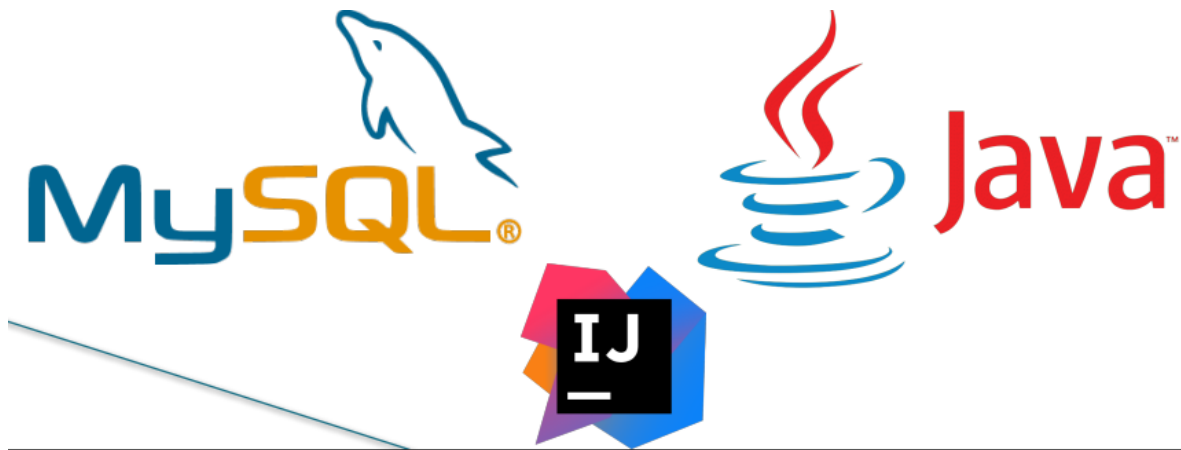


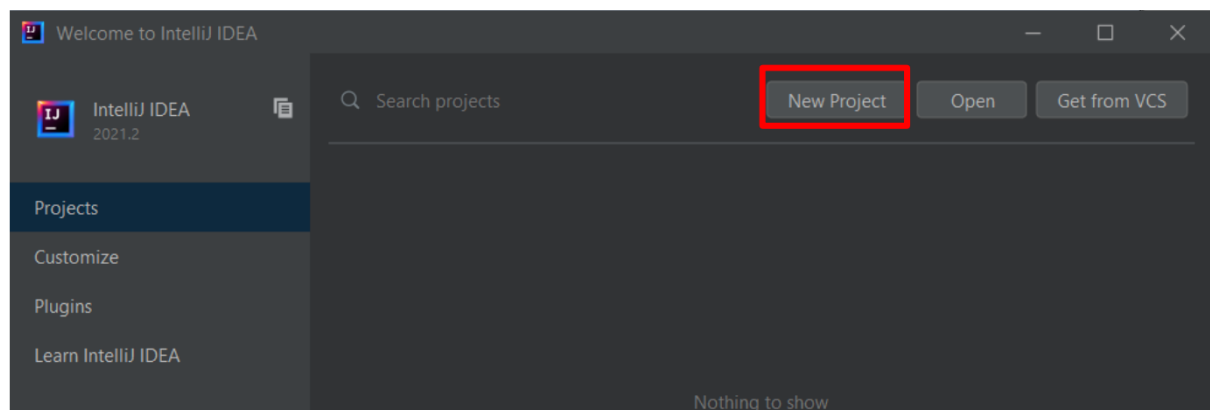
Aula 7 - Integração entre MySQL e Java

🕒 Created	@February 2, 2022 10:22 AM
▼ Class	
▼ Type	
🔗 Materials	
☑ Reviewed	<input type="checkbox"/>
☰ Property	



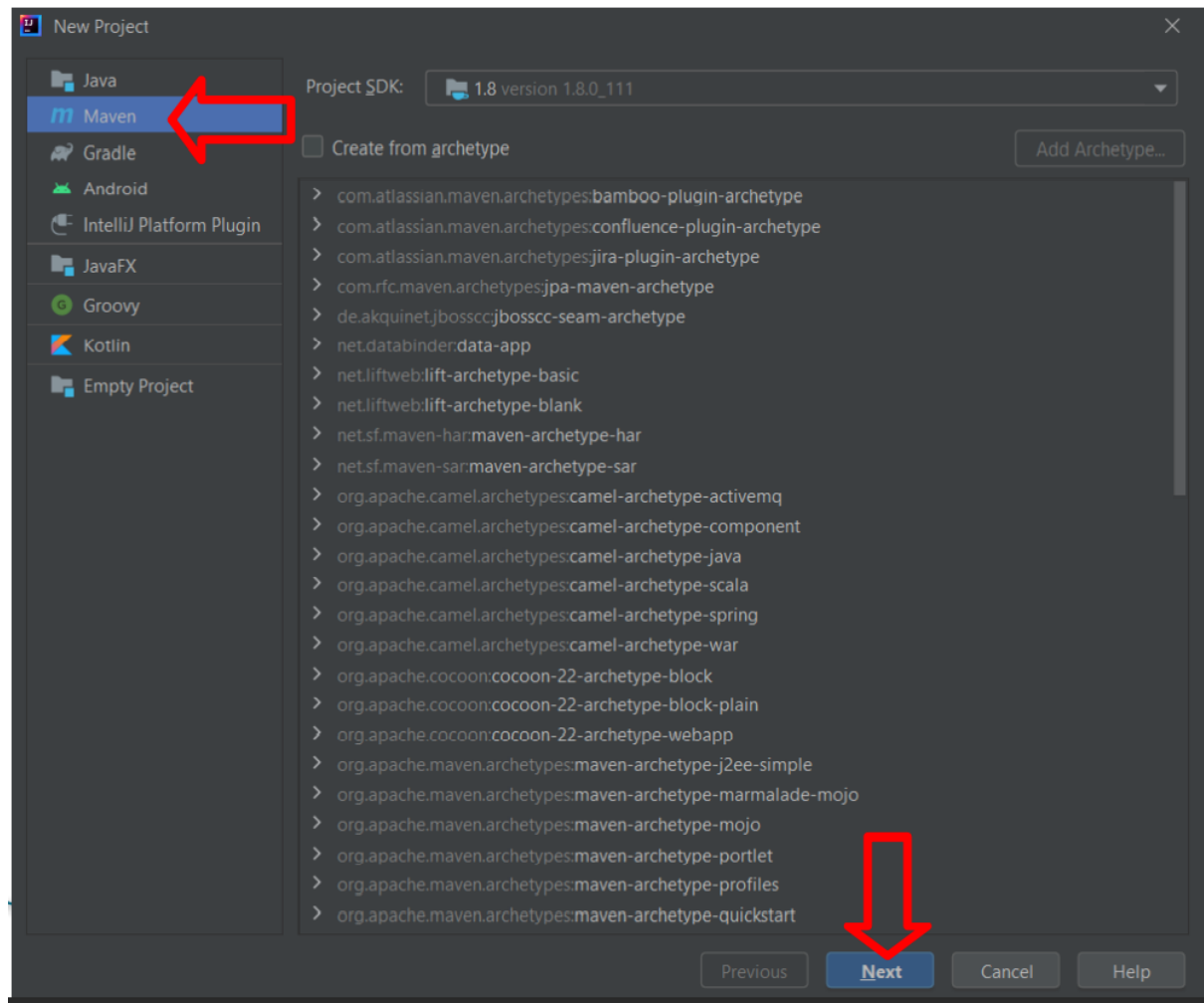
Criando novo projeto no IntelliJ

Ao iniciar o IntelliJ, se não tiver criado nenhum projeto anteriormente, a seguinte tela será exibida:

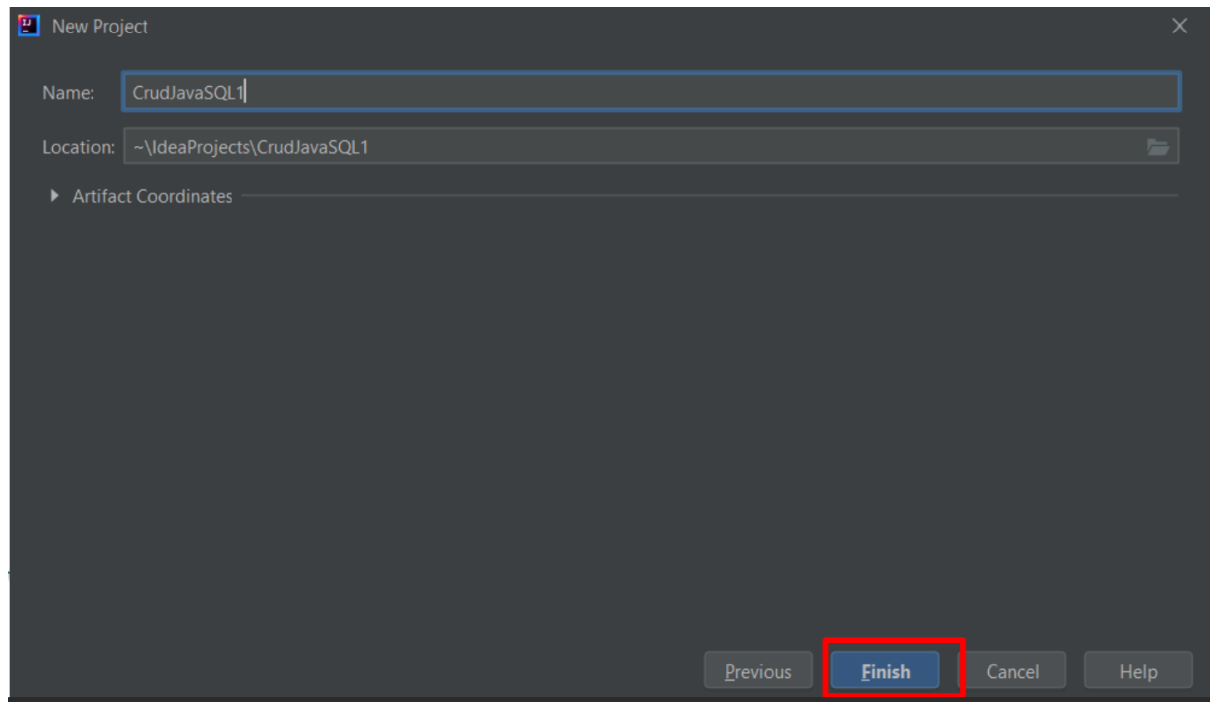


Clique em “New Project” para criar um novo projeto. Se tiver com projeto já aberto, vá em “File” → “Close Project”, será exibida a janela acima.

Escolha a opção Maven, a versão do Java e em seguida clique em “Next”:

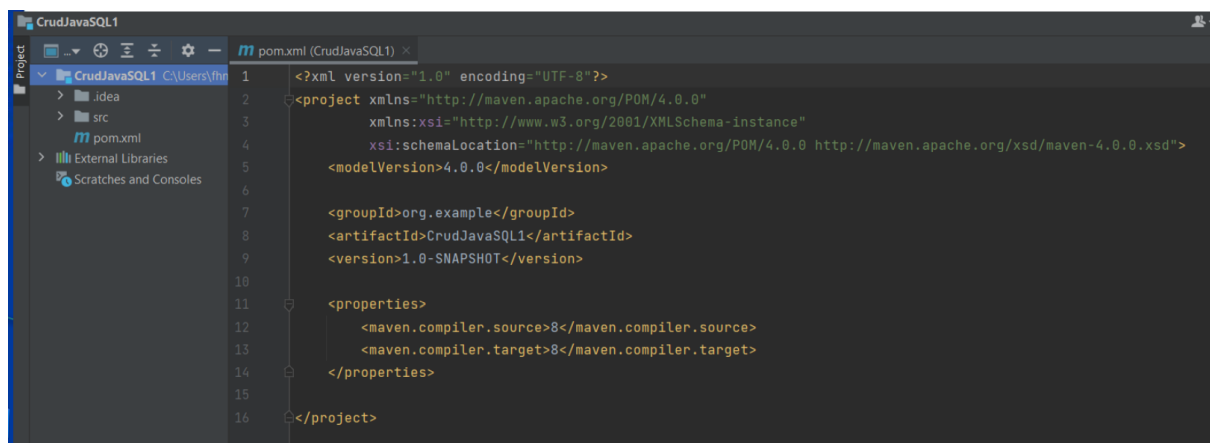


Nomeie seu projeto e clique em “Finish”:



Quando for finalizado a criação do projeto, o IntelliJ irá abrir o pom.xml, ele é o arquivo onde o Maven (Gerenciador do projeto Java) instala todas as dependências do projeto de maneira automática.

- O Maven é semelhante ao npm do JavaScript e o pip do Python;
- O pom.xml é um arquivo semelhante package.json do JavaScript ou o pubspec.yaml do Flutter;

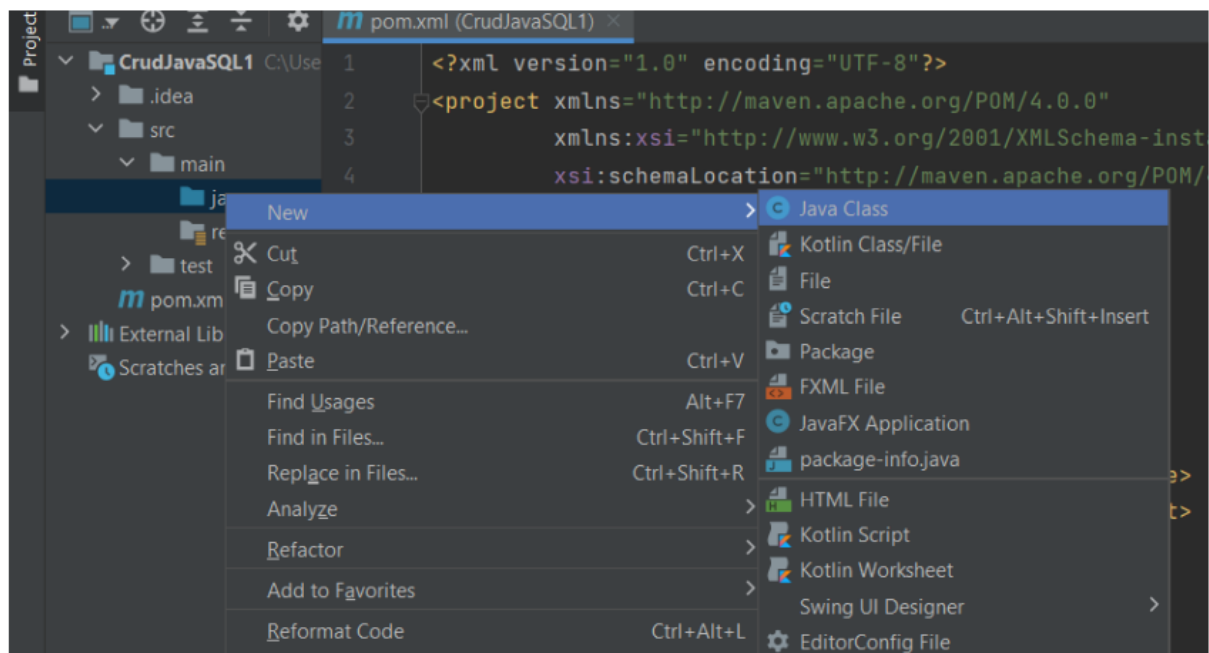


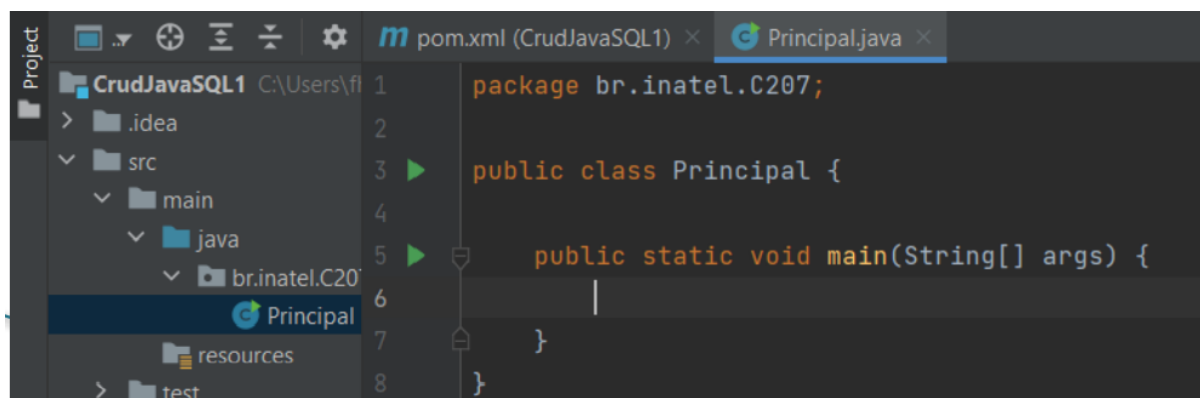
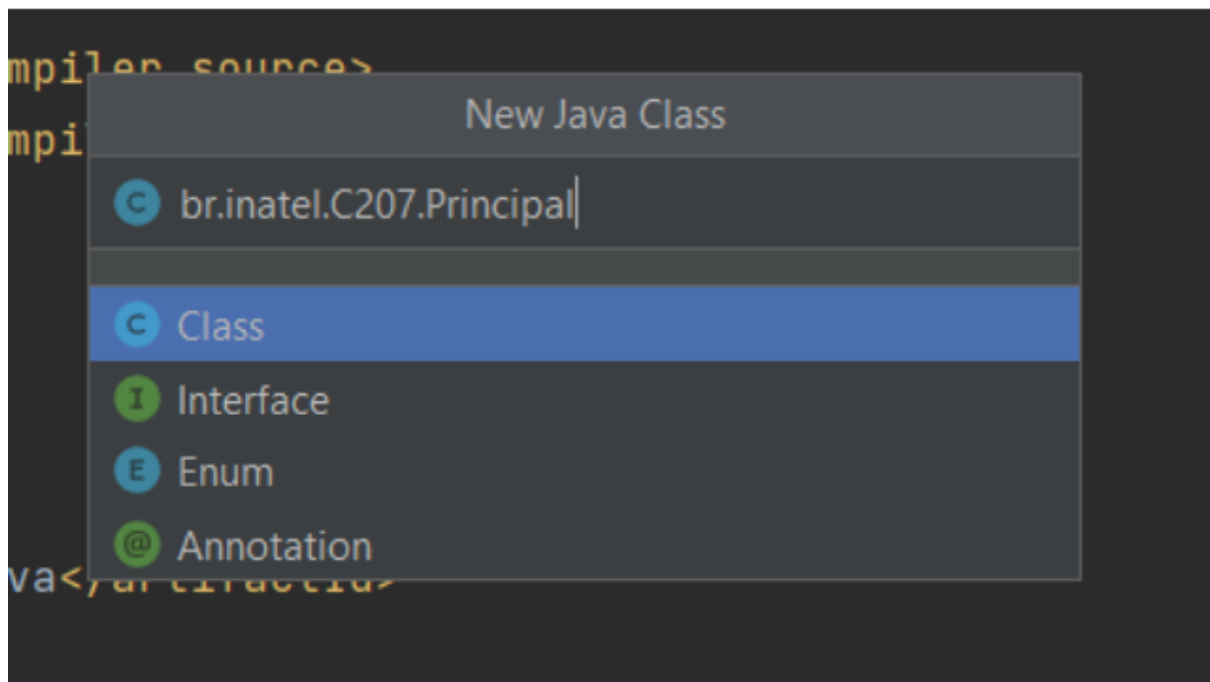
Vamos instalar a dependência do MySQL: Pra isso escreva “dependencies” e dentro dela, escreva “mysql”, o intellisense ajuda muito!



Criando Classe Principal

Dentro da pasta "java" em "main", clique com o botão direito, vá em "New" → "Java Class". Será aberto uma janela onde colocará o nome do pacote e da classe. Após pressionar "Enter", será criada a classe e deverá escrever o método main.





Qual banco vamos trabalhar?

Vamos criar um banco simples no Workbench para podermos realizar um CRUD. Ele terá somente uma tabela com três atributos.

De modo análogo a criação da classe principal, vamos criar uma classe User para se receber ou enviar registros vindos do banco.

```
DROP DATABASE IF EXISTS projeto;  
CREATE DATABASE projeto;  
USE projeto;
```

```
CREATE TABLE usuario(  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(45) NOT NULL,  
    cpf VARCHAR(14) NOT NULL  
);
```

```
package br.inatel.C207;

public class User {
    private String nome;
    private String cpf;
    public int id;

    public User(String nome, String cpf){
        this.nome = nome;
        this.cpf = cpf;
    }

    public String getNome(){
        return this.nome;
    }

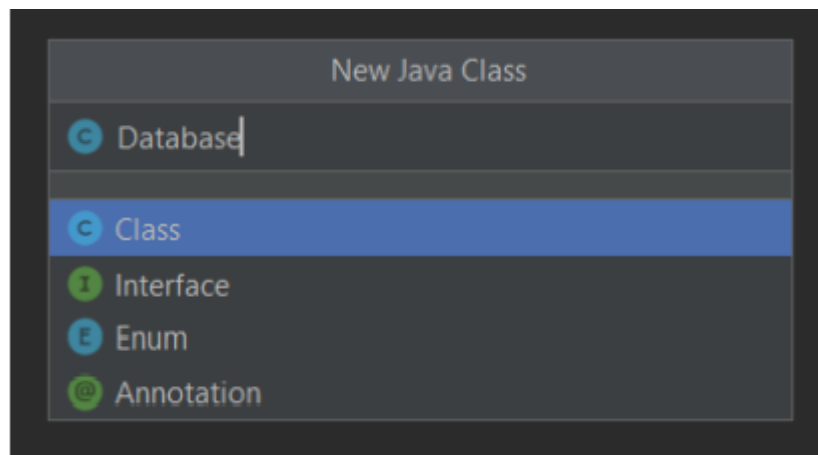
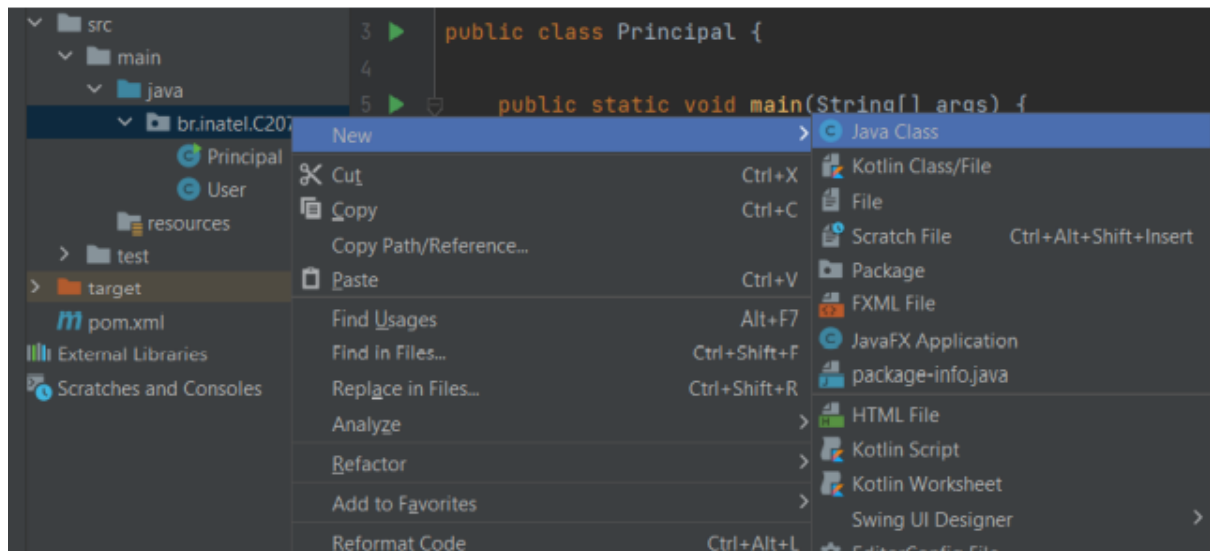
    public String getCpf(){
        return this.cpf;
    }

    public void setNome(String nome){
        this.nome = nome;
    }

    public void setCpf(String cpf){
        this.cpf = cpf;
    }
}
```

Criando a classe do banco

De maneira semelhante, crie também uma classe chamada Database, pode ser no mesmo pacote, ou em pacotes distintos, como por exemplo, respeitando o padrão MVC (na disciplina de POO será melhor explicado sobre esse padrão).



Dentro dessa classe, iremos colocar todos os métodos de conexão e CRUD com o MySQL. Para essa primeira parte, definimos os objetos necessários para trabalhar com os comandos SQL, as credenciais de usuário, o nome do banco e também a URL de conexão, sendo:


```
jdbc:mysql://localhost:3306/" + database +  
"?useTimezone=true&serverTimezone=UTC&useSSL=false
```

Dependendo a versão da sua JDK, pode usar essa URL:

```
jdbc:mysql://localhost:3306/" + database
```

```
5 public class Database {  
6     Connection connection; // objeto responsável por fazer a conexão com o servidor do MySQL  
7     Statement statement; // objeto responsável por preparar consultas "SELECT"  
8     ResultSet result; // objeto responsável por executar consultas "SELECT"  
9     PreparedStatement pst; // objeto responsável por preparar queries de manipulação dinâmicas (INSERT, UPDATE, DELETE)  
10  
11     static final String user = "root"; // usuário da instância local do servidor  
12     static final String password = "/MS-DOSV.6.22b"; // senha do usuário da instância local do servidor  
13     static final String database = "projeto"; // nome do banco de dados a ser utilizado  
14  
15     // string com URL de conexão com servidor  
16     static final String url = "jdbc:mysql://localhost:3306/" + database + "?useTimezone=true&serverTimezone=UTC&useSSL=false";  
17     private boolean check = false;
```

Esse método dentro da classe Database realiza a conexão com o banco já tratando erros. Caso dê erro ou não, será mostrado um log visual pelo terminal.

```
19 public void connect(){  
20     try {  
21         connection = DriverManager.getConnection(url, user, password);  
22         System.out.println("Conexão feita com sucesso: " + connection);  
23     } catch (SQLException e){  
24         System.out.println("Erro de conexão: " + e.getMessage());  
25     }  
26 }
```

Criando Método de Inserção

```

28 // -----INSERINDO NOVO REGISTRO-----
29 @ public boolean insertUser(User user){
30     connect();
31     String sql = "INSERT INTO usuario(nome, cpf) VALUES(?, ?)";
32     try {
33
34         pst = connection.prepareStatement(sql);
35         pst.setString(1, user.getNome()); // concatena nome na primeira ? do comando
36         pst.setString(2, user.getCpf()); // concatena nome na segunda ? do comando
37         pst.execute(); // executa o comando
38         check = true;
39
40     } catch (SQLException e) {
41         System.out.println("Erro de operação: " + e.getMessage());
42         check = false;
43     }
44     finally {
45         try{
46             connection.close();
47             pst.close();
48         }catch (SQLException e){
49             System.out.println("Erro ao fechar a conexão: " + e.getMessage());
50         }
51     }
52     return check;
53 }

```

Criando Método de Busca

```

56 // -----BUSCANDO TODOS REGISTROS-----
57 public ArrayList<User> researchUser(){
58     connect();
59     ArrayList<User> users = new ArrayList<>();
60     String sql = "SELECT * FROM usuario";
61     try{
62         statement = connection.createStatement();
63         result = statement.executeQuery(sql);
64
65         while(result.next()){
66             User userTemp = new User(result.getString(s: "nome"), result.getString(s: "cpf"));
67             userTemp.id = result.getInt(s: "id");
68             System.out.println("ID = " + userTemp.id);
69             System.out.println("Nome = " + userTemp.getNome());
70             System.out.println("CPF = " + userTemp.getCpf());
71             System.out.println("-----");
72             users.add(userTemp);
73         }
74     }catch (SQLException e){
75         System.out.println("Erro de operação: " + e.getMessage());
76     }finally {
77         try {
78             connection.close();
79             statement.close();
80             result.close();
81         }catch (SQLException e){
82             System.out.println("Erro ao fechar a conexão: " + e.getMessage());
83         }
84     }
85     return users;
86 }

```

Criando Método de Atualização

```

88 // -----ATUALIZANDO NOME NO REGISTRO-----
89 public boolean updateUser(int id, String nome){
90     connect();
91     String sql = "UPDATE usuario SET nome=? WHERE id=?";
92     try{
93         pst = connection.prepareStatement(sql);
94         pst.setString(1, nome);
95         pst.setInt(2, id);
96         pst.execute();
97         check = true;
98     }catch (SQLException e){
99         System.out.println("Erro de operação: " + e.getMessage());
100         check = false;
101     }finally {
102         try {
103             connection.close();
104             pst.close();
105         }catch (SQLException e) {
106             System.out.println("Erro ao fechar a conexão: " + e.getMessage());
107         }
108     }
109     return check;
110 }

```

Criando Método de Exclusão

```

112 // -----EXCLUINDO REGISTRO-----
113 public boolean deleteUser(int id) {
114     connect();
115     String sql = "DELETE FROM usuario WHERE id=?";
116     try{
117         pst = connection.prepareStatement(sql);
118         pst.setInt(1, id);
119         pst.execute();
120         check = true;
121     }catch (SQLException e){
122         System.out.println("Erro de operação: " + e.getMessage());
123         check = false;
124     }finally {
125         try {
126             connection.close();
127             pst.close();
128         }catch (SQLException e){
129             System.out.println("Erro ao fechar a conexão: " + e.getMessage());
130         }
131     }
132     return check;
133 }
134 }

```

Fazendo o CRUD na Main

Chamando todos os métodos na classe principal e realizando testes

```
5  ▶ public static void main(String[] args) {
6      Database database = new Database();
7      database.connect();
8      // criando objetos de usuário para serem inseridos no banco
9      User user = new User( nome: "Flávio", cpf: "111.111.111-11");
10     User user1 = new User( nome: "Fernando", cpf: "222.222.222-22");
11     User user2 = new User( nome: "Vânia", cpf: "333.333.333-33");
12     // Inserindo usuários no banco
13     database.insertUser(user);
14     database.insertUser(user1);
15     database.insertUser(user2);
16     //mostrando todos os usuarios
17     database.researchUser();
18     System.out.println("-----Atualizando nome-----");
19     database.updateUser( id: 1, nome: "Flavinho");
20
21     database.researchUser();
22
23     System.out.println("-----Excluindo usuário-----");
24     database.deleteUser( id: 1);
25     database.researchUser();
26 }
```

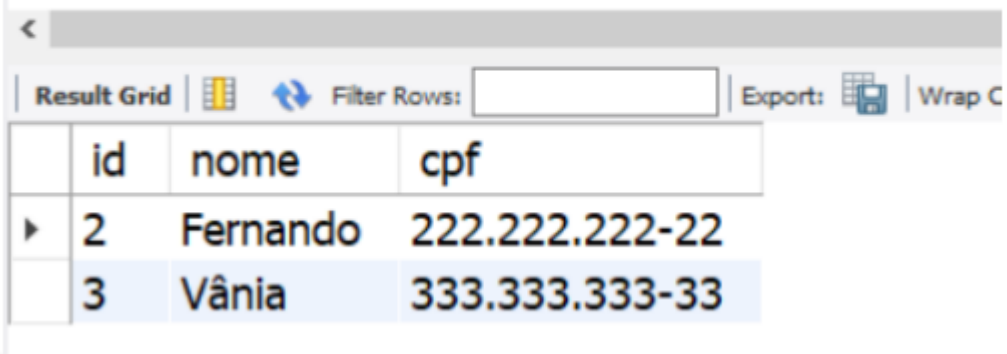
Resultados Finais

Observando o estado final tanto pela aplicação quanto pelo banco diretamente

```
Conexão feita com sucesso: com.mysql.cj.jdbc.ConnectionImpl@46d56d67
ID = 2
Nome = Fernando
CPF = 222.222.222-22
-----
ID = 3
Nome = Vânia
CPF = 333.333.333-33
-----

Process finished with exit code 0
```

11 • **SELECT * FROM usuario;**



	id	nome	cpf
▶	2	Fernando	222.222.222-22
	3	Vânia	333.333.333-33