**Documentação do Software**

**Nome do Software:** floatData

**Nome da Equipe: Innocode Solutions**

**Integrantes: André Flávio de Oliveira**

**Igor Vinicius Santos Fonseca**

**Jonatas Filipe Carvalho Ferreira**

**Mauro do Prado Santos**

**Samuel Lucas Vieira de Melo**

**Vitor Cezar de Souza**

**1. Introdução**

**1.1. Objetivo do Projeto**

O objetivo do projeto floatData é desenvolver uma solução completa para o monitoramento de dispositivos de rastreamento móveis, conhecidos como "derivadores". A solução engloba a criação de um dispositivo IoT para captura de dados e um sistema (web e mobile) para visualização em tempo real da geolocalização, velocidade e trajetória dos ativos rastreados.

**1.2. Escopo do Sistema**

O sistema é um aplicativo desenvolvido para Android e Web, com foco em fornecer uma plataforma de monitoramento de ativos em tempo real. Ele se comunica com um dispositivo IoT, recebe e processa dados de geolocalização, e os apresenta de forma intuitiva ao usuário final através de mapas interativos e relatórios.

**1.3. Público-Alvo**

Estudantes da Universidade Federal de Itajubá que fazem parte do Projeto Rio Doce.

**1.4. Siglas e/ou Abreviações**

* API: Interface de Programação de Aplicações
* APK: Android Package Kit
* CRUD: Create, Read, Update, Delete
* IoT: Internet of Things (Internet das Coisas)
* JWT: JSON Web Token
* SGBD: Sistema Gerenciador de Banco de Dados
* UI/UX: Interface/Experiência do Usuário

**2. Visão Geral do Sistema**

**2.1. Funcionalidades Principais**

* Cadastro e autenticação de usuários: Sistema de login seguro com e-mail e senha.
* Recuperação de senha: Mecanismo para que o usuário redefina sua senha.
* Dashboard com Mapa Interativo: Visualização da localização em tempo real de todos os dispositivos associados ao usuário.
* Cálculo de Velocidade e Trajetória: O sistema processa os dados recebidos para exibir a velocidade e o histórico de percurso do dispositivo.
* Relatórios e Exportação: Geração de relatórios com dados históricos e funcionalidade para exportá-los.
* Gerenciamento de Dispositivos: Tela dedicada para o usuário visualizar seus dispositivos cadastrados.
* Integração com Banco de Dados em Nuvem: Armazenamento centralizado e escalável das informações.
* Interface Responsiva: Compatibilidade com dispositivos móveis e desktops.

**2.2. Plataformas Suportadas**

* Android: (versão mínima sugerida: 5.0 Lollipop)
* Web: Acessível através de navegadores modernos em desktops e dispositivos móveis.
* iOS: O desenvolvimento em React Native permite a compilação para iOS, embora o foco inicial tenha sido a entrega para Android.

**3. Requisitos do Sistema**

**3.1. Requisitos Funcionais**

* **RF001:** O sistema deve coletar e armazenar coordenadas GPS;
* **RF002:** O sistema deve transmitir os dados coletados via rede celular.
* **RF003:** O sistema deve permitir que uma estação receba os dados transmitidos via rede celular e

os armazene em um SGBD.

* **RF004:** O sistema deve restringir o acesso a usuários autenticados.
* **RF005:** O sistema deve permitir a visualização do histórico de localizações por derivador e período.
* **RF006:** O sistema deve possibilitar o download de dados em formato CSV por derivador e período.

**3.2. Requisitos Não Funcionais**

* **RNF001:** O sistema deve permitir acesso aos dados por aplicativo para móveis.
* **RNF002:** O sistema deve apresentar as localizações dos derivadores em mapas interativos.
* **RNF003:** A interface do sistema deve ser responsiva.
* **RNF004:** O sistema deve incluir uma tela explicativa sobre o projeto.

**4. Arquitetura do Sistema**

**4.1. Tecnologias Utilizadas**

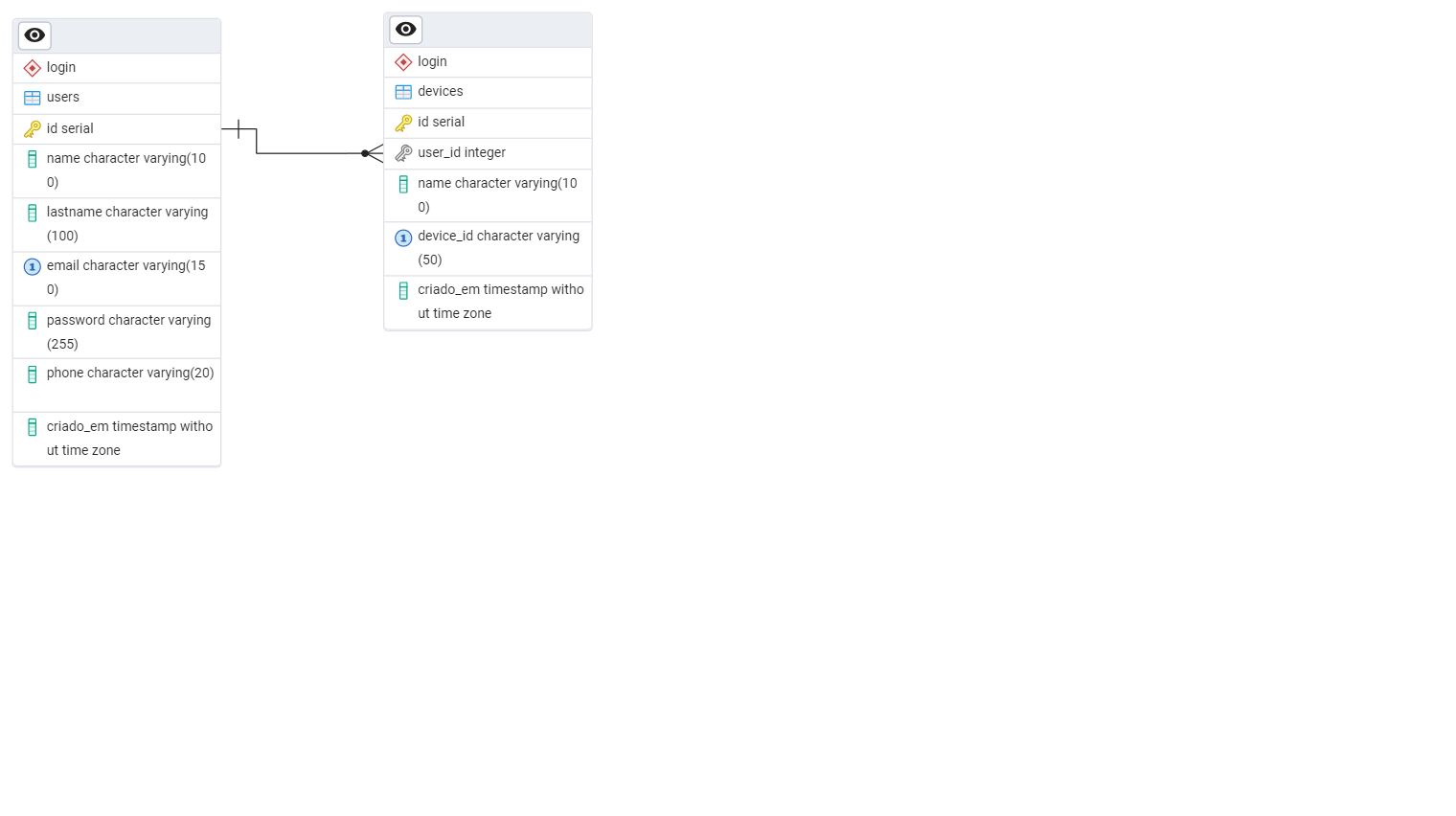
* Linguagem: TypeScript, C++
* Framework: React Native com Expo
* Banco de Dados: PostgreSQL
* Back-end: Node.js

**4.2. Camadas da Arquitetura**

* Apresentação (Front-end): Interface com o usuário
* Serviço: Requisições HTTP para o back-end
* Negócio: Validações e regras de negócio locais
* Persistência: Armazenamento local e sincronização com servidor

**5. Modelagem de Dados**

**5.1. Diagrama de Entidade-Relacionamento (DER)**

****

**5.2. Estrutura de Tabelas (exemplo)**

**Tabela: users**

* id: serial, chave primária
* name: texto
* lastname: texto
* email: texto, único
* password: texto
* phone: texto
* criado\_em: timestamp without timezone

**Tabela: devices**

* id: serial, chave primária
* user\_id: inteiro
* name: texto
* device\_id: texto, único
* criado\_em: timestamp without timezone

**6. Fluxos de Navegação**

**6.1. Telas e Componentes**

* Tela de Login
* Tela de Cadastro
* Tela Inicial (acesso para demais telas)
* Tela Meus Dispositivos
* Tela de Mapa
* Tela de Dashboard
* Tela de Sobre
* Tela de Minha Conta
* Tela de Relatórios

**6.2. Diagrama de Navegação**

1. O usuário inicia na Tela de Login

2. Pode navegar para a Tela de Cadastro e, após o sucesso, retorna ao Login.

3. Após o login bem-sucedido, o usuário é direcionado para a Tela Inicial (Dashboard).

4. Acessar aplicativo

5. A partir da Tela Inicial, uma barra de navegação permite o acesso às demais Telas.

**7. Segurança**

**7.1. Autenticação e Autorização**

* Uso de JWT (JSON Web Token)
* Senhas criptografadas com bcrypt
* Verificação de sessão ativa

**7.2. Armazenamento Seguro**

* O token JWT é armazenado de forma segura no dispositivo cliente (utilizando, por exemplo, AsyncStorage em React Native).
* Toda a comunicação com o servidor ocorre sobre o protocolo HTTPS para garantir a criptografia dos dados em trânsito.

**8. Testes**

**8.1. Tipos de Testes Aplicados**

* Testes Manuais de Funcionalidade: Foram realizados ao final de cada Sprint para validar as User Stories e garantir que os requisitos foram atendidos.
* Testes de Usabilidade: Verificação da interface e da experiência do usuário em diferentes tamanhos de tela (mobile e desktop).
* Testes de Integração (Manuais): Validação da comunicação ponta a ponta, desde o envio de dados pelo dispositivo IoT até sua exibição no front-end.

**8.2. Ferramentas de Teste (Sugeridas)**

* Jest / React Native Testing Library: Para implementação futura de testes unitários e de componentes no front-end.
* Postman / Thunder Client: Utilizados durante o desenvolvimento para testar os endpoints da API.

**9. Implantação**

**9.1. Ambientes**

* Desenvolvimento: Ambiente local nas máquinas dos desenvolvedores.
* Produção: Backend e Banco de Dados hospedados em um provedor de nuvem (Railway)

**9.2. Publicação**

* Android: Geração de um arquivo .APK para distribuição e instalação direta.
* Web: Publicação do front-end web em uma plataforma de hospedagem (Railway).

**10. Manutenção e Atualizações**

O plano de manutenção contínua prevê:

* Atualizações de segurança periódicas das bibliotecas e frameworks.
* Correções de bugs reportados pelos usuários ou identificados através de logs.
* Inclusão de novas funcionalidades conforme a evolução do produto e a demanda

**11. Anexos**

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.