Felipe
Tagawa
Reis

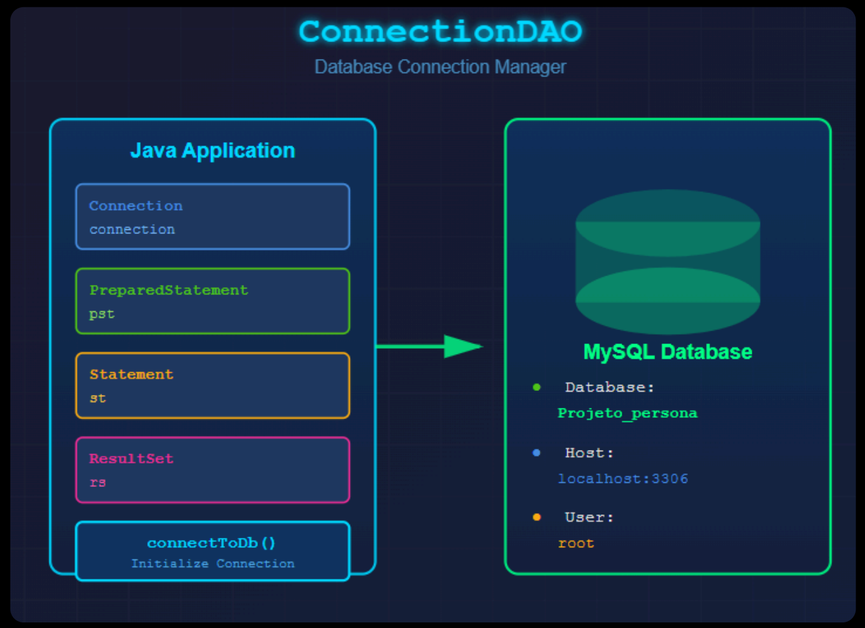
Integração Java MySQL

Instituto Nacional de
Telecomunicações -
INATEL

Classe Principal de Conexão

```java
public abstract class ConnectionDAO {

    Connection connection;          // Conexão com o banco

    // Parâmetros utilizados nas subclasses:

    PreparedStatement pst;          // Comando SQL com parâmetros
    Statement st;                   // Comando SQL simples (sem parâmetros)
    ResultSet rs;                   // Resultado das consultas SQL

    // Informações de acesso ao banco de dados:
    String database = "Projeto_persona"; // Nome do BD
    String user = "root";
    String password = "root";
    String url = "jdbc:mysql://localhost:3306/" + database;
```

```java
// Estabelecer a conexão com o banco:
public Connection connectToDb(){
    try {
        connection = DriverManager.getConnection(url, user, password);
    } catch (SQLException e) {
        System.out.println("Erro ao conectar com o banco de dados: " + e.getMessage());
    }
    return null;
}
```



A classe ConnectionDAO é uma classe abstrata que gerencia a conexão entre aplicações Java e o banco de dados MySQL. Ela centraliza os componentes essenciais (Connection, PreparedStatement, Statement e ResultSet) e as credenciais de acesso ao banco. Serve como base para outras classes DAO, promovendo reutilização de código e padronização no acesso aos dados do sistema.

Como Conectar?

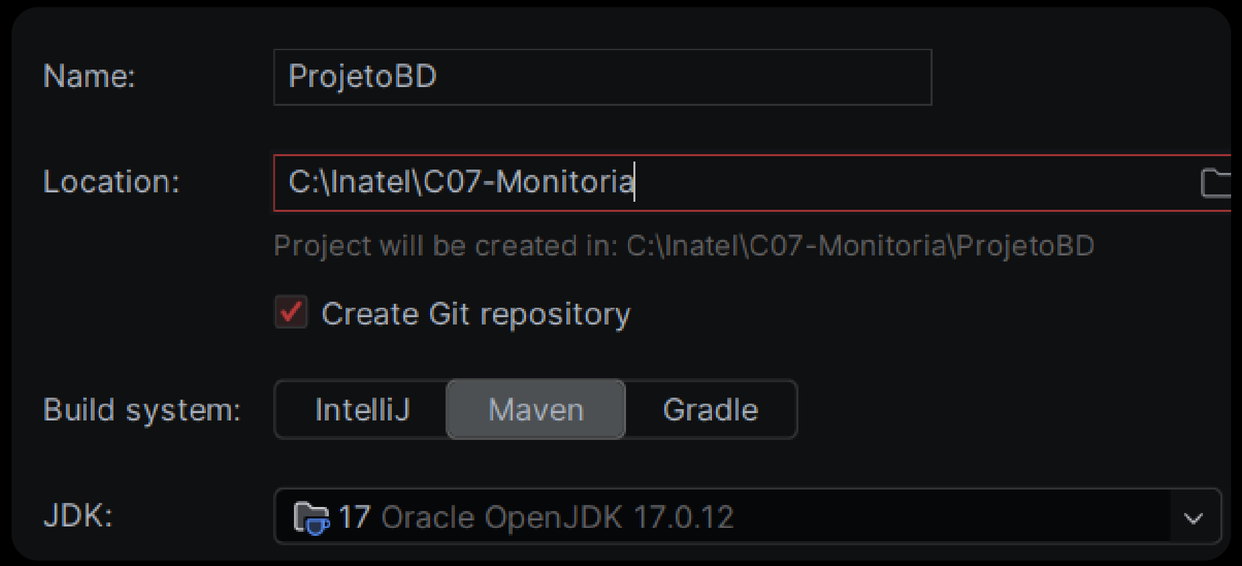
File

↓

New

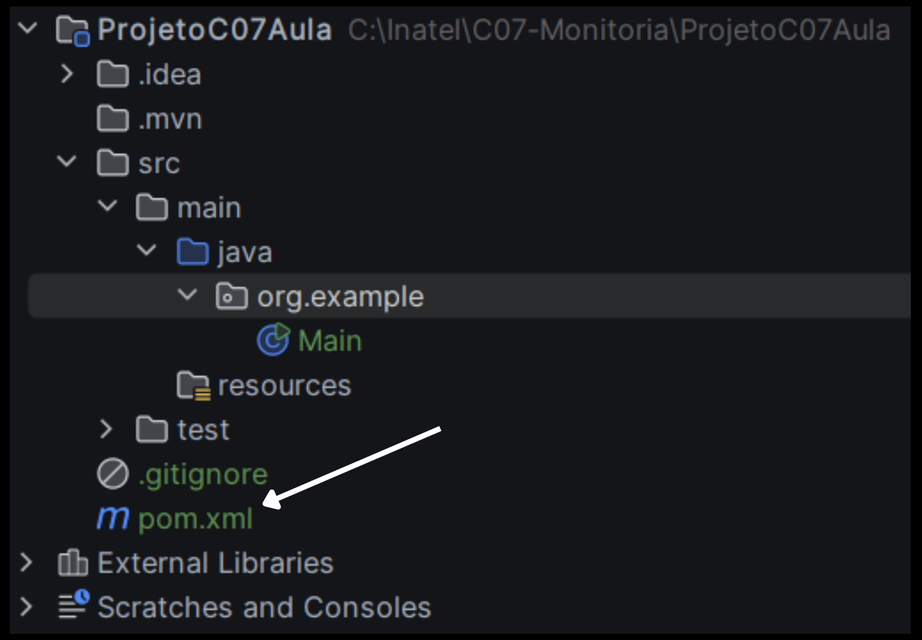
↓

Project...



Maven™

Prosseguindo...




```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>Integracao_SQL</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>23</maven.compiler.source>
        <maven.compiler.target>23</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.33</version>
        </dependency>
    </dependencies>

</project>
```

Como Inserir Dados/Instâncias dentro do Java?

```java
public class NPCDAO extends ConnectionDAO {

    public boolean insertNPC(NPC npc) {
        connectToDb(); // Abre conexão
        String sql = "INSERT INTO NPC(nome, idade, genero, ocupacao, arcana) VALUES(?, ?, ?, ?, ?)";

        try {
            pst = connection.prepareStatement(sql);              // Prepara comando
            pst.setString( parameterIndex: 1, npc.getNome());    // Seta parâmetro 1
            pst.setInt( parameterIndex: 2, npc.getIdade());      // Seta parâmetro 2
            pst.setString( parameterIndex: 3, npc.getGenero());  // Seta parâmetro 3
            pst.setString( parameterIndex: 4, npc.getOcupacao()); // Seta parâmetro 4
            pst.setString( parameterIndex: 5, npc.getArcana());  // Seta parâmetro 5
            pst.execute();                                       // Executa INSERT
            return true;

        } catch (SQLException e) {
            System.out.println("Erro ao inserir NPC: " + e.getMessage());
            return false;

        } finally {
            // SEMPRE executa - libera recursos na ordem correta
            try {
                if (pst != null) pst.close();        // Fecha statement primeiro
                if (connection != null) connection.close(); // Depois fecha conexão
            } catch (SQLException e) {
                System.out.println("Erro ao fechar recursos: " + e.getMessage());
            }
        }
    }
```

```java
public static void popularNPCs(Map<String, NPC> npcs) {


    npcs.put("Bunkichi e Mitsuko", new NPC( nome: "Bunkichi e Mitsuko",
    npcs.put("Kenji", new NPC( nome: "Kenji Tomochika", idade: 16, genero:
    npcs.put("Kazushi", new NPC( nome: "Kazushi Miyamoto", idade: 17, ger
    npcs.put("Odagiri", new NPC( nome: "Hidetoshi Odagiri", idade: 17, ge
    npcs.put("Yuko", new NPC( nome: "Yuko Nishiwaki", idade: 17, genero: "
    npcs.put("Chihiro", new NPC( nome: "Chihiro Fushimi", idade: 16, gene
    npcs.put("Maya", new NPC( nome: "Maya", idade: 27, genero: "Feminino",
    npcs.put("Suemitsu", new NPC( nome: "Nozomi Suemitsu", idade: 15, ger
    npcs.put("Hiraga", new NPC( nome: "Keisuke Hiraga", idade: 17, genero:
    npcs.put("Akinari", new NPC( nome: "Akinari Kamiki", idade: 17, gener
    npcs.put("Mutatsu", new NPC( nome: "Mutatsu", idade: 55, genero: "Masc
    npcs.put("Maiko", new NPC( nome: "Maiko", idade: 8, genero: "Feminino"
    npcs.put("Bebé", new NPC( nome: "Bebé", idade: 17, genero: "Masculino"
    npcs.put("Tanaka", new NPC( nome: "Tanaka", idade: 36, genero: "Mascul
    npcs.put("Pharos", new NPC( nome: "Pharos", idade: 8, genero: "indefin
    npcs.put("Hayase", new NPC( nome: "Hayase", idade: 17, genero: "Mascul

    // Inserção no banco de dados:
    NPCDAO npcDAO = new NPCDAO();
    for (String npcName : NPC_NAMES) {
        if (npcs.containsKey(npcName)) {
            npcDAO.insertNPC(npcs.get(npcName));
        }
    }
}
```

Como Atualizar Dados/Instâncias dentro do Java?

```java
public boolean updateNPC(NPC npc) {
    connectToDb();
    String sql = "UPDATE NPC SET idade=?, genero=?, ocupacao=?, arcana=? WHERE nome=?";

    try {
        pst = connection.prepareStatement(sql);
        pst.setInt( parameterIndex: 1, npc.getIdade());
        pst.setString( parameterIndex: 2, npc.getGenero());
        pst.setString( parameterIndex: 3, npc.getOcupacao());
        pst.setString( parameterIndex: 4, npc.getArcana());
        pst.setString( parameterIndex: 5, npc.getNome());
        pst.execute();
        return true;

    } catch (SQLException e) {
        System.out.println("Erro ao atualizar NPC: " + e.getMessage());
        return false;

    } finally {
        try {
            if (pst != null) pst.close();
            if (connection != null) connection.close();
        } catch (SQLException e) {
            System.out.println("Erro ao fechar recursos: " + e.getMessage());
        }
    }
}
```

Cláusula Finally

A cláusula finally é essencial para garantir que recursos como conexões de banco de dados sejam sempre liberados, independentemente de ocorrer sucesso ou erro na operação.

Como Remover Dados/Instâncias dentro do Java?

```java
public boolean deleteNPC(String nome) {
    connectToDb();
    String sql = "DELETE FROM NPC WHERE nome=?";

    try {
        pst = connection.prepareStatement(sql);
        pst.setString( parameterIndex: 1, nome);
        pst.execute();
        return true;

    } catch (SQLException e) {
        System.out.println("Erro ao deletar NPC: " + e.getMessage());
        return false;

    } finally {
        try {
            if (pst != null) pst.close();
            if (connection != null) connection.close();
        } catch (SQLException e) {
            System.out.println("Erro ao fechar recursos: " + e.getMessage());
        }
    }
}
```

Como Buscar/Mostrar Dados dentro do Java?

```java
public String selectArcana(String nome) {
    String arcana = null;
    connectToDb();
    String sql = "SELECT arcana FROM NPC WHERE nome=?";

    try {
        pst = connection.prepareStatement(sql);
        pst.setString( parameterIndex: 1, nome);
        rs = pst.executeQuery();
        if (rs.next()) {
            arcana = rs.getString( columnLabel: "arcana");
        }

    } catch (SQLException e) {
        System.out.println("Erro ao buscar arcana do NPC: " + e.getMessage());

    } finally {
        try {
            if (rs != null) rs.close();
            if (pst != null) pst.close();
            if (connection != null) connection.close();
        } catch (SQLException e) {
            System.out.println("Erro ao fechar recursos: " + e.getMessage());
        }
    }
    return arcana;
}
```

```java
public List<NPC> selectNPC() {
    List<NPC> npcs = new ArrayList<>();
    connectToDb();
    String sql = "SELECT * FROM NPC";

    try {
        st = connection.createStatement();
        rs = st.executeQuery(sql);
        while (rs.next()) {
            NPC npc = new NPC(
                    rs.getString( columnLabel: "nome"),
                    rs.getInt( columnLabel: "idade"),
                    rs.getString( columnLabel: "genero"),
                    rs.getString( columnLabel: "ocupacao"),
                    rs.getString( columnLabel: "arcana")
            );
            npcs.add(npc);
        }

    } catch (SQLException e) {
        System.out.println("Erro ao buscar NPCs: " + e.getMessage());

    } finally {
        try {
            if (rs != null) rs.close();
            if (st != null) st.close();
            if (connection != null) connection.close();
        } catch (SQLException e) {
            System.out.println("Erro ao fechar recursos: " + e.getMessage());
        }
    }
    return npcs;
}
```

Como fica o Script SQL?

Criação do Banco de Dados

```sql
DROP DATABASE IF EXISTS Projeto_persona;
CREATE DATABASE IF NOT EXISTS Projeto_persona;
USE Projeto_persona;
```

Criação da Tabela (Cuidado com nome)

```sql
DROP TABLE IF EXISTS npc;
CREATE TABLE IF NOT EXISTS npc (
  idNPC INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
  nome VARCHAR(45),
  idade INT,
  genero VARCHAR(45),
  ocupacao varchar(60),
  arcana varchar(10),
  Protagonista_idProtagonista INT NULL,
  CONSTRAINT fk_NPC_Protagonista FOREIGN KEY (Protagonista_idProtagonista)
    REFERENCES protagonista(idProtagonista) ON DELETE CASCADE
);
```

Uso de Estrutura

```sql
-- Uso de View é muito interessante para consultas complexas:
-- No caso, as informações de um NPC podem ser essenciais para criação de um Menu de mostraInf
-- As informações de NPCS menores do que 20 anos podem ser associadas aos alunos da escola
    -- Informações dos NPCs com idade < 20:
CREATE VIEW view_npcs_menores_20 AS
SELECT
    idNPC,
    nome,
    idade,
    genero
FROM npc
WHERE Idade < 20;

SELECT * FROM view_npcs_menores_20;
```

Mostrar Todas as Informações

```sql
SELECT * FROM npc;
```

Felipe
Tagawa
Reis

python™

Integração Python MySQL

Instituto Nacional de Telecomunicações - INATEL

O que vamos cobrir?

- **Configuração do Ambiente** - Preparação do Ambiente Python
- **Instalação de Dependências** - Conector MySQL
- **Criação do Módulo de Conexão** - Centralização do Acesso ao Banco
- **Manipulação de Dados (CRUD)** - Comandos Básicos

Configuração do Ambiente

- **VirtualEnv**

Por que usar?

Um ambiente virtual isola as dependências do seu projeto, evitando conflitos com outros projetos ou com os pacotes do sistema.

- Isolamento de projetos
- Evita conflitos de versão
- Mantém o sistema limpo

Como criar?

Use o módulo `venv` nativo do Python:

```
python -m venv venv
```

Ativação (Linux/Mac):

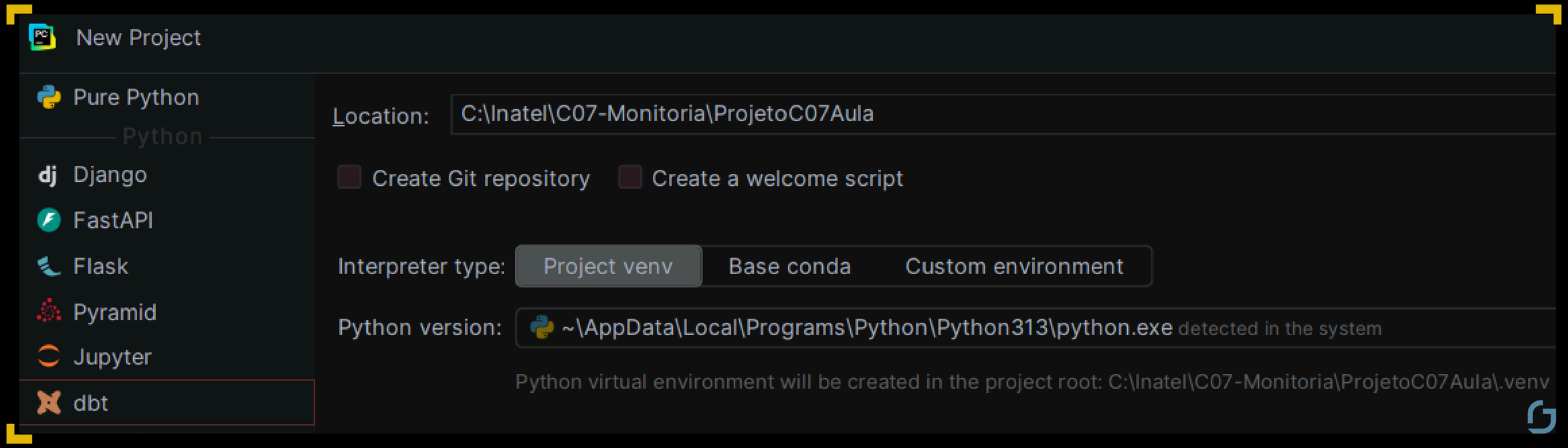
```
source venv/bin/activate
```

Ativação (Windows):

```
venv\Scripts\activate
```

Configuração do Ambiente

- **No Pycharm**

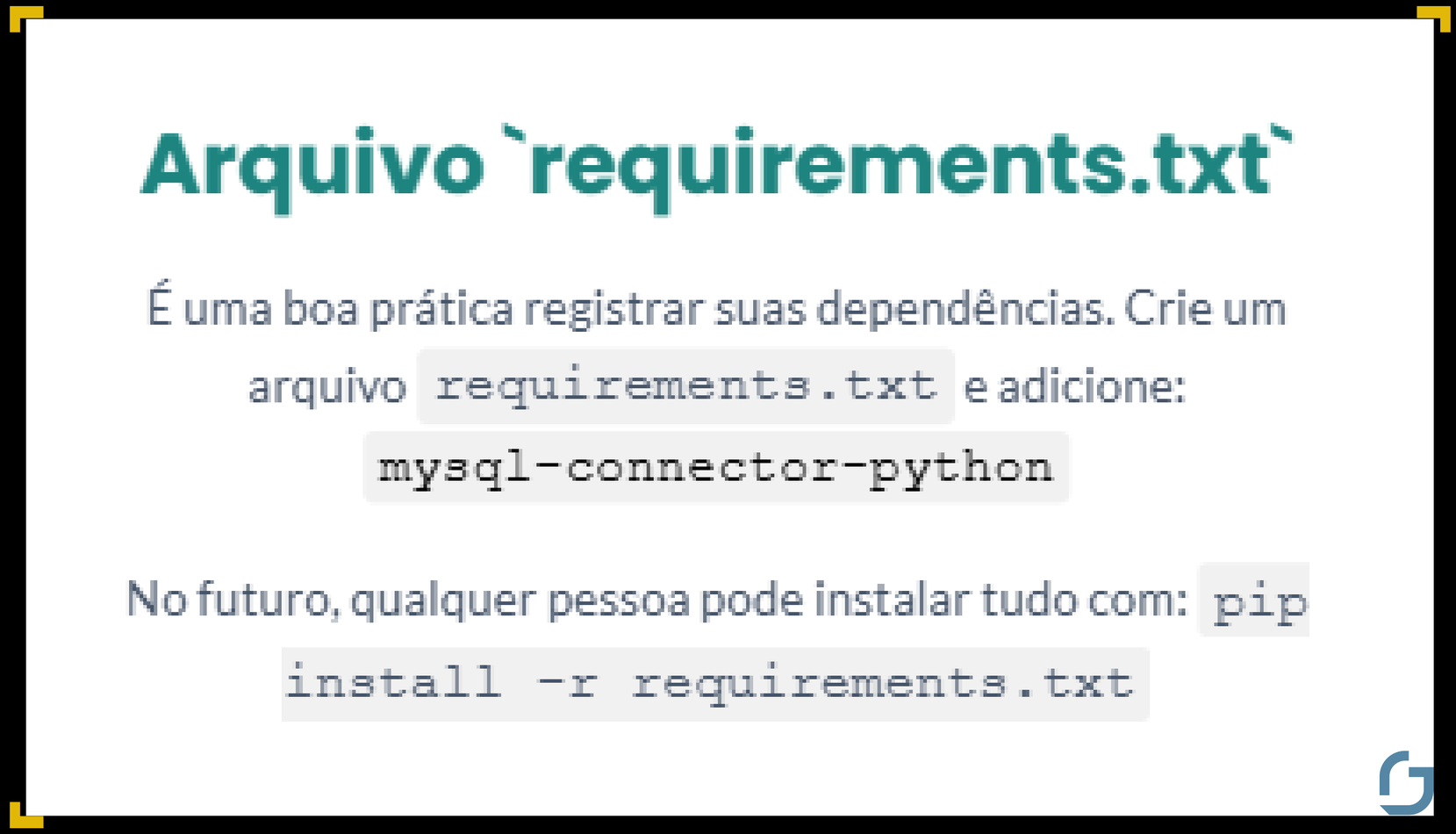


Instalação de Dependências

- **Comando pip**

Instalação de Dependências

- **requirements.txt**



Módulo de Conexão

connection_dao.py

DB

```python
# Módulo responsável por estabelecer e gerenciar
# a conexão com o banco de dados MySQL.

import mysql.connector
from mysql.connector import Error

DB_CONFIG = {
    'host': 'localhost',
    'database': 'seu_banco_de_dados',
    'user': 'seu_usuario',
    'password': 'sua_senha'
}

def get_connection():
    """Tenta estabelecer e retornar uma
    conexão com o MySQL."""
    connection = None
    try:
        connection = mysql.connector
```

```python
def close_connection(connection, cursor=None):
    """Fecha o cursor e a conexão se
    estiverem ativos."""
    try:
        cursor.close() / connection.close()
```

```python
if __name__ == '__main__':
    conn = get_connection()
    if conn:
        close_connection(conn)
```

Manipulação de Dados

- **Ocorre dentro das classes DAO**
- **Visualização no Pycharm**

Aula Concluída