25/10/2025

Documentação projeto Sistema de academia

Ambiente de dados

Equipe:

Lídia Araújo - 2410391

Ana Karolina Silva Ferreira - 2410489

Documentação projeto Sistema de academia

Funcionamento do Sistema

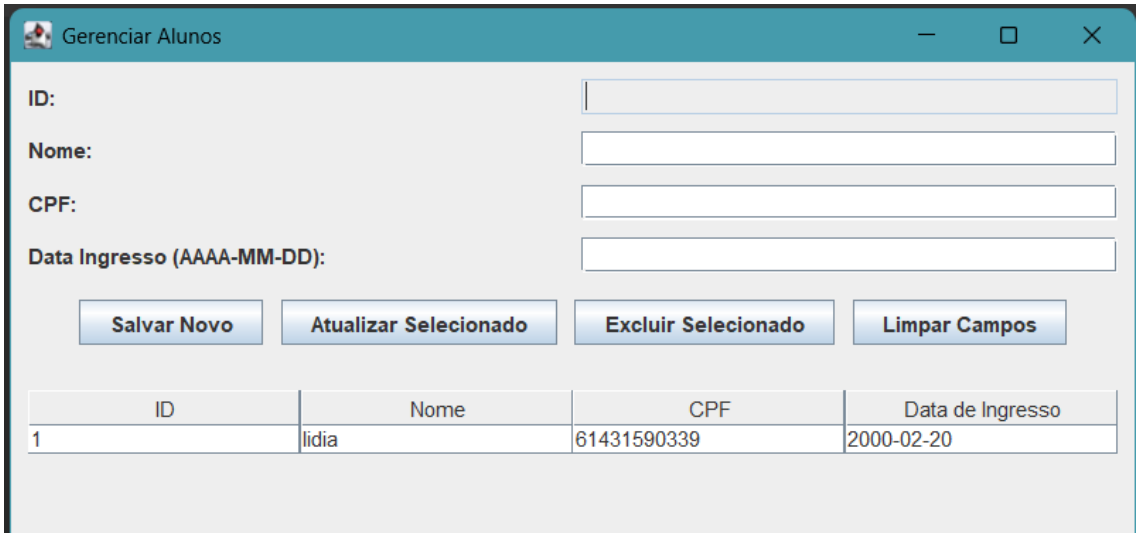
O **Sistema da Academia** foi desenvolvido para facilitar o gerenciamento de informações relacionadas aos alunos, instrutores, planos de treino e frequência.
Seu objetivo é oferecer uma interface simples e intuitiva para que administradores possam realizar cadastros, consultas e atualizações de forma rápida e organizada.

Ao iniciar o programa, é exibida a **tela principal** com as seguintes opções de gerenciamento:

A tela apresenta as seguintes opções:

- **Gerenciar Alunos:** permite cadastrar, atualizar e remover alunos. No cadastro, são solicitadas informações como(“nome, CPF, ID e data de ingresso”).

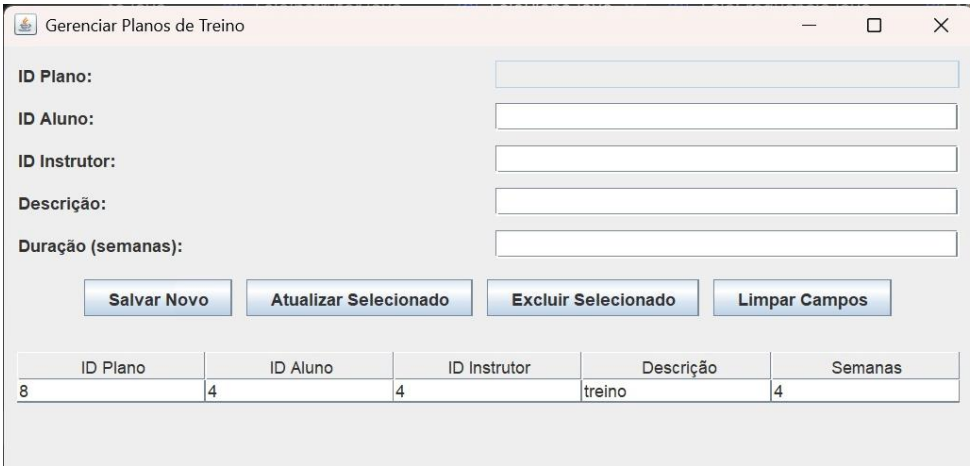
Documentação projeto Sistema de academia



- **Gerenciar Instrutores:** possibilita o cadastro e gerenciamento dos instrutores, incluindo campos como (“nome, ID e especialidade”).

- **Gerenciar Planos de Treino:** neste módulo é possível criar planos de treino personalizados, associando o ID do aluno, o ID do instrutor , descrição do que o aluno deseja , além de definir os dias da semana correspondentes.

Documentação projeto Sistema de academia



- **Gerenciar Frequência:** registra a presença dos alunos nas aulas, associando o ID do aluno, o ID do instrutor , descrição. Permitindo acompanhar a assiduidade e controle de treinos.

- **Sair:** encerra a execução do sistema e fecha a aplicação.

Toda a comunicação entre a interface gráfica e o banco de dados é feita por meio das **classes DAO (Data Access Object)**. Essas classes utilizam o **driver JDBC** para conectar o sistema ao **banco de dados MySQL**, onde todas as informações são armazenadas .

O banco de dados é composto por tabelas como:

- aluno — armazena os dados pessoais dos alunos;

- instrutor — contém as informações e especialidades dos instrutores;
- plano_treino — define a relação entre aluno, instrutor e dias de treino;
- frequencia — registra a presença dos alunos.

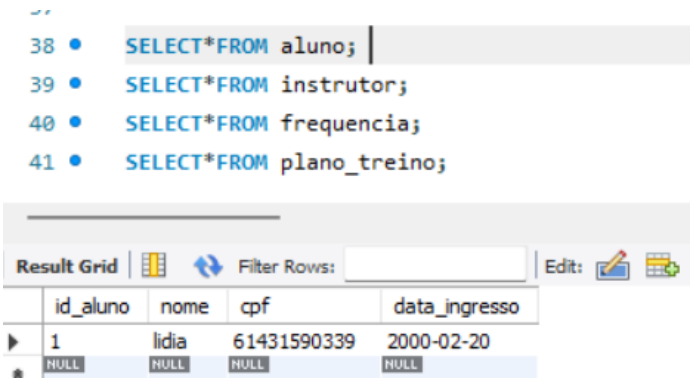
Funcionamento do Banco de Dados

Para visualizar as informações cadastradas no sistema, é necessário acessar o **MySQL** e utilizar o comando:

SELECT * FROM nome_da_tabela;

Por exemplo:

Ao executar essa linha, o MySQL exibirá todos os registros salvos na tabela **aluno** — como o nome, CPF e data de ingresso de cada aluno.
O mesmo pode ser feito para as outras tabelas:



Esses comandos permitem verificar se os dados cadastrados pelo sistema estão sendo salvos corretamente no banco de dados.

Dessa forma, cada ação feita nas telas (como cadastrar ,atualizar, excluir) é refletida diretamente no banco de dados, garantindo a integridade e consistência das informações.

Guia Passo a Passo para um Desenvolvedor Iniciar do Zero:

Abaixo descrevemos a sequência que seguimos no desenvolvimento, explicando as classes-chave criadas em cada etapa e como elas se relacionam.

1- Baixando o Eclipse IDE

O primeiro passo é a versão mais recente do Eclipse. Siga os passos abaixo:

1. Acesse o site oficial do Eclipse: https://www.eclipse.org/downloads/.
2. Clique no botão **"Download"** para baixar o instalador mais recente.
3. Aguarde o download ser concluído antes de seguir para a instalação.

Após baixar o instalador, siga os passos abaixo:

1. Execute o arquivo baixado (**eclipse-inst-jre-win64.exe**).
2. 1Na tela inicial, escolha a opção **Eclipse IDE for Java Developers**.
3. Escolha um local para a instalação ou mantenha o diretório padrão sugerido.
4. Clique em **Install** e aguarde a instalação ser concluída.
5. Quando finalizar, clique em **Launch** para abrir o Eclipse pela primeira vez.

Configurando o workspace

Documentação projeto Sistema de academia

Ao abrir o Eclipse pela primeira vez, será solicitado que você escolha um **workspace**. O workspace é o diretório onde seus projetos serão armazenados.

1. Escolha um local adequado no seu computador.
2. Marque a opção **"Use this as the default and do not ask again"** para evitar essa tela no futuro.
3. Clique em **Launch** para iniciar o Eclipse.

Agora seu Eclipse está pronto para ser usado!

Criando projeto Java no Eclipse

Agora que está instalado e configurado, vamos criar um projeto Java para garantir que está tudo funcionando corretamente.

1. No Eclipse, vá até **File à New Java à Project.**
2. um nome ao seu projeto (exemplo: SistemaAcademia).
3. Clique em **Finish**.
4. Dentro do projeto criado, clique com o botão direito em **src à New package**.
5. Dê um nome para a **package** (exemplo: dao) .
6. Dentro da **package** criada, clique com o botão direito , depois clique em **New, class**.
7. Clique em finish
8. Dê um nome para a classe (exemplo: Main) .
9. Clique em finish

Baixando MySQL Workbench

1. Para baixar o MySQL, você precisa acessar o site oficial do MySQL. Lá, selecione a versão do MySQL que deseja instalar e baixe-a. É recomendável baixar a versão mais recente disponível.
   Site oficial: https://dev.mysql.com/downloads/workbench/

Documentação projeto Sistema de academia

Após o download, você deve iniciar o processo de instalação do MySQL. Siga as instruções na tela e aceite os termos e condições da licença. É importante selecionar as opções de configuração apropriadas durante a instalação.

Configurar o MySQL

Após a instalação do MySQL, ele precisa ser configurado antes de ser utilizado. Recomenda-se criar um banco de dados e um usuário para acessá-lo. Para isso, você pode usar a ferramenta **banco de trabalho mysql.**

Criando um banco de dados no MySQL Workbench passo a passo:

1. **Efetue login no MySQL Workbench**

A primeira coisa que você precisa fazer é acessar o MySQL Workbench. Se você já possui uma conta, acesse com suas credenciais. Caso contrário, crie uma nova.

2. **Criar uma nova conexão**

Após efetuar login, clique na aba "Banco de Dados" e selecione "Nova Conexão". Em seguida, insira as informações necessárias, como nome da conexão, host, porta, nome de usuário e senha. Clique em "Testar Conexão" para garantir que a conexão seja estabelecida corretamente.

3. **Criar um novo banco de dados**

Após estabelecer a conexão, clique na aba "Esquema" e selecione "Criar Esquema". Digite o nome do banco de dados que deseja criar e clique em "Aplicar".

4. **Criar tabelas no banco de dados**

Após criar o banco de dados, você pode começar a criar tabelas. Clique na aba "Esquema" e selecione o banco de dados que você acabou de criar. Em seguida, clique em "Criar Tabela" e insira o nome da tabela e as colunas que deseja adicionar a ela.

5. **Adicionar dados à tabela**

Depois de criar a tabela, você pode começar a adicionar dados a ela. Clique na aba "Esquema" e selecione o banco de dados e a tabela que deseja editar. Em seguida, clique em "Inserir Dados" e adicione os dados que deseja adicionar à tabela.

Documentação projeto Sistema de academia

Iniciando o Projeto

Após concluir a instalação de todas as dependências e configurar o banco de dados, podemos iniciar o projeto para executar o sistema localmente.

Etapa 1 – Criação das Classes Construtoras (Model)

O desenvolvimento iniciou pelas classes (Model), responsáveis por representar as entidades do sistema.

Cada classe contém os atributos, construtores, métodos getters e setters, e serve de base para manipulação dos dados no sistema.

Exemplo:

Classe Aluno: representa um aluno da academia, com atributos como id, nome, cpf e data_ingresso.

Classe Instrutor: armazena dados do instrutor, incluindo especialidade.

Classe PlanoTreino: define o plano de treino vinculado a um aluno e um instrutor, com os dias da semana e descrição dos exercícios.

Essas classes não acessam diretamente o banco de dados elas são apenas "modelos" de objetos que serão salvos ou exibidos na interface**.**

Passo 2 — Conexão com o Banco de Dados

A classe Conexao é a responsável por criar a comunicação entre o Java e o MySQL.
Ela utiliza o driver JDBC (adicionado via arquivo .jar) para estabelecer essa conexão.

Exemplo de funcionamento:

- Define o endereço do banco (jdbc:mysql://localhost:3306/academia), usuário e senha.

- O método getConnection() retorna uma instância ativa da conexão, que será usada pelas classes DAO.
- Se houver erro de conexão (como credenciais erradas ou banco desligado), o sistema exibe a exceção no console.

- Conexao.java — centraliza URL, usuário, senha, e método getConnection() que retorna Connection.

```java
public class Conexao {
    private static final String URL = "jdbc:mysql://localhost:3306/academia";
    private static final String USER = "root";       //  usuário MySQL
    private static final String PASSWORD = "1234"; // SENHA

    public static Connection getConnection() throws SQLException {

        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

Passo 3 – Criação das Classes DAO

As classes DAO são responsáveis por realizar as operações de CRUD (Create, Read, Update, Delete) no banco de dados. Elas usam a classe Conexao para abrir a conexão e executar comandos SQL.

Exemplo:

Classe AlunoDAO

Essa classe gerencia os dados da tabela aluno.

- O método inserir(Aluno aluno) adiciona um novo registro no banco.
- O método listar() retorna todos os alunos cadastrados.
- O método atualizar(Aluno aluno) edita os dados de um aluno existente.
- O método excluir(int id) remove um aluno a partir de seu ID.

Esses métodos utilizam a interface PreparedStatement, que:

Documentação projeto Sistema de academia

- Evita SQL Injection (aumentando a segurança);
- Facilita o envio de parâmetros (?) nas consultas;
- Fecha automaticamente os recursos após o uso (graças ao try-with-resources.

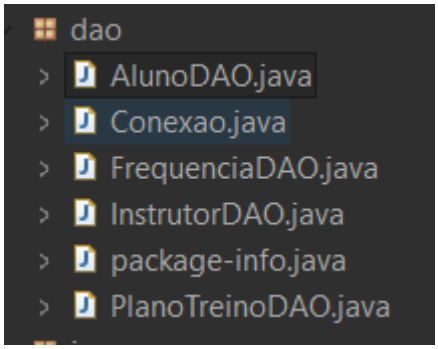
```java
public class AlunoDAO {
    public void inserir(Aluno aluno) {
        String sql = "INSERT INTO aluno (nome, cpf, data_ingresso) VALUES (?, ?, ?)";
        try (Connection conn = Conexao.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, aluno.getNome());
            stmt.setString(2, aluno.getCpf());
            stmt.setDate(3, aluno.getDataIngresso());
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```java
  public List<Aluno> listar() {
      List<Aluno> lista = new ArrayList<>();
      String sql = "SELECT * FROM aluno";
      try (Connection conn = Conexao.getConnection();
           PreparedStatement stmt = conn.prepareStatement(sql);
           ResultSet rs = stmt.executeQuery()) {
          while (rs.next()) {
              Aluno a = new Aluno();
              a.setId(rs.getInt("id_aluno"));
              a.setNome(rs.getString("nome"));
              a.setCpf(rs.getString("cpf"));
              a.setDataIngresso(rs.getDate("data_ingresso"));
              lista.add(a);
          }
      } catch (SQLException e) {
          e.printStackTrace();
      }
      return lista;
```

Documentação projeto Sistema de academia

```java
public void atualizar(Aluno aluno) {
    String sql = "UPDATE aluno SET nome = ?, cpf = ?, data_ingresso = ? WHERE id_aluno = ?";
    try (Connection conn = Conexao.getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setString(1, aluno.getNome());
        stmt.setString(2, aluno.getCpf());
        stmt.setDate(3, aluno.getDataIngresso());
        stmt.setInt(4, aluno.getId());
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public void excluir(int id) {
    String sql = "DELETE FROM aluno WHERE id_aluno = ?";
    try (Connection conn = Conexao.getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setInt(1, id);
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
```



- As outras seguimos nessa estrutura.

Passo 4 — servisse - Regras de negócio e validações

- Após as classes DAO , vamos para service

Objetivo: validar dados, aplicar regras (ex.: não cadastrar aluno menor de idade sem autorização), orquestrar chamadas a múltiplos DAOs.

Exemplo de classe:

AcademiaService.java — métodos como cadastrarAluno(Aluno a), que valida (nome não vazio, data válida) e chama AlunoDAO.inserir(a).

Estrutura do Projeto

O sistema da academia foi desenvolvido em **Java**, utilizando o **Eclipse** como ambiente de desenvolvimento.
A estrutura do projeto foi organizada em pacotes (packages) para manter o código limpo, modular e fácil de manter.

Documentação projeto Sistema de academia

Passo 5: view — Interface (Swing)
Cada pacote possui uma função específica dentro da aplicação, conforme descrito abaixo:

Objetivo: telas para interagir (cadastro, edição, listagem).

Telas/Classes usadas:

TelaPrincipal.java — menu com botões: Alunos, Instrutores, Planos, Frequência, Sair.

TelaAluno.java — campos: Nome, ID (campo desabilitado para novos cadastros), Data de Nascimento; botões Salvar, Editar, Excluir, Listar.

TelaInstrutor.java — campos: Nome, ID, Data de Nascimento, Especialidade.

TelaPlano.java — campos: ID Aluno, ID Instrutor, Descrição/Plano, Dias da Semana (checkboxes ou lista).

Passo 6: Classe principal / Inicialização

- **Arquivo:** Main.java
- **Função:** inicializa possível configuração (carregar .properties, verificar DB) e abre TelaPrincipal.

```
public class Main {
    public static void main(String[] args) {
        new TelaPrincipal().setVisible(true);
    }
}
```

Passo 7: Arquivo .jar do MySQL Connector

- **Como adicionamos:** mysql-connector-java-x.x.x.jar → Build Path → Add External JARs no Eclipse.
- **Por que:** esse .jar é o driver JDBC necessário para DriverManager.getConnection(...).

Documentação projeto Sistema de academia

Passo 8: Scripts SQL e dados de exemplo

- **O que incluir no repositório:** script_banco.sql contendo CREATE DATABASE academia; CREATE TABLE ... e INSERT com alguns dados de exemplo (1 instrutor, 2 alunos, 1 plano).
- **Como usar:** importar o script via MySQL Workbench ou linha de comando antes de rodar a aplicação.

Passo 9: Testes rápidos / Execução

- **Passos para testar localmente:**
    1. Importar o projeto no Eclipse.
    2. Adicionar o .jar do conector.
    3. Criar o DB e importar script_banco.sql.
    4. Ajustar Conexao.java se necessário (usuário/senha).
    5. Rodar Main.java.
    6. Testar: cadastrar um aluno, cadastrar um instrutor, criar um plano (associando aluno + instrutor), registrar frequência.
- **O que verificar:** se os registros aparecem no banco, se ao deletar/editar a alteração é refletida.

Guia passo a passo (resumido)

1- Modelagem: defini as entidades Aluno, Instrutor, PlanoTreino e Frequencia.

2- Model (classes construtoras): criei POJOs no pacote model com construtores, getters/setters e toString().

3- DAO (persistência): implementei AlunoDAO, InstrutorDAO, PlanoTreinoDAO e FrequenciaDAO, e a classe Conexao.java que fornece conexões via DriverManager. Usei PreparedStatement para as operações CRUD.

Documentação projeto Sistema de academia

4- Service (regras): AcademiaService concentra validações e orquestra operações entre a view e os DAOs.

5- View (UI): telas em Swing (TelaPrincipal, TelaAluno, TelaInstrutor, TelaPlano) que chamam os métodos do AcademiaService.

6- Inicialização: Main.java inicia a aplicação carregando a TelaPrincipal.

7- Dependência JDBC: adicionei o mysql-connector-java-x.x.x.jar ao projeto (Build Path) para permitir a comunicação com o MySQL.

8- Scripts: incluí script_banco.sql para criar as tabelas e inserir dados de exemplo.

Seguindo essa sequência, um desenvolvedor consegue reproduzir a aplicação do zero, entendendo a função de cada classe e como elas interagem.

Orientações para Reprodução do Projeto

Para reproduzir o projeto em outro ambiente, siga as orientações abaixo:

- **IDE utilizada:** Eclipse
- **Linguagem:** Java
- **Banco de Dados:** MySQL
- **Driver JDBC:** mysql-connector-java
- **Estrutura de pacotes:** model, dao, service, view
- **Classe principal:** Main.java

Antes de executar o projeto, o desenvolvedor deve:

Documentação projeto Sistema de academia

1. Instalar o MySQL e criar o banco de dados academia.
2. Importar o script SQL que cria as tabelas (aluno, instrutor, plano_treino, frequencia).
3. Configurar o arquivo Conexao.java com o usuário e senha do banco local.
4. Importar o driver JDBC no Eclipse (Build Path → Configure Build Path → Add External JARs).
5. Executar a aplicação e validar o funcionamento das telas e conexões.

Seguindo esses passos, qualquer desenvolvedor poderá reproduzir o projeto completo e testar todas as funcionalidades do Sistema da Academia em seu próprio ambiente.