Sistema IoT Sensor Monitor

Sumário

Visão Geral

O Sistema IoT Sensor Monitor agrega leituras de sensores pneumáticos e fornece visualização amigável para acompanhamento em tempo real. O projeto combina um backend RESTful seguro com uma aplicação mobile construída em React Native (Expo), entregando dashboards, listagens e detalhes das leituras por sensor.

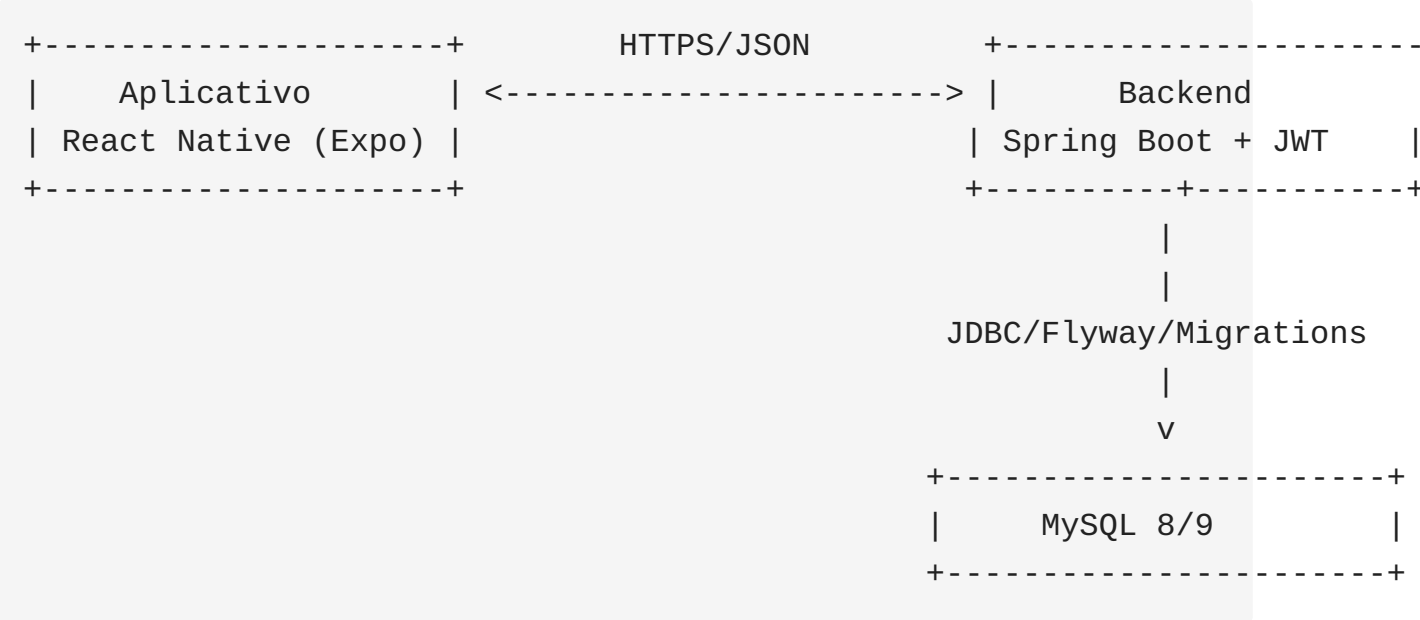
Descrição do Produto

- **Objetivo**: Monitorar leituras de sensores IoT, oferecendo dashboards consolidados, histórico de leituras e ferramentas administrativas.
- **Público-alvo**: Equipes de manutenção industrial, operadores de linhas de produção e gestores que precisam visibilidade imediata do estado dos sensores.
- **Principais funcionalidades**:

- Cadastro e autenticação de usuários com JWT.
- Registro e consulta de leituras via API REST.
- Dashboard com estatísticas (médias, máximos, mínimos) e gráficos por sensor.
- Lista de sensores com últimas leituras e navegação para detalhes.
- Ajuste dinâmico da URL da API dentro do app mobile.

Arquitetura da Solução

Visão Macro



Componentes do Backend

- **Framework**: Spring Boot 3.5.
- **Persistência**: Spring Data JPA, MySQL Connector/J.
- **Migrações**: Flyway (scripts `V1__create-reading-entity.sql` e `V2__create-users-table.sql`).
- **Segurança**: Spring Security com JWT (biblioteca JJWT 0.11.5).
- **Principais pacotes**:
- `controllers`: `AuthController`, `ReadingController`.
- `services`: `AuthService`, `JwtService`, `ReadingService`.
- `models`: `User`, `Reading` e DTOs.
- `config`: `SecurityConfig`, `JwtAuthenticationFilter`.

Componentes do Frontend

- **Framework**: React Native 0.79 com Expo 53.
- **Gerenciamento de estado**: Hooks e contexto de notificações.
- **Navegação**: React Navigation (stack navigator).
- **Serviços**: `apiService` com Axios, interceptores JWT e teste de conectividade.
- **Telas**: Login, Dashboard, Lista de Sensores, Detalhes do Sensor, Configurações.
- **Recursos adicionais**: Gráficos com `react-native-chart-kit`, armazenamento seguro com AsyncStorage.

Fluxo de Dados

1. Usuário realiza login/cadastro -> backend retorna token JWT.
2. Token é persistido em AsyncStorage e anexado às requisições.
3. API expõe endpoints sob `/api` para leitura/escrita de dados.
4. Leituras são armazenadas em MySQL; agregações são calculadas no frontend.
5. Notificações de sucesso/erro surgem via `NotificationContext` para feedback imediato.

Infraestrutura e Configuração

Banco de Dados

- MySQL (>= 8.0) com banco `api_readings`.
- Usuário padrão: `admin` / `root` (alterável via variáveis `SPRING_DATASOURCE_*`).
- Flyway cria as tabelas de `readings` e `users` na inicialização.

Backend

- Executar com o wrapper Maven: `./mvnw spring-boot:run`.
- Endpoints principais:
- `POST /api/auth/register`
- `POST /api/auth/login`
- `GET /api/readings`
- `POST /api/readings`

Frontend

- Instalar dependências: `npm install`.
- Iniciar bundler: `npm start`.
- Definir URL da API em Configurações do app, conforme ambiente (ex.: `http://localhost:8080/api`).

Segurança e Autenticação

- Tokens JWT assinam payload com segredo configurável.
- Filtro `JwtAuthenticationFilter` valida token e preenche contexto de segurança.
- Senhas armazenadas com hashing BCrypt via `PasswordEncoder` do Spring Security.
- CORS configurado para permitir chamadas do app mobile.

Cronograma do Projeto

Fase	Período	Entregas principais
Planejamento	01/09/2025 - 10/09/2025	Levantamento de requisitos, definição da arquitetura macro.
Configuração de Infra	11/09/2025 - 20/09/2025	Setup do MySQL, projeto Spring Boot inicial, configuração do Expo.
Desenvolvimento Backend	21/09/2025 - 10/10/2025	Migrations Flyway, endpoints REST, autenticação JWT, testes com cURL.
Desenvolvimento Frontend	11/10/2025 - 25/10/2025	Telas de Login, Dashboard, Lista e Detalhes, integração com API.
Integração e QA	26/10/2025 - 30/10/2025	Ajustes de CORS, notificações globais, testes end-to-end.
Entrega Final	31/10/2025	Documentação, publicação no repositório e validação com stakeholders.

Equipe

- **Caio Caram de Souza** – RM 552248
- **Isabella Ventura Diaz** – RM 551793
- **Lucas Gabriel Gianini Moreira** – RM 99921
- **Maria Eduarda de Carvalho Goda** – RM 552276
- **Maria Eloisa da Silva Santos** – RM 552294

Próximos Passos

1. Implementar alertas proativos com base em thresholds definidos por sensor.
2. Adicionar testes automatizados (JUnit + React Testing Library).
3. Disponibilizar build de produção e guia de deploy em ambiente cloud.