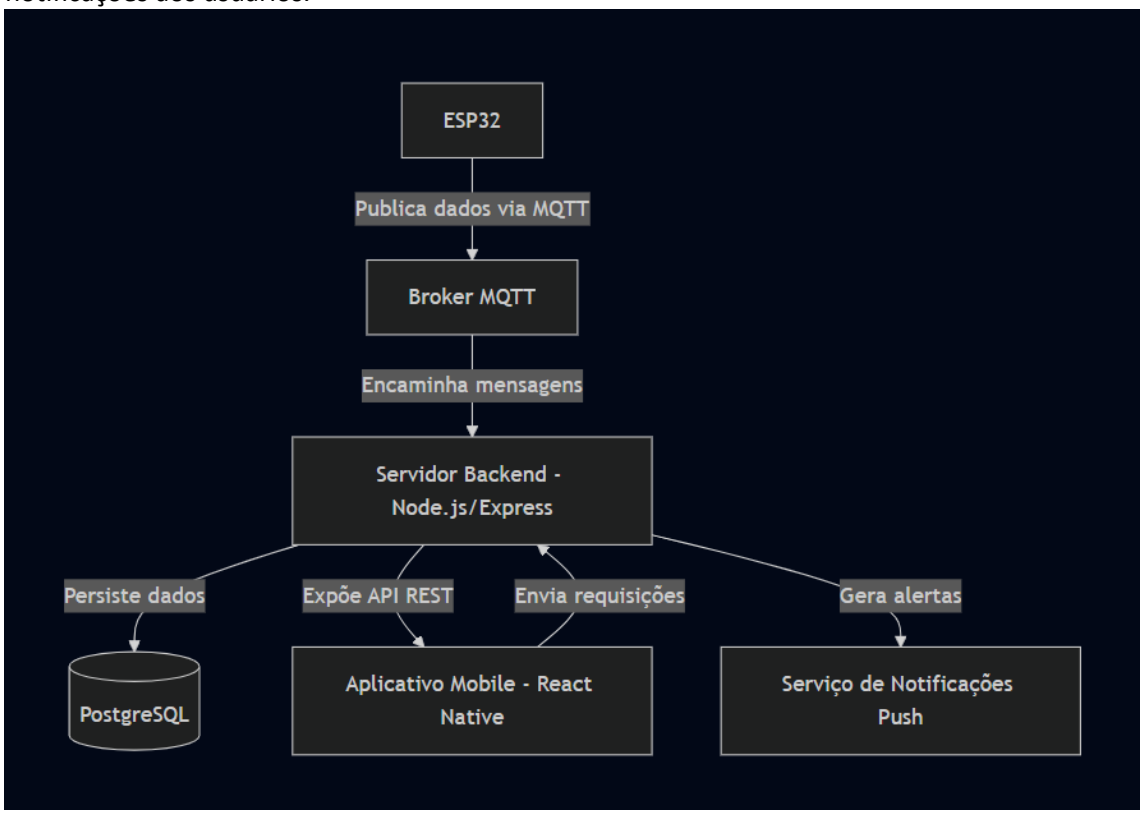
Arquitetura back-end

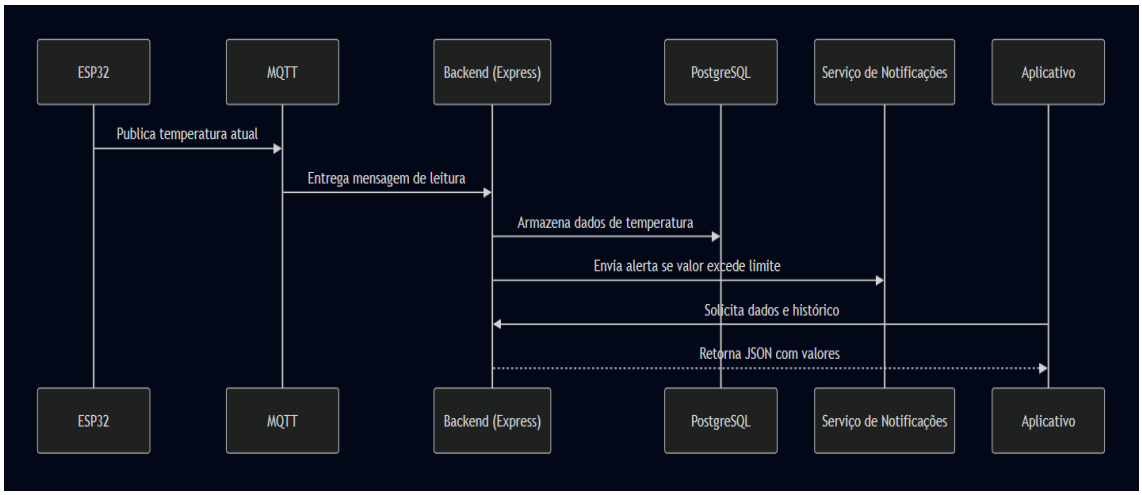
1. Introdução

Este documento descreve a arquitetura e o funcionamento do **backend** do sistema de **Projeto ABP (ConnectLab)**, desenvolvido pelo grupo HighTech.
O backend é responsável por receber os dados enviados pelo **ESP32** via **MQTT**, armazená-los no **PostgreSQL**, processar alertas e disponibilizar endpoints REST consumidos pelo aplicativo mobile.

2. Visão Geral da Arquitetura

O backend está desenvolvido em **Node.js + Express**, escrito em **TypeScript**, e implantado na plataforma **render**.
A aplicação segue um padrão **MVC (Model-View-Controller)** e utiliza o **Sequelize** como ORM para acesso ao banco PostgreSQL.
Também faz uso de **MQTT.js** para a comunicação com o ESP32 e **Expo Push API** para envio de notificações aos usuários.





3. Estrutura do Projeto

src/index.ts — Ponto de entrada principal do servidor Express.
src/config/ — Configurações (banco, MQTT, variáveis de ambiente).
src/models/ — Definição das entidades Sequelize (User, Sample, Reading, Alert).
src/controllers/ — Lógica de controle das rotas.
src/routes/ — Definição dos endpoints REST.
src/services/ — Integrações (MQTT, notificações, processamento de alertas).
src/middleware/ — Autenticação JWT e tratamento de erros.

4. Principais Funcionalidades

Recepção de Dados MQTT: o backend assina tópicos MQTT, recebendo dados de temperatura do ESP32.
Persistência: os dados são validados e armazenados no PostgreSQL via Sequelize.
API REST: endpoints permitem que o aplicativo acesse dados, histórico e status das amostras.
Geração de Alertas: valores fora da faixa disparam alertas registrados e enviados via push.
Notificações Push: integradas com o serviço Expo para envio em tempo real ao usuário.
Autenticação: JWT protege rotas sensíveis, garantindo acesso seguro.

5. Endpoints Principais

POST /auth/login – autenticação de usuário.
POST /auth/register – cadastro de novo usuário.
GET /samples – lista amostras em execução.
POST /samples – cria nova amostra.
GET /readings/:id – obtém leituras da amostra.
GET /alerts – retorna alertas registrados.
POST /notify – envia notificação push.

6. Banco de Dados

Banco: PostgreSQL (Render)
ORM: Sequelize
Entidades Principais:
User (id, nome, email, senha)
Sample (id, nome, status, data_início, data_fim)
Reading (id, id_amostra, temperatura, timestamp)
Alert (id, id_amostra, tipo, mensagem, timestamp)

7. Comunicação MQTT

Tópico de publicação: /lab/temperatura
Cliente: ESP32 publica leituras.
Assinante: Backend consome via mqtt.on('message', callback).
Broker: hospedado na nuvem (ex: Mosquitto).

8. Notificações Push

O backend mantém tokens de dispositivos Expo registrados pelos usuários.
Ao detectar uma temperatura fora do parâmetro, o servidor envia requisições à **Expo Push API**, entregando alertas diretamente no app.
11. Tecnologias Utilizadas
Linguagem: TypeScript
Framework: Express
Banco: PostgreSQL
ORM: Sequelize
Mensageria: MQTT.js
Autenticação: JSON Web Token (JWT)
Push: Expo Notifications API
Hospedagem: Render

9. Segurança e Boas Práticas

Criptografia de senhas com bcrypt.
JWT expira em tempo configurável.
Validação de entrada com express-validator.
Uso de variáveis de ambiente (.env).
Logs com Winston/Morgan.

10. Conclusão

O backend integra de forma robusta o dispositivo ESP32, o banco PostgreSQL e o aplicativo móvel, garantindo comunicação confiável e escalável.
Sua estrutura modular e aderência a boas práticas de desenvolvimento permitem fácil manutenção e expansão futura.