



Documentação Completa: Sistema de Gerenciamento de Academia

1. Introdução

1.1. Objetivo

Este projeto tem como propósito oferecer uma solução eficiente para o gerenciamento de academias, facilitando o cadastro de alunos, planos de assinatura e fichas de treino. Ele visa integrar funcionalidades essenciais para garantir uma administração organizada e acessível, além de permitir futuras expansões, como autenticação de usuários e integração com sistemas financeiros.

O sistema foi desenvolvido utilizando **Java** e **MySQL**, garantindo uma estrutura robusta, modular e de fácil manutenção.

1.2. Escopo

O sistema abrange funcionalidades essenciais para academias de pequeno e médio porte, incluindo:

- Cadastro e gerenciamento de alunos, com informações detalhadas.
- Associação de fichas de treino para monitoramento da evolução dos alunos.
- Gestão de planos de assinatura, permitindo flexibilização de preços e períodos.
- Monitoramento de status dos alunos: ativos, inativos ou bloqueados.
- Persistência de dados utilizando **MySQL**.
- Implementação de padrões arquitetônicos como **DAO** para separação de lógica de negócio e persistência.

A estrutura modular permite futuras integrações, como:

- Integração com sistemas de pagamento.
 - API REST para acesso via aplicações web ou mobile.
 - Módulo de autenticação e autorização.
-

2. Arquitetura do Sistema

2.1. Tecnologias Utilizadas

- **Backend:** Java (paradigma orientado a objetos).
 - **Banco de Dados:** MySQL.
 - **Gerenciamento de Dependências:** Maven.
 - **Interação com o usuário:** Linha de comando.
 - **Testes:** JUnit (futuro).
 - **CI/CD:** Scripts manuais, com previsão de integração futura via **GitHub Actions**.
-

2.2. Estrutura de Pastas e Arquitetura

```
ProjetoAcademia
├── .vscode      # Configurações do ambiente de desenvolvimento
├── bin          # Arquivos compilados (.class)
├── lib          # Bibliotecas externas (ex: mysql-connector-j)
├── src         # Código-fonte
│   ├── conexao # Gerenciamento da conexão com o banco
│   │   └── sql  # Scripts SQL de criação de banco e tabelas
│   ├── DAO     # Classes de acesso a dados (Data Access Object)
│   └── entity   # Definição das entidades do sistema
```

2.3. Explicação de Cada Diretório

- **.vscode** : Configurações do editor.
- **bin** : Contém os arquivos compilados.
- **lib** : Contém o conector **MySQL** (`mysql-connector-j-9.2.0.jar`).
- **src/conexao** : Classes responsáveis por estabelecer a conexão com o banco.

- **src/DAO** : Implementa os métodos CRUD e abstrai o acesso ao banco.
- **src/entity** : Representa as entidades do domínio, como Aluno, Plano, etc.

3. Funcionamento do Sistema

3.1. Estrutura de Dados (Banco de Dados)

O banco de dados possui tabelas inter-relacionadas para garantir integridade e consistência.

 **Exemplo de criação da tabela **Aluno** :**

```
CREATE TABLE Aluno (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    nome VARCHAR(100),  
    idade INT,  
    endereco VARCHAR(255),  
    plano_id INT,  
    FOREIGN KEY (plano_id) REFERENCES Plano(id)  
);
```

3.2. Comunicação entre Camadas

A arquitetura segue o padrão **DAO**, separando lógica de negócios e persistência.

 **Exemplo: Cadastrar Aluno**

```
public class AlunoDao {  
    public void cadastrarAluno(Aluno aluno) {  
        Connection conexao = Conexao.getConnection();  
        String sql = "INSERT INTO Aluno (nome, idade, endereco, plano_id) VA  
LUES (?, ?, ?, ?)";  
  
        try (PreparedStatement stmt = conexao.prepareStatement(sql)) {  
            stmt.setString(1, aluno.getNome());  
            stmt.setInt(2, aluno.getIdade());  
            stmt.setString(3, aluno.getEndereco());  
            stmt.setInt(4, aluno.getPlanold());  
        }  
    }  
}
```

```
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}
```

4. Pipeline DevOps

4.1. Integração Contínua (CI)

A compilação pode ser realizada com:

```
javac -cp ".;../lib/mysql-connector-j-9.2.0.jar" *.java conexao/*.java DAO/*.java entity/*.java
```

Melhorias previstas:

- Automação de builds com **Maven**.
- Integração com serviços como **GitHub Actions**.

4.2. Entrega Contínua (CD)

A execução do sistema ocorre via:

```
java -cp ".;../lib/mysql-connector-j-9.2.0.jar" SistemaAcademia
```

Futuro:

- Deploy automatizado.
- Containerização com **Docker**.

5. Segurança e Compliance

5.1. Autenticação e Controle de Acesso

Status atual: Não implementado.

Melhoria prevista:

Módulo de autenticação com controle de perfis (administrador, instrutor, aluno).

5.2. Proteção de Dados

- Futuras implementações de **criptografia** para dados sensíveis.
 - Conexões seguras utilizando **SSL**.
-

6. Requisitos do Sistema

- **Java**: JDK 11 ou superior.
 - **MySQL**: Servidor local ou remoto configurado.
 - **IDE**: Visual Studio Code, IntelliJ IDEA ou Eclipse.
 - **Bibliotecas**: MySQL Connector ([lib/mysql-connector-j-9.2.0.jar](#)).
-

7. Boas Práticas e Padrões Utilizados

Padrão DAO: Facilita manutenção e testabilidade.

Orientação a Objetos: Promove modularidade e reaproveitamento.


Modularização: Separação clara entre entidades, lógica de negócio e persistência.

8. Como Acessar e Executar o Projeto

Este projeto está disponível publicamente no GitHub, permitindo que qualquer desenvolvedor interessado possa clonar, estudar ou colaborar com o desenvolvimento.

8.1. Acessando o Repositório

O código-fonte completo está hospedado no GitHub, no seguinte link:

 <https://github.com/Gato-Fantasma/Sistema-de-Gestao-de-Academia.git>

8.2. Passos para Executar o Projeto

1. Clonar ou Baixar o Repositório:

- Você pode clonar o projeto utilizando o seguinte comando no terminal:

```
bash
CopiarEditar
git clone https://github.com/usuario/projeto-academia.git
```

- Ou então, acessar o link do GitHub e clicar em "**Code**" → "**Download ZIP**", e depois extrair o arquivo ZIP em sua máquina.

2. Abrir o Projeto:

- Após extrair o arquivo ZIP, abra a pasta do projeto no **Visual Studio Code** (VSCode) ou na sua IDE Java de preferência.

3. Configuração do Ambiente:

- Certifique-se de ter o **Java Development Kit (JDK)** instalado (preferencialmente versão 11 ou superior).
- Configure as dependências, caso esteja utilizando um gerenciador como **Maven** ou **Gradle**.
- O projeto inclui o driver MySQL (`mysql-connector-j-9.2.0.jar`) na pasta `lib` , que deve ser adicionado ao **classpath**.

4. Configurar o Banco de Dados:

- O script de criação das tabelas está localizado em `src/conexao/sql/Banco.sql` .
- Execute esse script no seu **MySQL** local ou remoto.
- Atualize o arquivo de configuração da conexão (`Conexao.java`) com os dados corretos de host, usuário e senha.

5. Compilar o Projeto:

- Acesse a pasta raiz do projeto no terminal e compile com o comando:

```
bash
CopiarEditar
javac -cp ".;lib/mysql-connector-j-9.2.0.jar" src/conexao/*.java src/DAO/*.java src/entity/*.java src/*.java -d bin
```

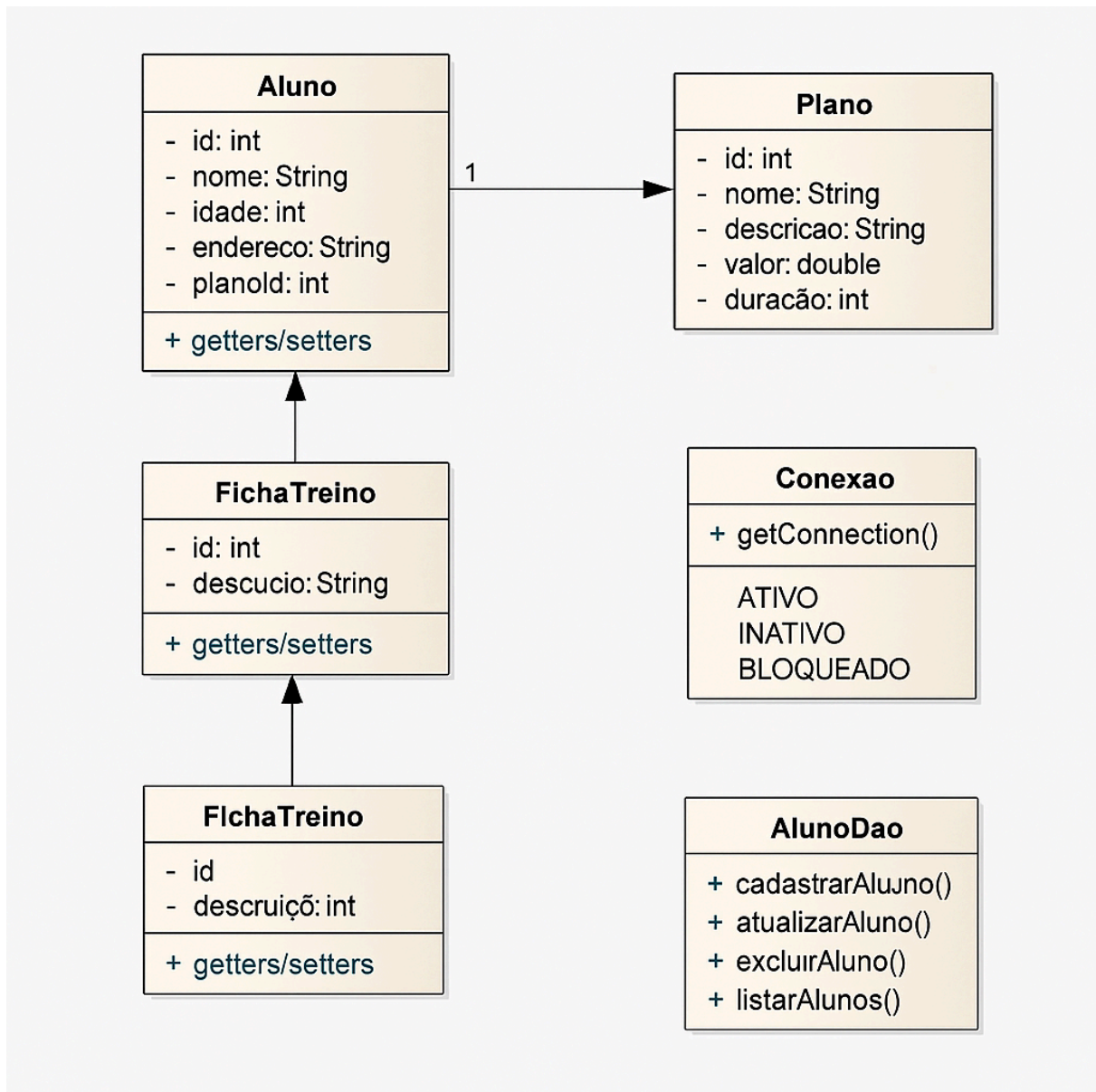
6. Executar o Sistema:

- Após compilar, execute o sistema com o comando:

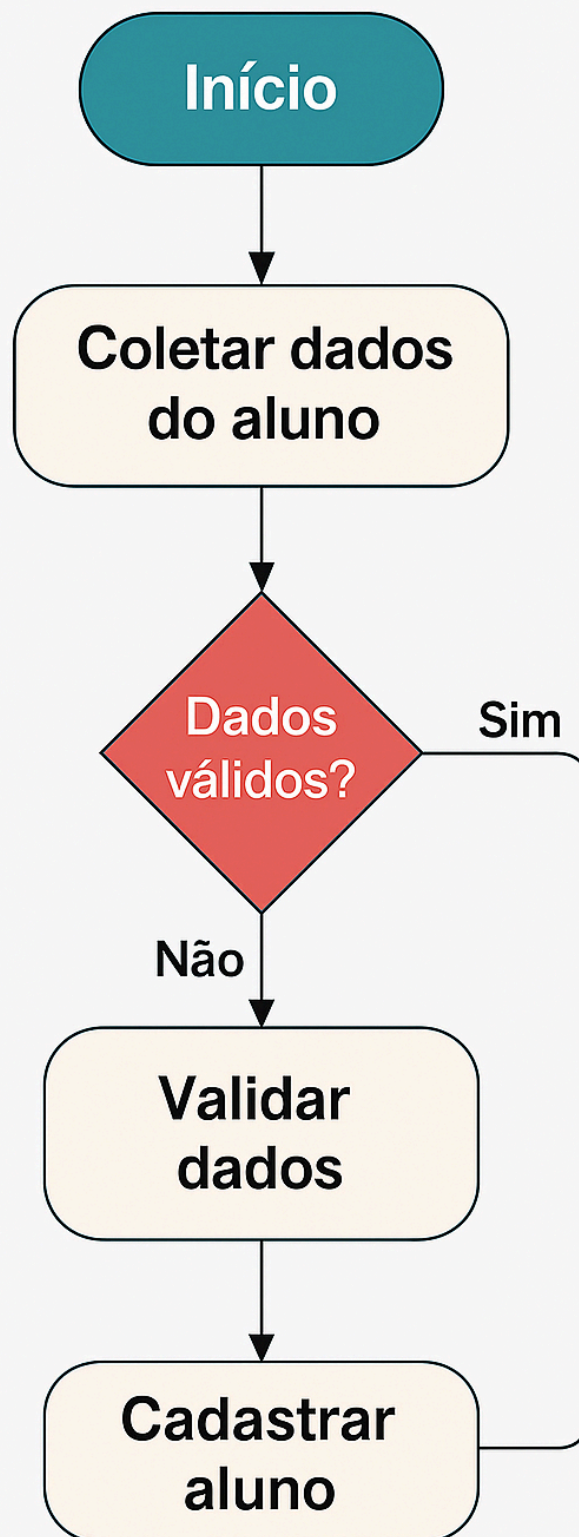
```
bash
CopiarEditar
java -cp "lib/mysql-connector-j-9.2.0.jar;bin" SistemaAcademia
```

9. Diagramas

9.1. Diagrama de Classes



9.2. Diagrama de Fluxo – Processo de Cadastro de Aluno



10. Considerações Finais

Este sistema de gerenciamento de academias é um projeto robusto, modular e escalável. Sua arquitetura baseada em padrões sólidos garante facilidade de manutenção e evolução.

Principais Qualidades:

- Modularidade.
- Facilidade de manutenção.
- Padrões arquiteturais bem aplicados.
- Facilidade de expansão.