Pós-condição</b>	Ter validado o e-mail
<b>Cenário Alternativo</b>	*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
<b>Inclusão</b>	
<b>Extensão</b>	

No Quadro 5 está explicado que o usuário irá optar por visualizar uma das listas sendo as opções a lista de itens achados, itens perdidos ou itens roubados.

**Quadro 5 – Use Case definir lista**

<b>Caso de Uso – Definir lista</b>	
<b>ID</b>	UC 002 – Definir qual lista deseja ter acesso (lista de achados, perdidos ou roubados)
<b>Descrição</b>	Este caso de uso tem por objetivo definir qual lista o usuário quer visualizar.
<b>Ator Primário</b>	Usuário
<b>Pré-condição</b>	Estar logado no sistema
<b>Cenário Principal</b>	1. O use case inicia quando o usuário faz login 2. O sistema mostra duas 3 opções

	3. O usuário seleciona a que deseja
<b>Pós-condição</b>	Não possui
<b>Cenário Alternativo</b>	*a – Em qualquer momento o usuário pode sair do sistema
<b>Inclusão</b>	
<b>Extensão</b>	

O Quadro 6 mostra como deve funcionar a lista de itens achados, onde é mostrado o cenário principal, e possíveis cenários alternativos.

**Quadro 6 – Use Case Visualizar Achados**

<b>Caso de Uso – Visualizar Achados</b>	
<b>ID</b>	UC 003 - Visualizar Achados
<b>Descrição</b>	Este caso de uso tem por objetivo visualizar a lista de Achados
<b>Ator Primário</b>	Usuário
<b>Pré-condição</b>	Efetuar login
<b>Cenário Principal</b>	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
<b>Pós-condição</b>	Não possui
<b>Cenário Alternativo</b>	*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
<b>Inclusão</b>	
<b>Extensão</b>	

O Quadro 7 apresenta 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**

<b>Caso de Uso – Visualizar Perdidos</b>	
<b>ID</b>	UC 004 - Visualizar Perdidos
<b>Descrição</b>	Este caso de uso tem por objetivo visualizar a lista dos Perdidos
<b>Ator Primário</b>	Usuário
<b>Pré-condição</b>	Efetuar login
<b>Cenário Principal</b>	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
<b>Pós-condição</b>	Não possui
<b>Cenário Alternativo</b>	*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
<b>Inclusão</b>	
<b>Extensão</b>	

No Quadro 8 tem-se 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**

<b>Caso de Uso – Visualizar Roubados</b>	
<b>ID</b>	UC 005 – Visualizar Roubados
<b>Descrição</b>	Este caso de uso tem por objetivo visualizar a lista dos Roubados
<b>Ator Primário</b>	Usuário
<b>Pré-condição</b>	Efetuar login
<b>Cenário Principal</b>	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
<b>Pós-condição</b>	Não possui
<b>Cenário Alternativo</b>	*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
<b>Inclusão</b>	
<b>Extensão</b>	

O Quadro 9 mostra como deve funcionar o cadastramento de itens achados perdidos ou roubados.

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

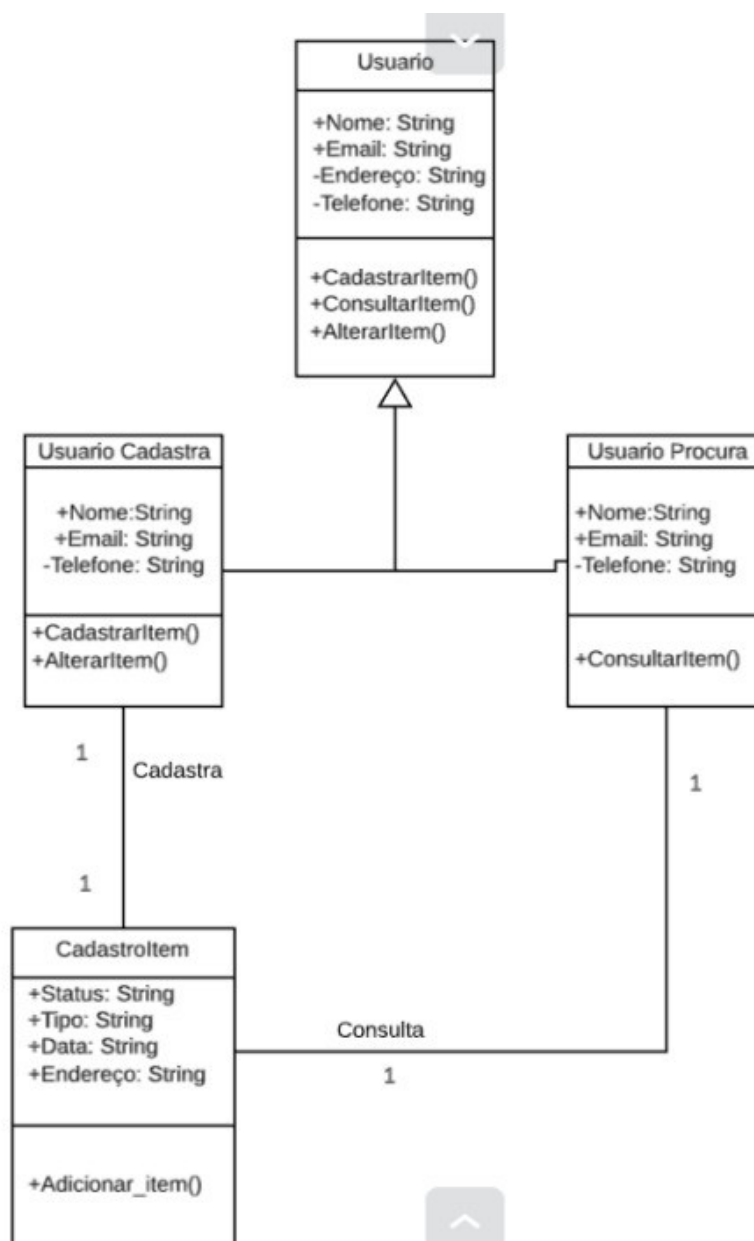
<b>Caso de Uso – Cadastro de item</b>	
<b>ID</b>	UC 006 – cadastrar item
<b>Descrição</b>	Este caso de uso tem por objetivo cadastrar itens perdidos, achados ou roubados
<b>Ator Primário</b>	Usuário
<b>Pré-condição</b>	Efetuar login
<b>Cenário Principal</b>	1. O use case inicia quando o usuário já inseriu 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.
<b>Pós-condição</b>	Não possui
<b>Cenário Alternativo</b>	*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

<b>Inclusão</b>	
<b>Extensão</b>	

## 2.7 Diagrama de Classes

A Figura 3 apresenta o diagrama de classe do sistema, que é 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.

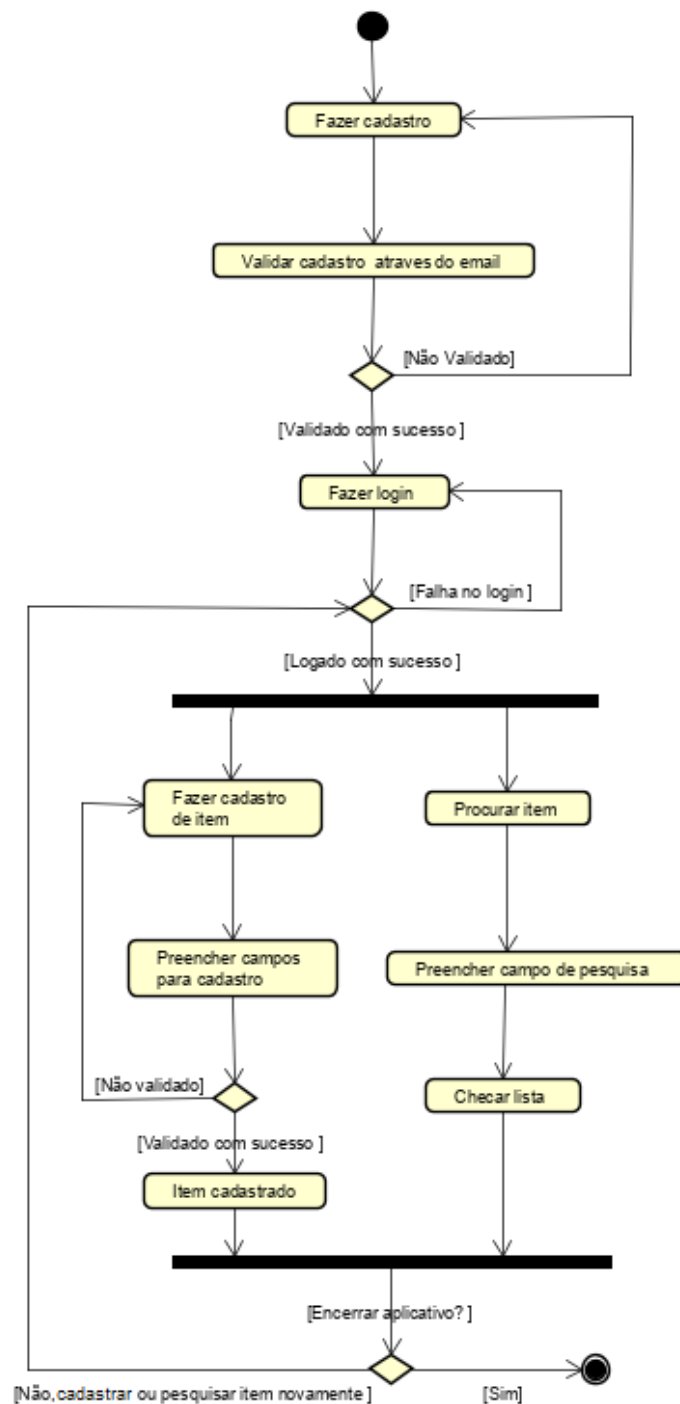


Fonte: Autores.

## 2.8 Diagrama de Atividades

Na Figura 4 pode-se observar o diagrama de atividades do sistema, que é 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.**

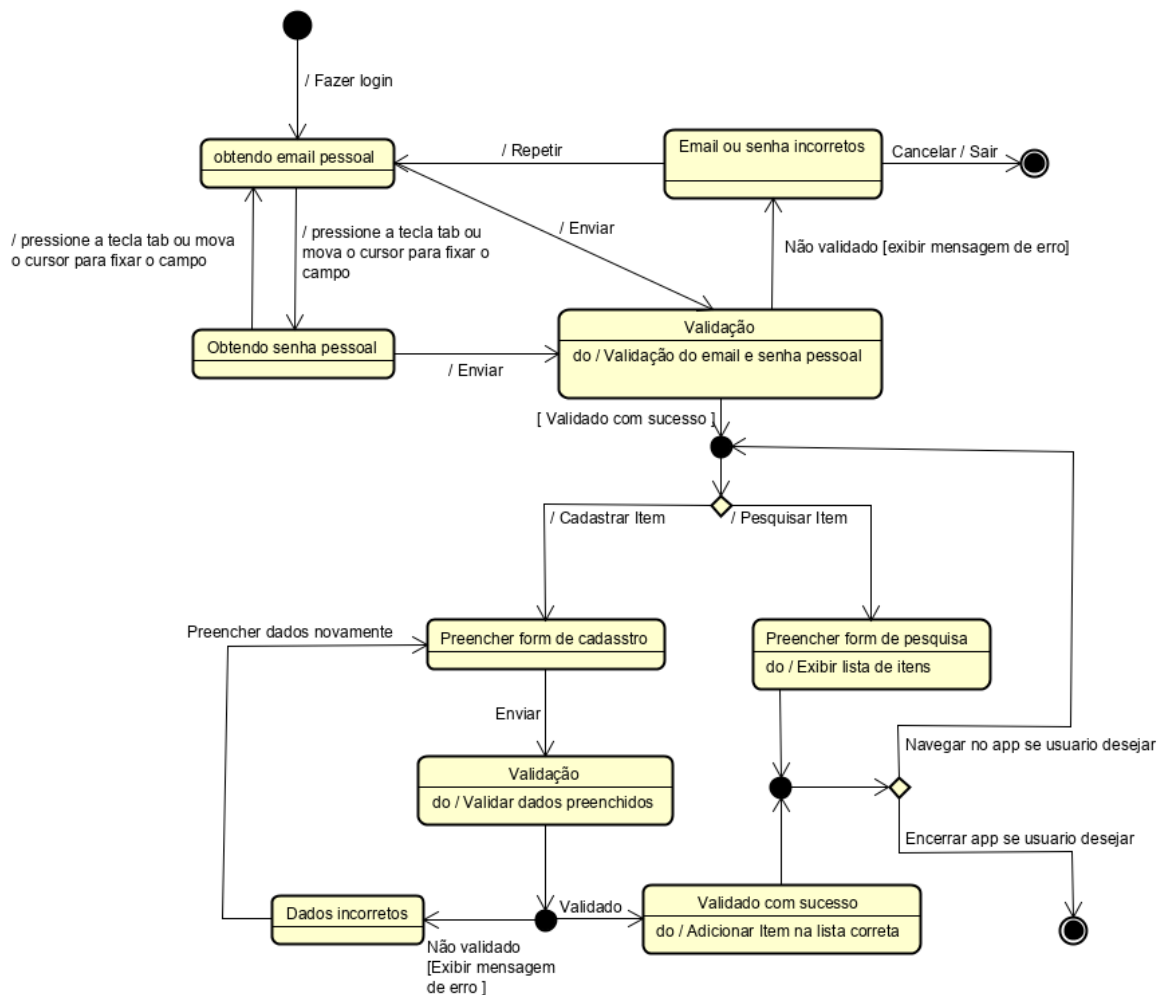


Fonte: Autores.

## 2.9 Diagrama de Estados

Na Figura 5 está o digrama de estado, que mostra os possíveis estados de um objeto e as transações responsáveis pelas suas mudanças de estado.

Figura 5 – Diagrama de estado.

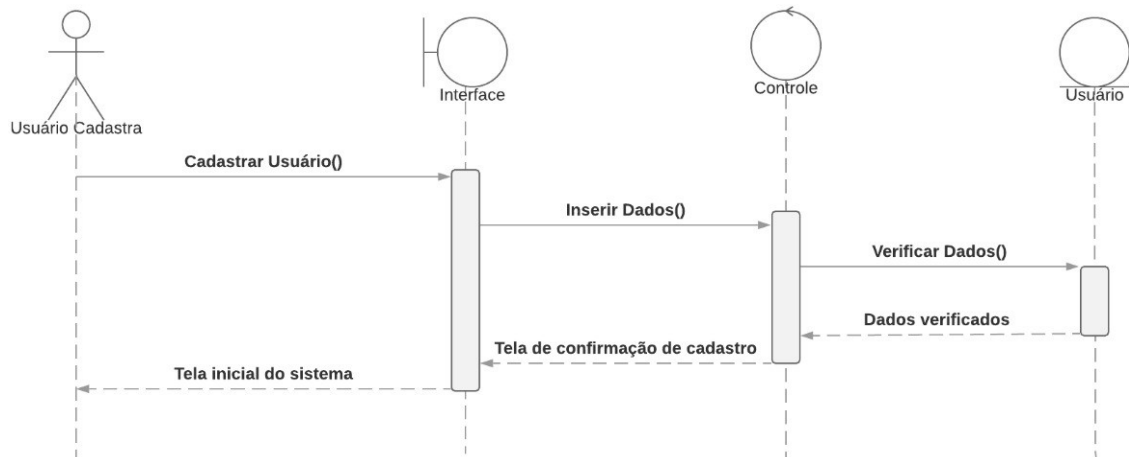


Fonte: Autores.

## 2.10 Diagrama de Sequência

Na Figura 6 está o diagrama de sequência, que apresenta a sequência de processos num programa de computador.

**Figura 6 – Diagrama de sequência.**

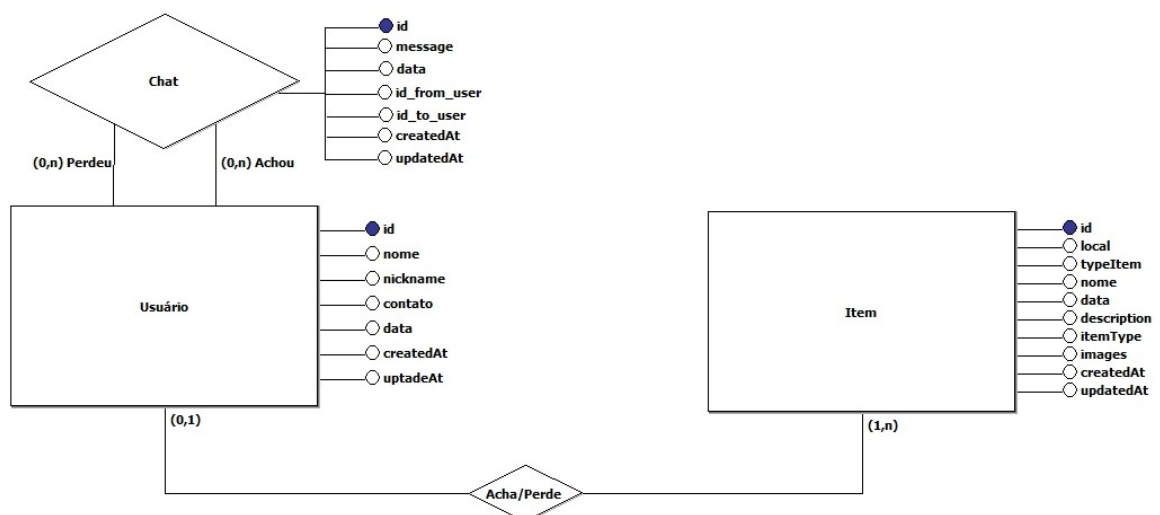


**Fonte:** Autores.

## 2.11 Diagrama Entidade-Relacionamento

A Figura 7 apresenta o diagrama entidade-relacionamento desenvolvido para o sistema, que 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.

### 3 Ferramentas e Métodos ou Desenvolvimento

#### 3.1 Ferramentas

##### 3.1.1 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) (MICROSOFT, *sd*). A versão utilizada da ferramenta no software foi a: 1.44.0. E a sua licença é: Código-fonte: Licença MIT; Binários: Freeware.

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

##### 3.1.2 Axios

Axios é um cliente HTTP (*Hypertext Transfer Protocol*), que funciona tanto no *browser* quanto em node.js. A biblioteca é basicamente uma API (*Application Programming Interface*) 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 (MARLON, 2015). A versão utilizada foi a: 0.19.0 e a licença da ferramenta é a licença: MIT.

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

##### 3.1.3 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 (UDACITY BRASIL, 13/09/2018). No software a versão utilizada foi a: 16.10.1, e a ferramenta possui a licença: MIT.

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

#### 3.1.4 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 (NODEBR, 14/11/2016). No software a versão utilizada foi a: 13.8.0 e a ferramenta possui a 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.

#### 3.1.5 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* disponível, 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 (EXPRESS, sd). A versão que foi utilizada no software foi a: 4.17.1 e a licença que a ferramenta possui é a licença: MIT.

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

#### 3.1.6 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 (CLAUDIO, sd). O software utilizou a versão: 5.21.4 da ferramenta, e ela possui a licença: Domínio público.

O uso se dá pela facilidade que a ferramenta 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.

### 3.1.7 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) ( KELYN, 10/12/2019).a versão utilizada no software foi a: 3.0.8 e a licença da ferramenta é a: MIT.

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

### 3.1.8 Tedious

Tedious é um pacote node que fornece uma implementação do protocolo TDS (*Tabular Data Stream*), usado para interagir com instâncias do SQL Server da Microsoft (PILSBURY, sd). A versão utilizada no software foi a: 8.0.1 e a licença: MIT.

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

### 3.1.9 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 (MDN, sd). No software versão utilizada da ferramenta foi a: 2.8.5 e a licença da ferramenta é a: 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.

#### 3.1.10 Mssql

O MS SQL Server é um SGBD – sistema gerenciador de Banco de Dados Relacional desenvolvido pela Microsoft. Criado em parceria com a SYBASE em 1988 inicialmente para a plataforma OS/2<sup>1</sup>. Parceria que durou até 1994, com o lançamento da versão para Windows NT e desde então a Microsoft mantém a manutenção do produto Como um Banco de Dados (DIGICARD, 05/05/2014). A versão da ferramenta utilizada no software foi a: 6.0.0 e a sua licença é a: Licença comercial.

O uso do Mssql 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.

#### 3.1.11 Nodemon

O Nodemon é um utilitário que monitora qualquer alteração na sua fonte e reinicia automaticamente o servidor (NODEMON, sd). A versão utilizada no software foi a: 1.19.3 e a sua licença é a 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*.

#### 3.1.12 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 (REACT, sd). A versão que foi utilizada no software foi a versão: 16.10.1 e a sua licença é a: MIT.

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

#### 3.1.13 React-router-dom

O componente React-router-dom serve para que possamos fazer a navegação entre componentes que vão funcionar como se fossem páginas (FELIPE, sd). No software foi utilizado a versão: 5.1.2 e a sua licença é a licença: MIT.

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

#### 3.1.14 React-scripts

O pacote react-scripts contém tudo o que se precisa como uma subdependência do seu projeto (RAFAEL, 07/12/2016). A versão da ferramenta utilizada foi a: 3.0.1 e a sua licença é a: 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.

#### 3.1.15 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. Porém, 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, que é o 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 (SOUSA, 11/02/2020). No software versão utilizada foi a: 2.17.1 e a sua licença é a: GNU GPLv2.

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

#### 3.1.16 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 (CUENCA, 24/01/2020). A versão utilizada no software foi a: 6.3.4 e a sua licença é a: Apache license.

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

#### 3.1.17 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 (SIGNIFICADOS, 28/05/2018). A versão utilizada no software foi a: 8.04.4 LTS e a licença da ferramenta é a: Free software + some proprietary device drivers.

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

#### 3.1.18 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 (DEV MEDIA, sd). A versão que foi utilizada no software foi a: 7.22.1 e a licença que a ferramenta possui é: none.

O uso do postman se deve ao fato de ele simular requisições de uma forma simples e rápida fazendo assim com que fosse possível poupar tempo para dedicar a apreender sobre outras ferramentas que foi utilizada.

### 3.2 Métodos ou Desenvolvimento

A sequência de figuras a seguir irá mostrar algumas partes mais importantes dos códigos que foram utilizados no software, para.

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 irá redirecioná-lo pra página de login.

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

```
front-end > src > assets > routes > privateRoutes.js > ...
1  import React from "react";
2  import { Route, Redirect } from "react-router-dom";
3
4  const PrivateRoutes = ({ component: Component, isAuthenticated, ...rest }) => (
5    <Route
6      {...rest}
7      render={props => (
8        isAuthenticated ? <Component {...props} /> : <Redirect to="/" />
9      )}
10    />
11  );
12
13  export default PrivateRoutes;
```

**Fonte:** Autores.

A Figura 9 mostra uma função chamada `handleSubmit()`, que é usada no login e fica no *front-end*. Ela é uma função assíncrona, porque ela vai no *back-end* no Banco de Dados buscar as 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 à 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.

```
front-end > src > assets > pages > Login.js > ...  
/  
8  class Login extends React.Component {  
9  
10     constructor(props) {  
11         super(props);  
12         this.handleSubmit = this.handleSubmit.bind(this);  
13     }  
14  
15     state = {  
16         contato: "",  
17         password: "",  
18         error: "",  
19         checkbox: true  
20     }  
21  
22     async handleSubmit(e) {  
23         e.preventDefault();  
24  
25         const response = await api.post('/signup', {  
26             contato: this.state.contato,  
27             password: this.state.password  
28         })  
29         .then(res => {  
30             this.props.authenticated();  
31             localStorage.setItem('token', res.data.token);  
32             this.setLocalStorage();  
33             this.props.history.push('/main');  
34         })  
35         .catch(error => {  
36             this.setState({ error: "Usuário ou senha inválidos!" });  
37         })  
38     }  
}
```

**Fonte:** Autores.

Na Figura 10, essa parte do código está localizada no *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 usuário 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 inválido 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.




```
back-end > controllers > UserController.js > store > user > where
1  const Sequelize = require('sequelize');
2  const DataTypes = Sequelize.DataTypes;
3  const data = require('../config/config.json');
4  const dbDev = data.development;
5  const bcrypt = require('bcrypt');
6  const sequelize = new Sequelize(dbDev.database, dbDev.username, dbDev.password, {
7    host: dbDev.host,
8    dialect: dbDev.dialect
9  });
10 const User = require('../models/usuario.js')(sequelize, DataTypes);
11
12 async function store(req, res) {
13   let { nome, nickname, contato, dia, mes, ano, senha } = req.body;
14   let user = await User.findOne({ where: { contato: contato } });
15
16   if (user) {
17     return res.status(200).send('Dados inválidos');
18   }
19
20   user = await User.create({
21     nome: nome,
22     nickname: nickname,
23     contato: contato,
24     dia: dia,
25     mes: mes,
26     ano: ano,
27     senha: bcrypt.hashSync(senha, 8)
28   });
29
30   res.status(201).send('Usuário criado!');
31 }
32
33 module.exports = { store };
```

**Fonte:** Autores.

Na Figura 11 está localizado no *back-end*, no qual são montadas 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 criptografá-la para assim retornar o *token* para se possa gravá-la no Banco de Dados, e quando necessário vai descriptografar a mesma.



**Figura 11** – Criptografia da senha do usuário.

```
back-end > routes > middleware >  middleware.js >  withAuth >  jwt.verify() callback
1  const jwt = require('jsonwebtoken');
2  const secret = 'shhhhh';
3  const withAuth = function(req, res, next) {
4      const token = req.headers['authorization'].replace(/^JWT\s/, '');
5
6      if (!token) {
7          res.status(401).send('Unauthorized: No token provided');
8      } else {
9          jwt.verify(token, secret, function(err, decoded) {
10             if (err) {
11                 res.status(401).send('Unauthorized: Invalid token');
12             } else {
13                 req.contato = decoded.contato;
14                 next();
15             }
16         });
17     }
18 }
19 module.exports = withAuth;
```

**Fonte:** Autores.

A Figura 12 mostra uma função localizada no *front-end*. A `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 renderizada.

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

```
front-end > src > assets > pages > Main.js > Main > componentDidMount
1  import React, { Component } from 'react';
2  import './Main.css';
3  import axios from 'axios';
4  import { withRouter } from 'react-router-dom';
5
6  class Main extends React.Component {
7    componentDidMount() {
8      this.beforeLogin();
9    }
10
11    async beforeLogin() {
12      const axiosInstance = axios.create({
13        baseURL: 'http://127.0.0.1:8000/',
14        headers: {
15          Authorization: localStorage.getItem('token')
16        }
17      });
18
19      const response = await axiosInstance({ method: 'GET', url: '/user' })
20        .then(res => {
21          })
22        .catch(error => {
23          })
24    }
25  }
```

Fonte: Autores.

## 4 Resultados e Discussão

Nessa sequência de figuras, será mostrado as telas que apareceram pros usuários e suas respectivas funcionalidades.

Na Figura 13 é apresentada a tela inicial do software, que possui dois campos que precisam ser preenchidos para que se possa fazer login no sistema. O primeiro campo solicita o e-mail já cadastrado no sistema, caso o usuário já possua conta, e o segundo campo solicita senha também já cadastrada no sistema, caso o usuário possua conta. Logo abaixo tem um *check-box* para que fiquem salvos os dados

digitados e assim quando acessar a página futuramente não será necessário digitar tudo novamente. A linha abaixo possui um link para usuários que tenham perdido ou esquecido a senha assim conseguindo recuperá-la, e o botão “Entrar” para que os usuários consigam concluir o login. Caso o usuário não possua uma conta já cadastrada no sistema, também 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

A screenshot of the initial login screen of a software application. The background is a vibrant purple and blue gradient. In the center, there is a white rounded rectangle containing the login form. The title 'Achados, Perdidos & Roubados' is displayed in a bold, black, sans-serif font. Below the title are two input fields: 'Email ou Usuário' and 'Senha', both with light gray placeholder text. A black button with the white text 'Entrar' is positioned below the input fields. At the bottom of the white rectangle, there is a link that reads 'Ainda não é cadastrado? Cadastro-se' in a smaller, gray font.

**Fonte:** Autores.

Na Figura 14 é mostrada a tela de formulário para a criação de conta dos usuários. O primeiro campo solicita o nome pessoal do usuário, e no campo seguinte é solicitado para que o usuário preencha com um “nick” que é o nome que irá aparecer para os outros usuários no chat, por exemplo. No campo abaixo é solicitado o endereço de um e-mail válido ou um número de telefone válido que serão usados para que possa fazer login futuramente. Em seguida é solicitada sua data de nascimento e uma senha que será utilizada para fazer login futuramente. No outro campo é solicitado que o usuário confirme a senha como uma medida de segurança para que não haja engano na hora de digitar a senha. Logo abaixo há um *check-box* que é necessário o usuário ler e aceitar os termos para que possa concluir o cadastro. 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.

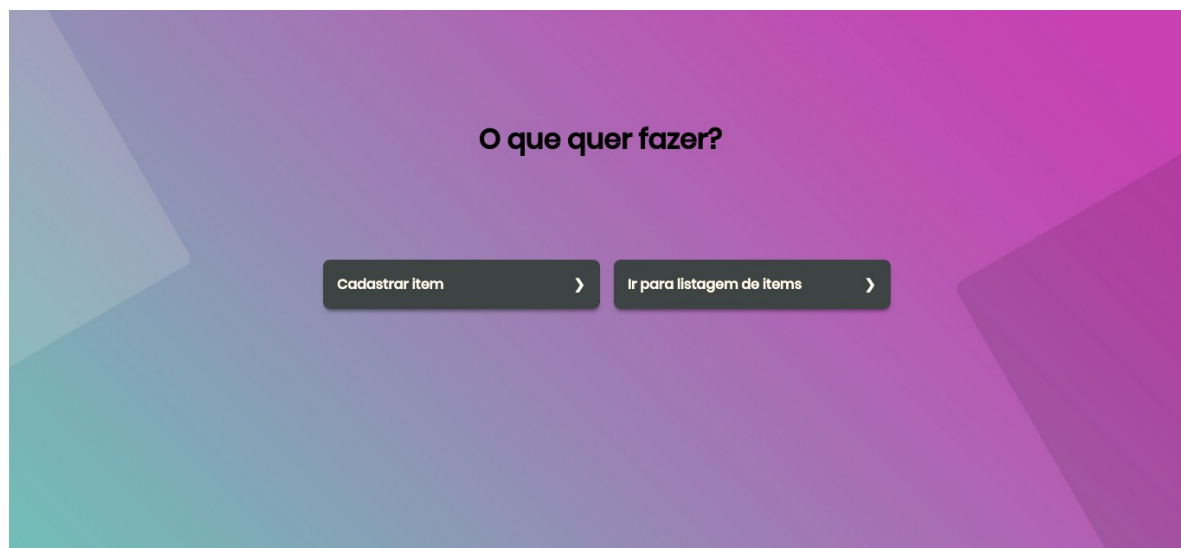
O formulário de cadastro está centralizado em uma tela com fundo abstrato em tons de roxo e azul. O formulário em si é branco com bordas arredondadas e contém o seguinte conteúdo:

- Título:** *Achados, Perdidos & Roubados*
- Campos de entrada:**
  - Nome (campo único)
  - Usuário (campo único)
  - Email ou Telefone (campo único)
  - Seletores para dia, mês e ano (ex: 1, Janeiro, 2002)
  - Nova senha (campo único)
  - Confirmar senha (campo único)
- Termos de Uso:** ☐ Li e aceito os Termos de Uso e a Política de Privacidade
- Botão:** Cadastro (botão preto com texto branco)

**Fonte:** Autores.

Na Figura 15 é mostrada a tela que aparece logo após 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 clicar 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.



**Fonte:** Autores.

A Figura 16 mostra a tela de cadastro de item, na qual, 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, e em seguida deverá inserir o tipo de item, se é um *smartphone*, *notebook* e etc. No campo logo abaixo o usuário precisa colocar o nome do tipo, por exemplo: se o item é um *smartphone*, portanto irá colocar "Samsung galaxy A50". Em seguida tem o campo data que o usuário deverá informar a data em que o item foi achado, perdido ou roubado. No próximo campo, o usuário irá informar a descrição do item, como marcas de uso no item, cor e etc. A seguir o usuário tem a opção de colocar uma foto do objeto para que assim seja mais fácil o reconhecimento do item. Logo abaixo 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 às informações preenchidas.

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

Fonte: Autores.

A Figura 17 mostra a tela que os usuários podem acessar os formulários preenchidos anteriormente por usuários que acharam, perderam ou tiveram 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.

Fonte: Autores.

## Considerações finais

O objetivo inicial era criar um aplicativo mobile que pudesse ajudar as pessoas, porque muitas passam por isso todos os dias e por meio do aplicativo muitos dos casos poderiam ser resolvidos de maneira rápida e simples. Porém não foi possível desenvolver um aplicativo e, portanto, os autores optaram por continuar com o projeto, mas para web assim conseguindo continuar de certa forma com a proposta do projeto.

Acreditamos que conseguimos manter a proposta do projeto pois sempre mantivemos o objetivo inicial como o foco principal.

Os objetivos futuros estão relacionados a fazer melhorias com base em *feedbacks* dos usuários e conseguir desenvolver o aplicativo fazendo com que ele se torne ainda mais fácil e simples para o uso.

## Referências

BERNARDES MARLON. **Como usar Axios como cliente HTTP**, 16/07/2015. Disponível em: <<http://codeheaven.io/how-to-use-axios-as-your-http-client-pt>>. Acesso em 01.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.

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

DIGICARD. **MS SQL Server, o que é , como funciona e para quem é direcionado**, 05/05/2014. Disponível em: <[https://www.oficinadanet.com.br/artigo/2227/mysql\\_-\\_o\\_que\\_e](https://www.oficinadanet.com.br/artigo/2227/mysql_-_o_que_e)>. Acesso em 01. Jun. 2020.

EXPRESS. **Express Framework web rápido, flexível e minimalista para Node.js**, sd. Disponível em: <<https://expressjs.com/pt-br/>>. Acesso em 18. Jun. 2020.

FELIPE DAVI. **React router dom**, sd. Disponível em: <<http://www.davifelipe.com.br/react-router-dom>>. Acesso em 02. 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)**, sd. Disponível em: <[https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Controle\\_Acesso\\_CORS](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Controle_Acesso_CORS)>. Acesso em 01. Jun. 2020.

MICROSOFT. **Visual Studio Code**, sd. Disponível em: <<https://code.visualstudio.com>>. Acesso em: 19.abr.2020.

NÓBILE VINICIUS. **O que é BPMN (Business Process Model and Notation) e como aplicar essa notação na Modelagem de Processos**, 08/02/2017. Disponível em <<https://www.euax.com.br/2017/02/o-que-e-bpmn-business-process-model-and-notation/>>. Acesso em 18. 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.

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

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

PILSBURY MIKE. **Tedious**, sd. Disponível em <<https://tediousjs.github.io/tedious/>>. Acesso em 18. Jun. 2020.

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

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

SIGNIFICADOS. **Significado de Ubuntu**, 28/05/2018. Disponível em <<https://www.significados.com.br/ubuntu/>>. 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.