■ DISTRISCHOOL – DOCUMENTAÇÃO DE PRÉ-REQUISITOS

O projeto **DistriSchool** é uma aplicação de **gestão escolar distribuída**, desenvolvida com uma arquitetura moderna baseada em **microsserviços**. Utiliza tecnologias como **Spring Boot**, **Apache Kafka**, **React + Vite**, **PostgreSQL** e **Docker**, garantindo alta escalabilidade, comunicação assíncrona e interface moderna.

■ REQUISITOS DE AMBIENTE

Antes de executar o projeto, é necessário garantir que o ambiente contenha as seguintes dependências instaladas:

1. Java JDK 17 ou superior – execução do backend com Spring Boot.
2. Node.js 18.x ou superior – execução do frontend React + Vite.
3. npm ou yarn – gerenciamento de pacotes do frontend.
4. Docker e Docker Compose – criação e orquestração de containers.
5. PostgreSQL 14 ou superior – banco de dados relacional principal.
6. Apache Kafka 3.x – mensageria entre microsserviços.
7. Flyway – controle e versionamento de migrações do banco de dados.

■■ ESTRUTURA DO SISTEMA

O sistema é dividido em múltiplos serviços independentes, responsáveis por diferentes domínios da aplicação: - **school-core-service** → Gestão de alunos e turmas (Spring Boot, Spring Data JPA) - **notification-service** → Envio de notificações e mensagens (Spring Boot, Spring Kafka) - **auth-service** → Autenticação e autorização (Spring Security, OAuth2, JWT) - **frontend** → Interface do usuário em React + Vite - **broker (Kafka)** → Comunicação assíncrona entre serviços - **database (PostgreSQL)** → Armazenamento relacional com versionamento Flyway

■ REQUISITOS FUNCIONAIS

- **Autenticação e Autorização:** cadastro/login de usuários (alunos, professores, admins, pais), recuperação de senha, autenticação JWT, perfis de acesso e verificação por e-mail. - **Gestão de Alunos:** cadastro, edição e exclusão de alunos; busca por matrícula/nome; upload de documentos; histórico acadêmico; exportação em PDF/CSV. - **Gestão de Professores:** cadastro de professores; atribuição de disciplinas e turmas; integração com Google Calendar; relatórios de desempenho. - **Gestão de Turmas e Horários:** criação de turmas, detecção de conflitos, importação via Excel, suporte a múltiplos turnos e escolas. - **Registro de Presenças:** chamada diária, relatórios de faltas, histórico anual, notificações automáticas e suporte a QR code/biometria. - **Gestão de Notas e Avaliações:** registro de notas e geração de relatórios de desempenho.

■ REQUISITOS NÃO FUNCIONAIS

- **Desempenho:** tempo de resposta < 2 segundos para login e consultas básicas. - **Segurança:** criptografia de senhas (BCrypt), autenticação via JWT, conformidade com LGPD e backup diário. - **Escalabilidade:** arquitetura distribuída com microsserviços e Kafka. - **Disponibilidade:** uptime mínimo de 99%. - **Usabilidade:** interface responsiva e multilíngue (PT-BR / EN). - **Integração:** APIs REST e eventos Kafka (`student.created`, `user.logged`, `teacher.assigned`, `schedule.updated`, `attendance.recorded`).

■ EXECUÇÃO DO PROJETO

Backend

```
cd backend/school-core-service
mvn spring-boot:run
```

Frontend

```
cd frontend
npm install
npm run dev
```

Docker Compose

```
docker-compose up --build
```

■ ESTRUTURA DE CONTAINERS (DOCKER)

- **frontend** → React + Vite - **auth-service** → Spring Boot (JWT/OAuth2) - **school-core-service** → Spring Boot (alunos/turmas) - **notification-service** → Spring Boot (mensageria) - **kafka / zookeeper** → Comunicação entre microsserviços - **postgres** → Banco de dados relacional - **flyway** → Migração e versionamento de schema

■ OBSERVAÇÕES FINAIS

Todos os serviços se comunicam via **Kafka** utilizando tópicos como `student.created` e `notification.sent`. O **React + Vite** garante um fluxo de desenvolvimento rápido e reativo. O projeto segue o padrão **RESTful**, podendo ser implantado em **ambientes cloud-native** como Kubernetes, AWS ou Azure.

■ *Fonte: Documento de Requisitos do Projeto Gestão Escolar e Repositório oficial – github.com/unifor-online/districschool*