Conteúdo

[Projeto 1](#_Toc166657631)

[Título e objetivo 1](#_Toc166657632)

[Autor, data, contexto do trabalho 1](#_Toc166657633)

[Descrição do problema 1](#_Toc166657634)

[Ferramentas e versões 2](#_Toc166657635)

[Análise de dados 2](#_Toc166657636)

[Descrição de processos 3](#_Toc166657637)

[Dicionário 4](#_Toc166657638)

[String connection, instâncias, bases de dados 4](#_Toc166657639)

[Lista de tabelas, campos e tipos 4](#_Toc166657640)

[Namespaces 5](#_Toc166657641)

[Classes e métodos 5](#_Toc166657642)

[Outras informações 7](#_Toc166657643)

# Projeto

## Título e objetivo

1. Título: Gestão Educacional Colégio Colibri (Java).
2. Objetivo: Desenvolver uma API para uma escola primária com um sistema eficiente, para centralizar e otimizar o uso de dados e informações relacionados a alunos, funcionários, logins, turmas, disciplinas, notas e períodos, visando melhorar a eficiência e a otimização na parte especifica com tratamento de dados da gestão educacional.

## Autor, data, contexto do trabalho

Autor: Camila Santos, Data Início: 19/04/2024, Data Final: 20/05/2024.

O sistema visa integrar e automatizar processos acadêmicos e administrativos, melhorando a comunicação entre a instituição de ensino, seus funcionários e seus alunos, este projeto com uma futura parte de frontEnd busca promover a interatividade e a participação ativa dos alunos, através de uma plataforma (por hora ainda não desenvolvida). Este projeto é parte de um esforço maior para implementar um novo Sistema de Gestão Educacional, para melhorar o sistema operacional de uma escola.

## Descrição do problema

Temos uma escola primaria que precisa de um sistema para gerir professores, turmas, alunos, notas e períodos. Este sistema permitira a cada utilizador de acordo com o seu nível de acesso uma consulta, adição, remoção e edição.

1. O Diretor com todos os privilégios pode gerir professores, turmas, alunos e notas;
2. os professores poderão consultar turmas e alunos e adicionar, remover e editar notas;
3. e os alunos tem acesso somente a lista de alunos da sua turma e a consulta somente das suas notas.

Estes são objetivos fundamentais para a gestão eficaz de uma escola, permitindo que os funcionários, alunos e pais acedam e atualizem informações relevantes de forma segura e eficiente.

## Ferramentas e versões

IntelliJ IDEA Ultimate

Tipo de projeto: Java/Maven

Linguagem: Java

Versão: 17

Dependencias:

* Spring Boot versão

3.2.5

* Mapstruct

1.5.5

* mysql-connector-j

8.3.0

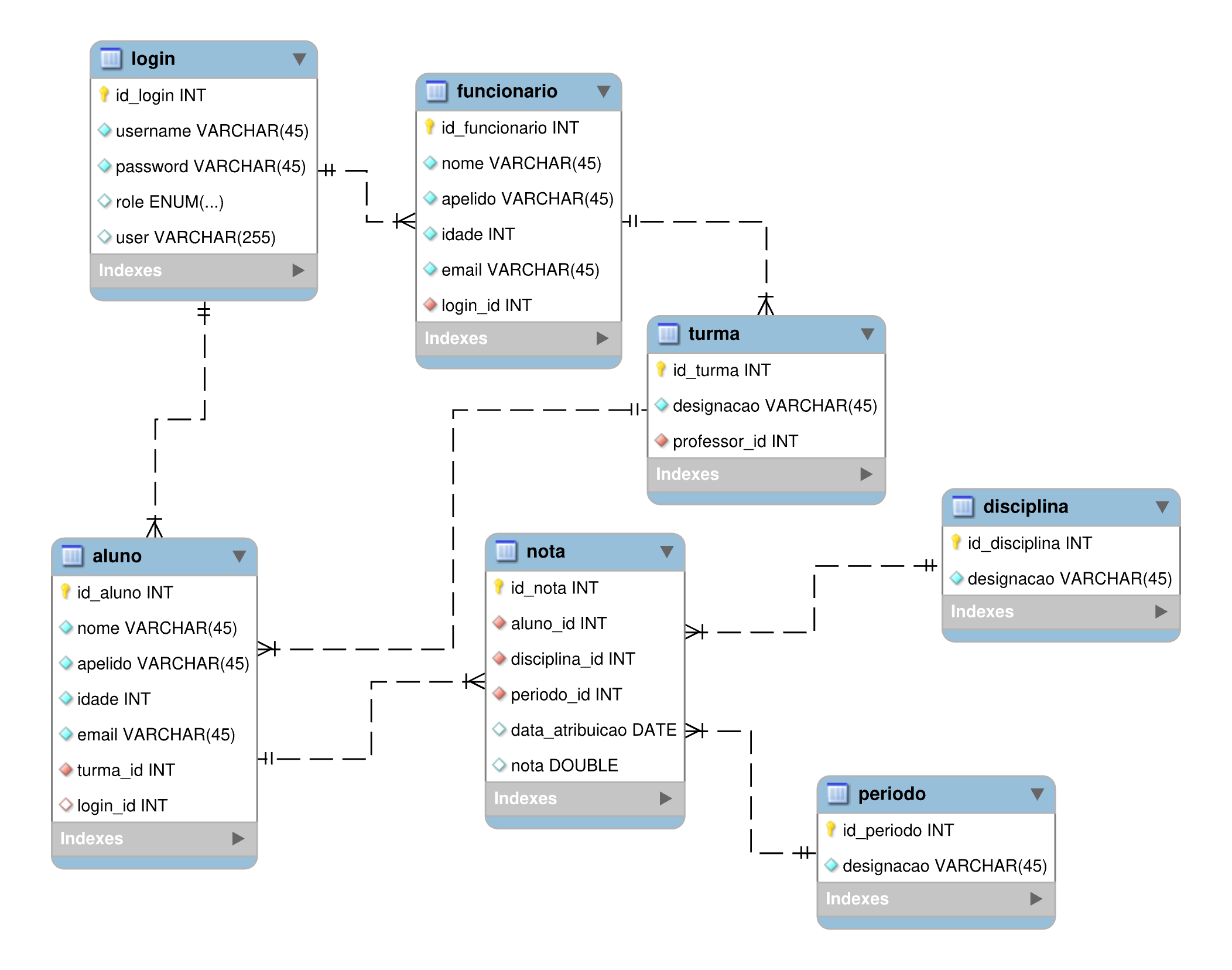
* Springdoc-openapi

2.1.0

# Análise de dados

A base de dados "bd\_projeto\_colegio\_colibri" é estruturada em torno de várias entidades, incluindo login, funcionario, turma, aluno, nota e disciplina. Cada entidade é representada por uma tabela que contém colunas específicas para armazenar informações relevantes. Por exemplo, a tabela aluno armazena detalhes sobre os alunos, como nome, apelido, idade, e-mail, e a tabela nota regista as notas atribuídas aos alunos em diferentes disciplinas.

O projeto é concebido para atender a uma ampla gama de necessidades dentro da instituição educacional, incluindo o gerenciamento de logins para acesso ao sistema, a gestão de informações sobre funcionários e turmas, e a avaliação do desempenho dos alunos através de notas e disciplinas. A estruturação da base de dados permite uma fácil integração e análise de dados, facilitando a identificação de tendências, a avaliação do desempenho dos alunos e a otimização dos recursos educacionais.



# Descrição de processos

A descrição de processos para o sistema de gestão escolar é detalhada da seguinte forma:

1. Autenticação de Usuários:
   * O processo começa com a autenticação do usuário, que é realizada através da tabela login. Os utilizadores inserem seu nome de usuário e senha, que são verificados contra os registos na tabela login. Se as credenciais correspondem, o utilizador é autenticado e tem acesso ao sistema.
2. Gestão de Funcionários:
   * Após a autenticação, os funcionários podem aceder suas informações pessoais e detalhes de trabalho, que são armazenados na tabela funcionário. Isso inclui o nome do funcionário e o ID de login associado, permitindo a personalização da experiência do utilizador.
3. Gestão de Turmas:
   * Os diretores podem gerenciar as turmas através da tabela turma, que contém informações sobre cada turma, incluindo a designação da turma e o ID do professor responsável. Isso permite a criação, atualização e exclusão de turmas, bem como a atribuição de professores a turmas específicas.
4. Gestão de Alunos:
   * Os alunos são gerenciados através da tabela aluno, que armazena informações como nome, apelido, idade, e-mail, e o ID da turma a qual o aluno pertence. Isso permite a adição, atualização e exclusão de alunos, bem como a visualização de detalhes do aluno.
5. Gestão de Notas:
   * As notas dos alunos são registadas na tabela nota, que inclui o ID do aluno, o ID da disciplina, a data de atribuição da nota e a nota em si. Isso permite a entrada, atualização e visualização de notas para cada aluno em cada disciplina.
6. Gestão de Disciplinas:
   * As disciplinas são gerenciadas através da tabela disciplina, que contém informações sobre cada disciplina, incluindo a designação da disciplina. Isso permite a criação, atualização e exclusão de disciplinas, bem como a atribuição de disciplinas a turmas.
7. Gestão de Período:
   * Os períodos são gerenciados através da tabela periodo, que contém informações sobre o Id do período e sua designação. Isso permite a criação, atualização e exclusão de períodos, bem como a atribuição de períodos as notas para melhorar a consulta das notas por períodos.

# Dicionário

## String connection, instâncias, bases de dados

* String connection:

url: jdbc:mysql://${MYSQL\_HOST}:3306/bd\_projeto\_colegio\_colibri

username: ${MYSQL\_USERNAME}//(EFACAMILA)

password: ${MYSQL\_PASSWORD}//(123.Abc##)

* Instância: “lagostin”(MySQL)
* Base de dados: “bd\_projeto\_colegio\_colibri”

## Lista de tabelas, campos e tipos

1. Entidades:
   1. login:
      1. id\_login(int)
      2. user(varchar)
      3. password(varchar)
      4. role(varchar)
   2. funcionario:
      1. id\_funcionario(int)
      2. nome(varchar)
      3. login\_id(int)
   3. turma:
      1. id\_turma(int)
      2. designação(varchar)
      3. professor\_id(int)
   4. aluno:
      1. id\_aluno(int)
      2. nome(varchar)
      3. apelido(varchar)
      4. idade(int)
      5. email(varchar)
      6. turma\_id(int)
      7. login\_id(int)
   5. nota:
      1. id\_nota(int)
      2. aluno\_id(int)
      3. disciplina\_id(int)
      4. periodo\_id(int)
      5. data\_atribuicao(datetime)
      6. nota(decimal)
   6. disciplina:
      1. id\_disciplina(int)
      2. designação(varchar)
   7. período:
      1. id\_periodo(int)
      2. designação(varchar)

## Namespaces

Projeto\_colegio\_colibri

## Classes e métodos

1. pt.colegio.colibri:
   1. Business:
      1. Core:
         1. Enuns:
            1. Roles (Enum)
         2. Aluno (Classe)
         3. Disciplina (Classe)
         4. Funcionario (Classe)
         5. Login (Classe)
         6. Nota (Classe)
         7. Periodo (Classe)
         8. Pessoa (Classe)
         9. Turma (Classe)
      2. Servisse:
         1. AlunoService (Classe)
         2. DisciplinaService (Classe)
         3. FuncionarioService (Classe)
         4. LoginService (Classe)
         5. NotaService (Classe)
         6. PeriodoService (Classe)
         7. TurmaService (Classe)
         8. UserDetailsServiceIMPL(Classe)
      3. Config:
         1. Error:
            1. RestExceptionHandler(Classe)
         2. Security:
            1. WebSecurityConfig
   2. Controller:
      1. DTOs:
         1. Resquest:
            1. AlunoResquestDTO (Classe)
            2. DisciplinaResquestDTO (Classe)
            3. FuncionarioResquestDTO (Classe)
            4. LoginResquestDTO (Classe)
            5. NotaResquestDTO (Classe)
            6. PeriodoResquestDTO (Classe)
            7. TurmaResquestDTO (Classe)
         2. Response:
            1. AlunoResponseDTO (Classe)
            2. DisciplinaResponseDTO (Classe)
            3. FuncionarioResponseDTO (Classe)
            4. LoginResponseDTO (Classe)
            5. NotaResponseDTO (Classe)
            6. PeriodoResponseDTO (Classe)
            7. TurmaResponseDTO (Classe)
      2. Mapper:
         1. AlunoControllerMapper (Interface)
         2. DisciplinaControllerMapper (Interface)
         3. FuncionarioControllerMapper (Interface)
         4. LoginControllerMapper (Interface)
         5. NotaControllerMapper (Interface)
         6. PeriodoControllerMapper (Interface)
         7. TurmaControllerMapper (Interface)
      3. AlunoController (Classe)
      4. DisciplinaController (Classe)
      5. FuncionarioController (Classe)
      6. LoginController (Classe)
      7. NotaController (Classe)
      8. PeriodoController (Classe)
      9. TurmaController (Classe)
   3. Model:
      1. Entity:
         1. AlunoEntity (Classe)
         2. DisciplinaEntity (Classe)
         3. FuncionarioEntity (Classe)
         4. LoginEntity (Classe)
         5. NotaEntity (Classe)
         6. PeriodoEntity (Classe)
         7. TurmaEntity (Classe)
      2. Mapper:
         1. AlunoModelMapper (Interface)
         2. DisciplinaModelMapper (Interface)
         3. FuncionarioModelMapper (Interface)
         4. LoginModelMapper (Interface)
         5. NotaModelMapper (Interface)
         6. PeriodoModelMapper (Interface)
         7. TurmaModelMapper (Interface)
      3. Repository:
         1. AlunoRepository (Interface)
         2. DisciplinaRepository (Interface)
         3. FuncionarioRepository (Interface)
         4. LoginRepository (Interface)
         5. NotaRepository (Interface)
         6. PeriodoRepository (Interface)
         7. TurmaRepository (Interface)
      4. AlunoModel (Classe)
      5. DisciplinaModel (Classe)
      6. FuncionarioModel(Classe)
      7. LoginModel(Classe)
      8. NotaModel (Classe)
      9. PeriodoModel (Classe)
      10. TurmaModel (Classe)
   4. Resources:
      1. Db.migration:
         1. V1\_db\_estrutura.sql
         2. V2\_db\_dados.sql
      2. Aplication.yaml
      3. Aplication-db.yaml
      4. Aplication-local.yaml

## Outras informações

Estou trabalhando em uma aplicação Java Maven com Spring Boot, e decidi organizar meu código de uma maneira que fosse tanto funcional quanto fácil de entender para mim e também para corresponder ao modelo bastante utilizado no mercado de trabalho. Comecei definindo o pacote raiz como **pt.colegio.colibri**, que representa o nome do meu projeto.

Dentro deste pacote raiz, criei o pacote **business**, onde coloquei todas as classes que contêm a lógica principal da minha aplicação. Dividi isso ainda mais em dois subpacotes: core e service. No core, coloquei as classes que são fundamentais para o funcionamento do meu negócio, enquanto no service, coloquei as classes de serviço que encapsulam a lógica de negócios específica.

Para a configuração global da aplicação, criei o pacote **config**, dividido em error e security. No error, coloquei classes relacionadas à manipulação de erros, como exceções personalizadas e filtros de erro. Já no security, agrupei as configurações de segurança, incluindo autenticação, autorização e proteção contra ataques comuns.

No pacote **controller,** organizei as classes de controlador, que lidam com as requisições HTTP e respostas. Dividi isso em dtos (Data Transfer Objects) e mapper. Os dtos representam os dados enviados entre processos ou camadas da aplicação, incluindo objetos para solicitações e respostas. Os mappers são usados para converter objetos DTOs em entidades e vice-versa, facilitando a transferência de dados entre diferentes partes da aplicação.

Finalmente, no pacote **model**, coloquei as definições de entidade e mapeadores, além dos repositórios. As entidades representam os conceitos e relações do mundo real que minha aplicação está modelando. Os mappers ajudam na conversão entre entidades e DTOs, garantindo a consistência dos dados ao longo da aplicação. Os repositórios abstraem o acesso ao banco de dados, permitindo que eu me concentre na lógica de negócios sem me preocupar com os detalhes de persistência.

Esta estrutura me ajuda a manter o código organizado e modular, facilitando a navegação pelo projeto e a separação de responsabilidades entre diferentes partes da aplicação.