Arquitetura front-end

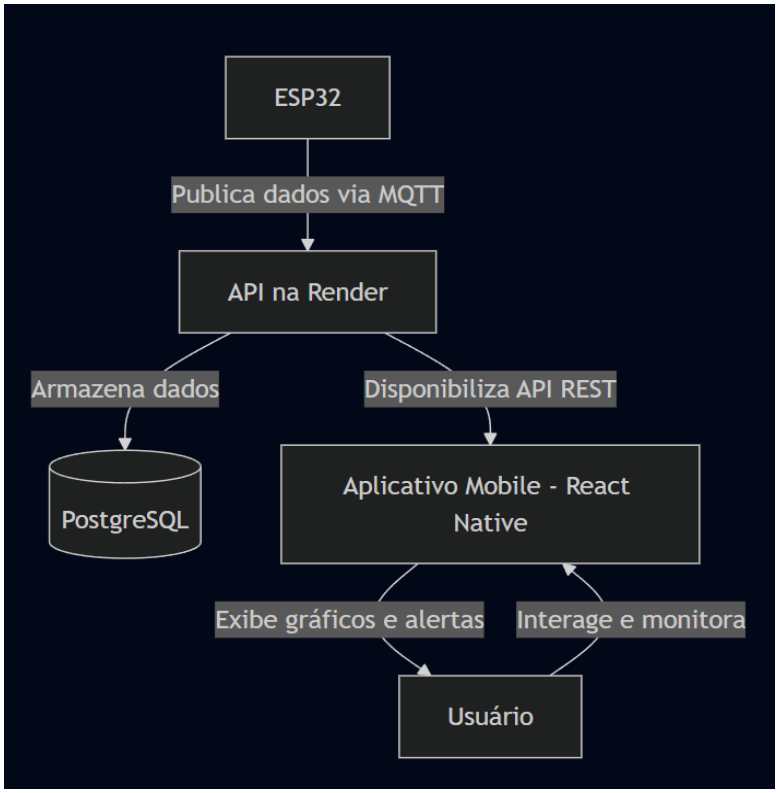
1. Introdução

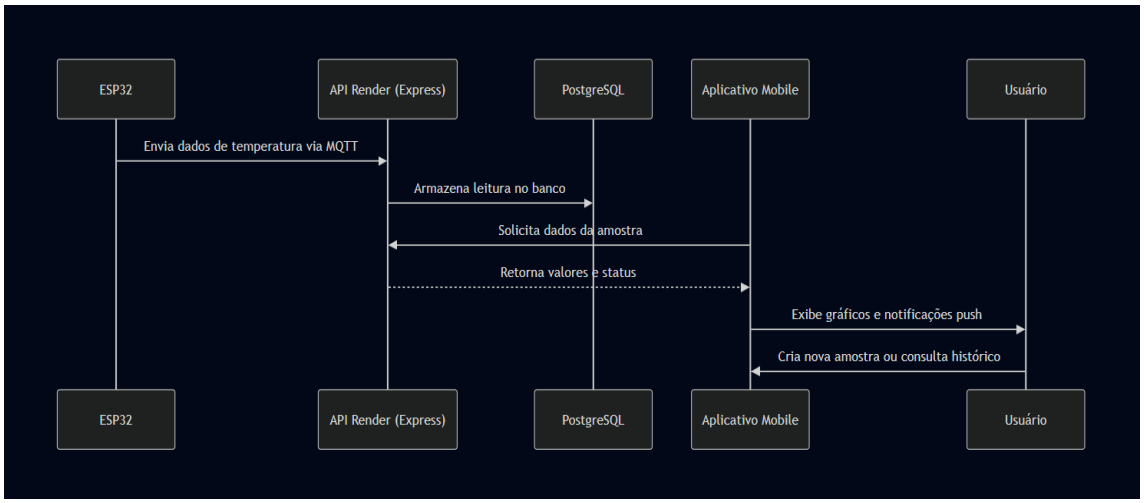
Este documento descreve a arquitetura, os componentes e o fluxo de dados do sistema do **Projeto ABP (ConnectLab)**, desenvolvido pelo grupo HighTech com **React (frontend)**, **Node.js/Express (backend)** e **ESP32 (dispositivo IoT)**, utilizando comunicação via **MQTT**.
O app foi desenvolvido em **React Native + Expo**, integrado a um backend hospedado na **Render**, que se comunica com um **ESP32** responsável por coletar dados de temperatura em tempo real.
O objetivo é apresentar uma visão técnica clara e direta do funcionamento do sistema e das interações entre seus módulos.

2. Arquitetura Geral

O sistema é composto por três camadas principais:

- **Dispositivo IoT (ESP32)**
  Responsável pela coleta e envio dos dados de sensores via protocolo MQTT.
- **API na Render:** recebe e processa os dados do ESP32, armazenando-os no banco **PostgreSQL**.
- **Backend (Node.js + Express + PostgreSQL)**
  Processa os dados recebidos, armazena no banco de dados e fornece uma API REST para o frontend.
- **Frontend (React + Styled-components + Vite)**
  Interface gráfica que consome os dados do backend e apresenta as informações ao usuário, exibe gráficos, histórico e envia notificações push.





3. Estrutura do Projeto

Frontend (/frontend)

- **frontend/src/app.tsx** — Ponto de entrada principal do Expo.
- **frontend/src/screens/** — Telas (Login, Cadastro, Dashboard, Amostras, Gráficos, Histórico, Notificações, Conta).
- **frontend/src/components/** — Componentes reutilizáveis (botões, cards, gráficos).
- **frontend/src/services/api.ts** — Comunicação HTTP com a API na Render.
- **frontend/src/hooks/** — Hooks personalizados (autenticação, controle MQTT).
- **frontend/src/context/** — Contextos globais (usuário, sessão, dados).

4. Principais Telas e Funcionalidades

- **Login / Cadastro** – Autenticação de usuário e criação de conta.
- Tela Inicial (Dashboard) – Mostra temperatura atual e status do forno.
- **Amostra** – Criação e acompanhamento de uma amostra em execução.
- **Gráficos** – Exibição de dados em barras e linhas, com biblioteca de gráficos integrada.
- **Histórico** – Exibe registros antigos armazenados no PostgreSQL.
- **Notificações** – Exibe alertas via push notification em caso de temperatura fora do parâmetro.
- **Minha Conta** – Informações pessoais e opções de logout.

5. Banco de Dados (MongoDB)

- **ESP32 → API Render:** via MQTT.
- **API Render → PostgreSQL:** persistência dos dados de sensores.
- **Aplicativo → API Render:** via requisições HTTP (Axios/Fetch).
- **Notificações Push:** geradas pelo backend e entregues ao app via serviço Expo Notifications.

6. Fluxo de Autenticação

1. Usuário insere login e senha no frontend.
2. O backend valida credenciais e gera um **token JWT**.
3. O token é armazenado no localStorage e enviado em cada requisição.
4. O middleware no backend verifica o token antes de permitir acesso às rotas protegidas.

7. Tecnologias Utilizadas

- **Frontend:** React Native, Expo, TypeScript, Styled-components, Axios, MQTT.js.
- **Backend:** Node.js, Express, PostgreSQL, Sequelize (ORM), MQTT Broker.
- **Nuvem:** Render (hospedagem da API e banco).
- **IoT:** ESP32 com sensor de temperatura.

8. Conclusão

O sistema integra hardware e software de forma escalável, permitindo monitoramento em tempo real de dados de temperatura de forno de laboratório.
A arquitetura foi projetada para suportar futuras expansões e novas funcionalidades sem comprometer desempenho ou segurança.