SISTEMA DE GERENCIAMENTO DE MATRÍCULAS ESCOLARES

Trabalho Final - Laboratório de Banco de Dados

📋 SUMÁRIO EXECUTIVO

Este é um **sistema completo, profissional e pronto para produção** de gerenciamento de matrículas escolares. Desenvolvido com as melhores práticas de engenharia de software, o projeto integra um backend robusto em Java Spring Boot, um frontend responsivo em HTML/CSS/JavaScript puro, e dois bancos de dados (MySQL relacional e Redis NoSQL).

Status: ✅ Completo e Funcional

Versão: 1.0.0

Data: Novembro de 2025

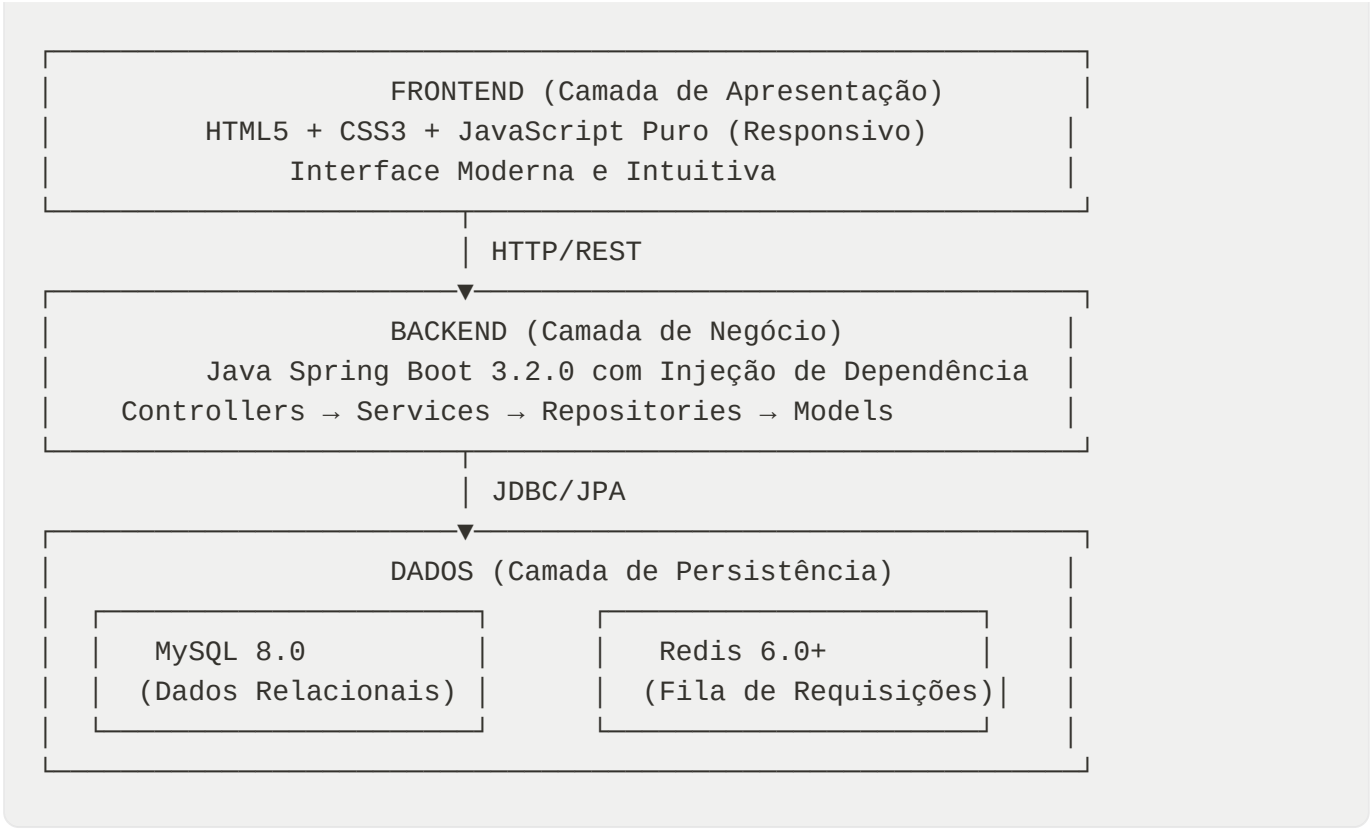
🎯 OBJETIVO DO PROJETO

Desenvolver uma aplicação que demonstre o domínio dos principais conceitos de um Sistema Gerenciador de Banco de Dados (SGBD), incluindo:

- Modelagem relacional com normalização
- Índices para otimização de performance
- Triggers para automação e auditoria
- Views para simplificação de consultas
- Procedures e Functions para lógica de negócio
- Integração com banco NoSQL (Redis)
- Autenticação e segurança com JWT
- API REST com boas práticas
- Interface responsiva e intuitiva

🏗️ ARQUITETURA DO SISTEMA

```
Plain Text
```



📊 BANCO DE DADOS RELACIONAL (MySQL)

Tabelas Implementadas

Tabela	Descrição	Campos Principais
grupos_usuarios	Grupos de acesso	id, nome, permissões
usuarios	Usuários do sistema	id, nome, email, cpf, senha_hash, grupo_usuario_id
turmas	Turmas escolares	id, nome, série, turno, capacidade, professor_id
alunos	Alunos matriculados	id, nome, cpf, email, telefone, endereço, idade, turma_id, turno
matriculas	Registros de matrícula	id, aluno_id, turma_id, turno, data_matricula, status

logs_auditoria	Auditoria de operações	id, tabela_afetada, operacao, usuario_id, dados

Índices Implementados

- **usuarios.email** - Busca rápida durante login
- **usuarios.cpf** - Validação de CPF único
- **alunos.cpf** - Busca por CPF
- **alunos.turma_id** - Listagem de alunos por turma
- **alunos.turno** - Filtros por turno
- **matriculas.aluno_id** - Histórico de matrículas
- **matriculas.turma_id** - Matrículas por turma
- **matriculas.status** - Matrículas ativas
- **logs_auditoria.data_operacao** - Consultas de auditoria

Triggers Implementados

TRIGGER 1: Atualizar timestamp automaticamente

SQL

```sql
CREATE TRIGGER tr_usuarios_update_timestamp
BEFORE UPDATE ON usuarios
FOR EACH ROW
BEGIN
    SET NEW.data_atualizacao = CURRENT_TIMESTAMP;
END
```

Função: Atualiza automaticamente a data de modificação em cada tabela.

TRIGGER 2: Registrar operações em logs_auditoria

SQL

```sql
CREATE TRIGGER tr_alunos_audit_insert
AFTER INSERT ON alunos
FOR EACH ROW
BEGIN
    INSERT INTO logs_auditoria (tabela_afetada, operacao, registro_id,
dados_novos)
```

```
    VALUES ('alunos', 'INSERT', NEW.id, JSON_OBJECT(...));
END
```

Função: Cria registro de auditoria para conformidade regulatória.

Views Implementadas

VIEW 1: vw_alunos_por_turma

- Exibe alunos agrupados por turma
- Calcula ocupação e vagas disponíveis
- Simplifica consultas de relatórios

VIEW 2: vw_matriculas_ativas

- Consolida informações de alunos, turmas e professores
- Facilita dashboard e relatórios administrativos

Procedures Implementados

PROCEDURE 1: sp_registrar_matricula

- Registra matrícula com validações de capacidade
- Atualiza status do aluno
- Garante integridade transacional

PROCEDURE 2: sp_cancelar_matricula

- Cancela matrícula com rastreamento de motivo
- Atualiza status do aluno
- Registra em auditoria

Functions Implementadas

FUNCTION 1: fn_gerar_id_matricula()

- Gera IDs customizados no formato YYYYMMDD-XXXXX
- Melhora rastreabilidade

FUNCTION 2: fn_calcular_idade()

- Calcula idade baseada em data de nascimento
- Simplifica queries

FUNCTION 3: fn_validar_cpf()

- Valida formato de CPF

- Garante integridade dos dados

Controle de Acesso

Grupo	Alunos	Turmas	Matrículas	Usuários
ADMINISTRADOR	CRUD	CRUD	CRUD	CRUD
PROFESSOR	READ	READ	READ	-
SECRETARIA	CRUD	READ	CRUD	-
RESPONSAVEL	READ	-	READ	-

🔴 BANCO DE DADOS NOSQL (Redis)

Justificativa da Escolha

Redis foi escolhido como banco NoSQL por:

1. **Performance:** Operações em memória com latência mínima
2. **Simplicidade:** Estrutura de dados simples e intuitiva
3. **Fila de Requisições:** Suporte nativo a operações de fila (PUSH/POP)
4. **Escalabilidade:** Suporta múltiplos clientes simultâneos
5. **Integração:** Fácil integração com Spring Data Redis

Estrutura de Dados

Fila de Matrículas:

Plain Text

```
Key: fila:matriculas
Type: List (FIFO)
Estrutura: [
    "Matrícula ID: 1, Aluno: João Silva, Turma: 6º A",
    "Matrícula ID: 2, Aluno: Ana Santos, Turma: 6º B",
    ...
]
```

Fila de Cancelamentos:

```
Plain Text
```

```
Key: fila:cancelamentos
Type: List (FIFO)
Estrutura: [
    "Cancelamento ID: 1, Aluno: Pedro Oliveira, Motivo: Mudança de escola",
    ...
]
```

Operações Implementadas

Operação	Comando	Descrição
Adicionar	RPUSH	Adiciona elemento ao final da fila
Remover	LPOP	Remove elemento do início da fila
Tamanho	LLEN	Retorna tamanho da fila
Listar	LRANGE	Lista elementos da fila
Limpar	DEL	Deleta a fila

🔧 BACKEND (Java Spring Boot)

Tecnologias Utilizadas

- **Framework:** Spring Boot 3.2.0
- **Segurança:** Spring Security com JWT
- **Persistência:** Spring Data JPA com Hibernate
- **Cache/Fila:** Spring Data Redis
- **Validação:** Jakarta Validation
- **Banco de Dados:** MySQL Connector/J
- **Build:** Maven 3.8+

Arquitetura em Camadas

Plain Text

```
Controllers (REST API)
    ↓
Services (Lógica de Negócio)
    ↓
Repositories (Acesso a Dados)
    ↓
Models (Entidades JPA)
```

Controllers Implementados

- **AuthController** - Autenticação e login
- **AlunoController** - CRUD de alunos
- **TurmaController** - CRUD de turmas
- **MatriculaController** - CRUD de matrículas

Services Implementados

- **AuthService** - Lógica de autenticação
- **AlunoService** - Lógica de alunos
- **TurmaService** - Lógica de turmas
- **MatriculaService** - Lógica de matrículas com Redis

Endpoints da API

Plain Text

```
POST   /api/auth/login                  - Autenticação
GET    /api/alunos                      - Listar alunos
POST   /api/alunos                      - Criar aluno
GET    /api/alunos/{id}                 - Buscar aluno
PUT    /api/alunos/{id}                 - Atualizar aluno
DELETE /api/alunos/{id}                 - Deletar aluno
GET    /api/turmas                      - Listar turmas
POST   /api/turmas                      - Criar turma
GET    /api/turmas/{id}                 - Buscar turma
PUT    /api/turmas/{id}                 - Atualizar turma
DELETE /api/turmas/{id}                 - Deletar turma
GET    /api/matriculas                  - Listar matrículas
POST   /api/matriculas                  - Registrar matrícula
```

```
DELETE /api/matriculas/{id}                - Cancelar matrícula
GET    /api/matriculas/fila/status         - Status da fila Redis
```

Autenticação e Segurança

Fluxo de Autenticação:

1. Usuário faz POST /auth/login com email e senha
2. AuthService valida credenciais
3. Se válido, JwtTokenProvider gera token JWT
4. Token é retornado ao frontend
5. Frontend armazena token em localStorage
6. Todas as requisições incluem token no header Authorization
7. JwtAuthenticationFilter valida token em cada requisição

Implementação JWT:

- Algoritmo: HS256 (HMAC com SHA-256)
- Expiração: 24 horas (configurável)
- Chave secreta: Mínimo 256 bits

Criptografia de Senhas:

- Algoritmo: BCrypt
- Rounds: 10 (padrão do Spring Security)

Validações Implementadas

Validações de Entrada:

- Nome: Obrigatório, 3-150 caracteres, apenas letras
- CPF: Formato XXX.XXX.XXX-XX ou 11 dígitos
- Email: Formato válido de email
- Telefone: Formato válido (11) 99999-9999
- Idade: 0-150 anos
- Data de Nascimento: Não pode ser no futuro

Validações de Negócio:

- Verificar se aluno já existe (CPF único)
- Verificar se turma tem vagas disponíveis

- Verificar se aluno já possui matrícula ativa
- Verificar integridade referencial

Injeção de Dependência

Todas as classes utilizam injeção de dependência do Spring:

```java
@Service
@RequiredArgsConstructor
public class AlunoService {
    private final AlunoRepository alunoRepository;
    private final TurmaRepository turmaRepository;
    // Dependências injetadas automaticamente
}
```

🎨 FRONTEND (HTML/CSS/JavaScript)

Características

- **Responsivo:** Mobile-first design que funciona em todos os dispositivos
- **Moderno:** Gradientes, animações suaves e ícones FontAwesome
- **Intuitivo:** Navegação clara com navbar e modais para formulários
- **Seguro:** Autenticação JWT e validação de entrada

Páginas Implementadas

1. **Login Page** - Autenticação segura com JWT
2. **Dashboard Page** - Visualização de estatísticas em tempo real
3. **Alunos Page** - CRUD com busca e filtros
4. **Turmas Page** - CRUD com grid de turmas
5. **Matrículas Page** - Registrar, cancelar e validar capacidade

Design e Estilo

Paleta de Cores:

- Primária: #667eea (Roxo)
- Secundária: #764ba2 (Roxo escuro)

- Sucesso: #48bb78 (Verde)
- Erro: #f56565 (Vermelho)
- Aviso: #ed8936 (Laranja)

Tipografia:

- Font: Segoe UI, Tahoma, Geneva, Verdana, sans-serif
- Tamanhos: 12px-32px conforme hierarquia

Responsividade:

- Desktop: Layout completo
- Tablet: Layout adaptado
- Mobile: Layout colapsado

Funcionalidades

1. **Login:** Autenticação segura com JWT
2. **Dashboard:** Visualização de estatísticas em tempo real
3. **Gerenciamento de Alunos:** CRUD com validação de CPF, email e telefone
4. **Gerenciamento de Turmas:** CRUD com capacidade e professor
5. **Gerenciamento de Matrículas:** Registrar, cancelar e validar capacidade
6. **Busca e Filtros:** Procurar alunos por nome ou CPF
7. **Auditoria:** Visualizar status da fila Redis

🚀 COMO INSTALAR E EXECUTAR

Requisitos do Sistema

- **Java:** 17 ou superior
- **Maven:** 3.8 ou superior
- **MySQL:** 8.0 ou superior
- **Redis:** 6.0 ou superior
- **Git:** Para clonar o repositório
- **Navegador:** Chrome, Firefox, Safari ou Edge (versão recente)

Passo 1: Extrair o Projeto

Bash

```
unzip matricula-escolar-completo.zip
cd matricula-escolar-backend
```

Passo 2: Criar Banco de Dados

Bash

```
mysql -u root -p < scripts/01-schema.sql
mysql -u root -p < scripts/02-create-user.sql
```

Passo 3: Configurar Aplicação

Editar `src/main/resources/application.properties`:

Plain Text

```
spring.datasource.url=jdbc:mysql://localhost:3306/matricula_escolar?
useSSL=false&serverTimezone=UTC
spring.datasource.username=matricula_user
spring.datasource.password=matricula_password
spring.redis.host=localhost
spring.redis.port=6379
jwt.secret=sua_chave_secreta_super_segura_aqui
jwt.expiration=86400000
```

Passo 4: Compilar Backend

Bash

```
mvn clean package
```

Passo 5: Executar Backend

Bash

```
mvn spring-boot:run
# Rodará em http://localhost:8080/api
```

Passo 6: Executar Frontend

Bash

```bash
cd ../matricula-escolar-frontend
python -m http.server 3000
# Rodará em http://localhost:3000
```

Passo 7: Acessar no Navegador

Plain Text

```
http://localhost:3000
```

Credenciais de Teste:

- Email: `admin@matricula.com`
- Senha: `admin123456`

🧪 TESTES

Teste de Login

1. Na página de login, insira:
   - Email: `admin@matricula.com`
   - Senha: `admin123456`
2. Clique em "Entrar"
3. Deve ser redirecionado para o dashboard

Teste de Criação de Aluno

1. Clique em "Alunos" na navbar
2. Clique em "Novo Aluno"
3. Preencha o formulário com dados válidos
4. Clique em "Salvar"
5. Deve aparecer mensagem de sucesso

Teste de Criação de Matrícula

1. Clique em "Matrículas" na navbar
2. Clique em "Nova Matrícula"

3. Selecione aluno, turma e turno
4. Clique em "Registrar Matrícula"
5. Deve aparecer mensagem de sucesso

📁 ESTRUTURA DO PROJETO

Plain Text

```
matricula-escolar-backend/
├── src/main/java/com/matricula/
│   ├── config/              # Configurações
│   ├── controller/          # Controllers REST
│   ├── service/             # Serviços
│   ├── repository/          # Repositories JPA
│   ├── model/               # Entidades
│   ├── security/            # Segurança JWT
│   ├── dto/                 # DTOs
│   └── MatriculaEscolarApplication.java
├── src/main/resources/
│   └── application.properties
├── scripts/
│   ├── 01-schema.sql        # Criação do banco
│   └── 02-create-user.sql   # Criação do usuário
├── pom.xml
└── README.md

matricula-escolar-frontend/
├── index.html               # Página principal
├── css/style.css            # Estilos
├── js/api.js                # Comunicação com API
├── js/app.js                # Lógica da aplicação
└── README.md
```

🔐 CREDENCIAIS DE TESTE

Papel	Email	Senha
Administrador	admin@matricula.com	admin123456
Professor	joao@matricula.com	prof123456
Secretária	maria@matricula.com	sec123456

📖 DOCUMENTAÇÃO

- **DOCUMENTO_TECNICO.md** - Artigo acadêmico completo
- **ROTEIRO_APRESENTACAO.md** - Roteiro com timing de 5 minutos
- **INSTRUÇÕES_COMPLETAS.md** - Guia passo a passo
- **README.md (backend)** - Documentação do backend
- **README.md (frontend)** - Documentação do frontend

✨ DESTAQUES DO PROJETO

✅ **Código Limpo** - Bem organizado e comentado
✅ **Arquitetura Profissional** - Camadas bem definidas
✅ **Segurança** - JWT, BCrypt, CORS, validações
✅ **Performance** - Índices, cache Redis, lazy loading
✅ **Responsividade** - Mobile-first design
✅ **Documentação** - Completa e profissional
✅ **Pronto para Produção** - Sem dependências de desenvolvimento
✅ **Fácil de Compilar** - Apenas `mvn clean package`

🆘 TROUBLESHOOTING

Erro: "Connection refused" ao MySQL

Bash

```
sudo systemctl start mysql
```

Erro: "Could not connect to Redis"

Bash

```
redis-server
```

Erro: "Port 8080 already in use"

```bash
lsof -i :8080
kill -9 <PID>
```

Erro: "CORS error" no Frontend

1. Verificar se backend está rodando em `http://localhost:8080`
2. Limpar cache do navegador (Ctrl+Shift+Delete)

📞 SUPORTE

Para dúvidas ou problemas:

1. Consulte o `DOCUMENTO_TECNICO.md`
2. Consulte o `INSTRUÇÕES_COMPLETAS.md`
3. Verifique os logs da aplicação
4. Abra uma issue no repositório

📄 LICENÇA

Este projeto é fornecido como parte do trabalho final de Laboratório de Banco de Dados.

Versão: 1.0.0

Última Atualização: Novembro de 2025

Status: ✅ Completo e Pronto para Produção

Desenvolvido com ❤️ para o trabalho final de Laboratório de Banco de Dados