

Classe DAO e o CRUD no JAVA

Professor Isaac

Exemplo de Inserindo dados no Banco

DML

- A **DML** (Data Manipulation Language) contém os comandos relacionados com a manipulação dos dados contidos pela estrutura do banco de dados, ou seja, pelas operações de inclusão, atualização, remoção e consulta no BD.
- Esse conjunto de operações também é conhecido por **CRUD**.

CRUD

- **CRUD** (acrónimo de Create, Read, Update e Delete na língua Inglesa) para as quatro operações básicas utilizadas em bases de dados relacionais ou em interface para utilizadores para criação, consulta, atualização e destruição de dados.
- A abreviação CRUD mapeada para o padrão ISO/SQL:
 - **Create** **INSERT**
 - **Read (Retrieve)** **SELECT**
 - **Update** **UPDATE**
 - **Delete** **DELETE**

DAO

- O Padrão **DAO** (Data Access Object) se refere a classe que irá fazer a comunicação de seu programa com o banco de dados (classe onde será implementado o CRUD).
- Esse padrão é muito importante. Ele promove a reutilização de código, também promove a boa manutenção do código pois ele não deixa que o código de acesso ao banco de dados fique misturado com outras partes da aplicação que possuem outros objetivos.

EXEMPLO

- Criar uma classe DAO para inserir, atualizar, remover e consultar no BD.

Classe conexão

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Conexao {
    private Connection conn;

    public Connection conectar() {
        try {
            // Informando qual driver de conexão será utilizado pelo DriverManager
            Class.forName("org.postgresql.Driver");
            // Criando a conexão com o BD
            String url = "jdbc:postgresql://localhost:5432/testeJava";
            String username = "postgres";
            String password = "123456";
            conn = DriverManager.getConnection(url, username, password);
            System.out.println("Conectado com Sucesso");
            return conn;
        } catch (ClassNotFoundException | SQLException e) {
            System.err.println("Erro ao conectar: " + e.getMessage());
            return null;
        }
    }

    public void desconectar() {
        try {
            if (conn != null && !conn.isClosed()) {
                // Desconectando do BD
                conn.close();
            }
        } catch (SQLException e) {
        }
    }
}
```

EXEMPLO

```
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class DAO {
    private Connection conn;
    private PreparedStatement pstmt;
    private ResultSet rs;
```


Método Inserir - (INSERT)

```
public void inserir(int numero, String nomeDepartamento, String local) {  
    // Abrindo a conexão com o banco  
    Conexao conexao = new Conexao();  
    conn = conexao.conectar();  
    try {  
        // Instanciando o objeto preparedStatement (pstmt)  
        pstmt = conn.prepareStatement("INSERT INTO DEPT (DEPTNO, DNAME, LOCAL) VALUES (?, ?, ?)");  
        // Setando o valor aos parâmetros  
        pstmt.setInt(1, numero);  
        pstmt.setString(2, nomeDepartamento);  
        pstmt.setString(3, local);  
        // Executando o comando sql do objeto preparedStatement  
        pstmt.execute();  
        System.out.println("Inserido com Sucesso");  
        conexao.desconectar(); // Fechando a conexão com o banco  
    } catch (SQLException e) {  
        // Fechando a conexão com o banco  
        conexao.desconectar();  
        System.err.println("Falha em Inserir no DB" + e.getMessage());  
    }  
}
```

Executando o método inserir

```
1 package simpledb;
2
3 public class Principal {
4     public static void main(String[] args) {
5         // TODO code application logic here
6         DAO crud = new DAO();
7         crud.inserir(100, "ENGENHARIA", "SAO PAULO - SP");
8         crud.inserir(80, "IT", "SAO PAULO");
9         crud.inserir(10, "management", "SAO PAULO - SP");
10    }
11 }
12
```

Output X

SQL 1 execution X SimpleDB_postgre (run) X

run:

Conectado com Sucesso
Inserido com Sucesso
Conectado com Sucesso
Inserido com Sucesso
Conectado com Sucesso
Inserido com Sucesso
BUILD SUCCESSFUL (total time: 1 second)

Verificando o BD

The screenshot shows the pgAdmin 4 web interface in a browser. The address bar indicates the connection to 127.0.0.1:50982/browser/. The left sidebar displays the database structure, with the 'public' schema expanded and the 'dept' table selected. The main panel shows the 'Query Editor' with the SQL query: `SELECT * FROM public.dept`. Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format. The table has four columns: 'deptno', 'dname', and 'local'. The results show three rows of data. A green notification box at the bottom right indicates that the query was successfully run, with a total runtime of 58 msec and 3 rows affected.

pgAdmin 4

127.0.0.1:50982/browser/

pgAdmin File Object Tools Help

Browser

Schemas (1)

- public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (2)
 - agenda
 - dept
 - Trigger Functions
 - Types
 - Views
- Login/Group Roles
- Tablespaces

Query Editor Query History

```
1 SELECT * FROM public.dept
2
```

Data Output Explain Messages Notifications

| | deptno | dname | local |
|---|--------|------------|----------------|
| 1 | 100 | ENGENHARIA | SAO PAULO - SP |
| 2 | 80 | IT | SAO PAULO |
| 3 | 10 | management | SAO PAULO - SP |

Successfully run. Total query runtime: 58 msec. 3 rows affected.

Método alterar - (UPDATE)

```
public void alterar(int numero, String nomeDepartamento) {  
    // Abrindo a conexão com o banco  
    Conexao conexao = new Conexao();  
    conn = conexao.conectar();  
    try {  
        // Instanciando o objeto preparedStatement (pstmt)  
        pstmt = conn.prepareStatement("UPDATE DEPT SET DNAME = ? WHERE DEPTNO = ?");  
        // Setando o valor ao parâmetro  
        pstmt.setString(1, nomeDepartamento);  
        pstmt.setInt(2, numero);  
        // Executando o comando sql do objeto preparedStatement  
        pstmt.execute();  
        System.out.println("Alterado com Sucesso");  
        // Fechando a conexão com o banco  
        conexao.desconectar();  
    } catch (SQLException e) {  
        // Fechando a conexão com o banco  
        conexao.desconectar();  
    }  
}
```

Executando o método alterar

```
1 package simpledb;
2
3 public class Principal {
4     public static void main(String[] args) {
5         // TODO code application logic here
6         DAO crud = new DAO();
7         crud.alterar(80, "TI");
8     }
9 }
10
```

Output ×

SQL 1 execution × SimpleDB_postgre (run) ×

run:
Conectado com Sucesso
Alterado com Sucesso
BUILD SUCCESSFUL (total time: 0 seconds)

Verificando o BD

The screenshot shows the pgAdmin 4 web interface in a browser. The address bar shows the URL `127.0.0.1:50982/browser/`. The interface includes a left sidebar with a tree view of the database structure, a top menu bar, and a main workspace. The workspace is divided into a 'Query Editor' and a 'Data Output' section.

Database Structure (Left Sidebar):

- Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (2)
 - agenda
 - dept
 - Trigger Functions
 - Types
 - Views
 - Login/Group Roles
 - Tablespaces

Query Editor:

```
1 SELECT * FROM public.dept
2
```

Data Output:

| deptno | dname | local |
|--------|----------------|----------------|
| 1 | 100 ENGENHARIA | SAO PAULO - SP |
| 2 | 10 management | SAO PAULO - SP |
| 3 | 80 TI | SAO PAULO |

Método remover - (DELETE)

```
public void remover(int deptno) {  
    // Abrindo a conexão com o banco  
    Conexao conexao = new Conexao();  
    conn = conexao.conectar();  
    try {  
        // Instanciando o objeto preparedStatement (pstmt)  
        String remover = "DELETE FROM DEPT WHERE DEPTNO = ?";  
        // Instanciando o objeto preparedStatement (pstmt)  
        pstmt = conn.prepareStatement(remover);  
        // Setando o valor ao parâmetro  
        pstmt.setLong(1, deptno);  
        // Executando o comando sql do objeto preparedStatement  
        pstmt.execute();  
        System.out.println("Removido com Sucesso");  
        conexao.desconectar(); // Fechando a conexão com o banco  
    } catch (SQLException e) {  
        conexao.desconectar(); // Fechando a conexão com o banco  
    }  
}
```

Executando o método remover()

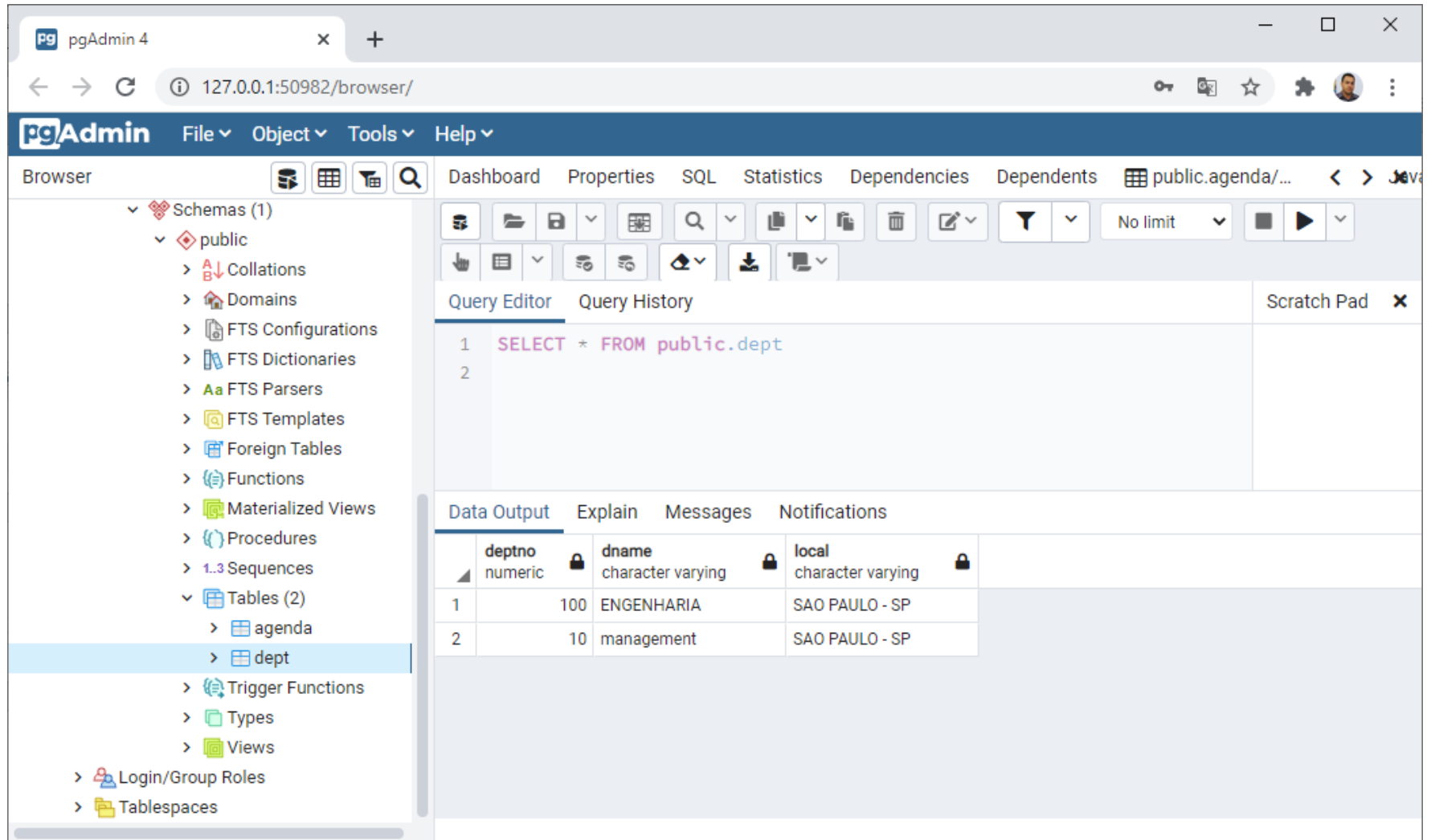
```
1  package simpledb;
2
3  public class Principal {
4      public static void main(String[] args) {
5          // TODO code application logic here
6          DAO crud = new DAO();
7          crud.remover(80);
8      }
9  }
```

Output ×

SQL 1 execution × SimpleDB_postgre (run) ×

run:
Conectado com Sucesso
Removido com Sucesso
BUILD SUCCESSFUL (total time: 1 second)

Verificando o BD



pgAdmin 4

127.0.0.1:50982/browser/

pgAdmin File Object Tools Help

Browser

- Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - 1.3 Sequences
 - Tables (2)
 - agenda
 - dept
 - Trigger Functions
 - Types
 - Views
 - Login/Group Roles
 - Tablespaces

Dashboard Properties SQL Statistics Dependencies Dependents public.agenda/...

Query Editor Query History Scratch Pad

```
1 SELECT * FROM public.dept
2
```

Data Output Explain Messages Notifications

| | deptno | | dname | | local |
|---|---------|--|-------------------|--|-------------------|
| | numeric | | character varying | | character varying |
| 1 | 100 | | ENGENHARIA | | SAO PAULO - SP |
| 2 | 10 | | management | | SAO PAULO - SP |

Método buscar - (SELECT)

```
public ResultSet buscar() {  
    // Abrindo a conexão com o banco  
    Conexao conexao = new Conexao();  
    conn = conexao.conectar();  
    try {  
        // Instanciando o objeto preparedStatement (pstmt)  
        pstmt = conn.prepareStatement("SELECT * FROM DEPT ORDER BY DEPTNO");  
        // Executando o comando sql e armazenando no ResultSet  
        rs = pstmt.executeQuery();  
        //Retornando o ResultSet  
        return rs;  
    } catch (SQLException e) {  
        conexao.desconectar();  
        return null;  
    }  
}
```

Executando o método buscar()

```
1 package simpledb;
2
3 import java.sql.SQLException;
4 import java.sql.ResultSet;
5
6 public class Principal {
7     public static void main(String[] args) {
8         DAO crud = new DAO();
9         try {
10             ResultSet rs = crud.buscar();
11             while (rs.next()) {
12                 String lista = ("No do Depto: " + rs.getInt("DEPTNO") + " - " + "Nome: "
13                     + rs.getString("DNAME") + " - " + "Localização: " + rs.getString("LOCAL"));
14                 System.out.println(lista);
15             }
16         } catch (SQLException e) {
17             System.out.println("Consulta não foi possível" + e.getMessage());
18         }
19     }
20 }
```

Output ×

SQL 1 execution × SimpleDB_postgre (run) ×



run:

Conectado com Sucesso

No do Depto: 10 - Nome: management - Localização: SAO PAULO - SP

No do Depto: 100 - Nome: ENGENHARIA - Localização: SAO PAULO - SP

BUILD SUCCESSFUL (total time: 0 seconds)

Método buscar por nome - (SELECT)

```
public ResultSet buscarPorNome(String nome) {  
    // Abrindo a conexão com o banco  
    Conexao conexao = new Conexao();  
    conn = conexao.conectar();  
    try {  
        // Instanciando o objeto preparedStatement (pstmt)  
        pstmt = conn.prepareStatement("SELECT * FROM DEPT WHERE DNAME = ? ORDER BY DEPTNO");  
        pstmt.setString(1, nome);  
        // Executando o comando sql e armazenando no ResultSet  
        rs = pstmt.executeQuery();  
        //Retornando o ResultSet  
        return rs;  
    } catch (SQLException e) {  
        conexao.desconectar();  
        return null;  
    }  
}
```

Executando o método buscarPorNome()

```
1  package simpledb;
2
3  import java.sql.SQLException;
4  import java.sql.ResultSet;
5
6  public class Principal {
7      public static void main(String[] args) {
8          DAO crud = new DAO();
9          try {
10             ResultSet rs = crud.buscarPorNome("ENGENHARIA");
11             while (rs.next()) {
12                 String lista = ("No do Depto: " + rs.getInt("DEPTNO") + " - " + "Nome: "
13                     + rs.getString("DNAME") + " - " + "Localização: " + rs.getString("LOCAL"));
14                 System.out.println(lista);
15             }
16         } catch (SQLException e) {
17             System.out.println("Consulta não foi possível");
18         }
19     }
20 }
```

Output ×

SQL 1 execution × SimpleDB_postgre (run) ×

run:

Conectado com Sucesso

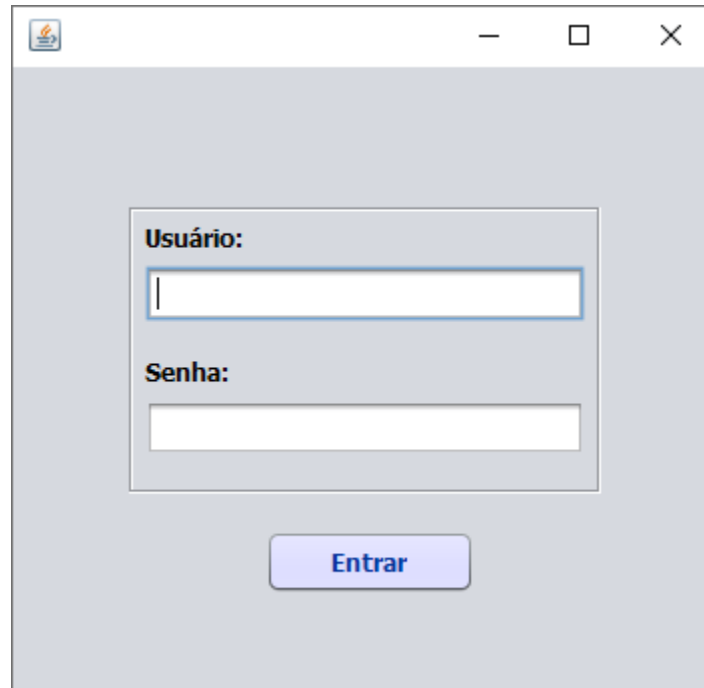
No do Depto: 100 - Nome: ENGENHARIA - Localização: SAO PAULO - SP

BUILD SUCCESSFUL (total time: 1 second)

Exemplo 02

EXEMPLO 2

- Criando uma Tela de Login e Senha.



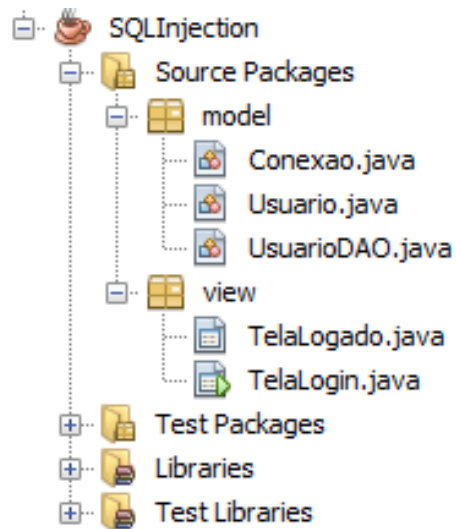
Usuário:

Senha:

Entrar

EXEMPLO 2

- Criar uma classe DAO buscar o login e senha para autenticação.



Classe conexão

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Conexao {
    private Connection conn;

    public Connection conectar() {
        try {
            // Informando qual driver de conexão será utilizado pelo DriverManager
            Class.forName("org.postgresql.Driver");
            // Criando a conexão com o BD
            String url = "jdbc:postgresql://localhost:5432/testeJava";
            String username = "postgres";
            String password = "123456";
            conn = DriverManager.getConnection(url, username, password);
            System.out.println("Conectado com Sucesso");
            return conn;
        } catch (ClassNotFoundException | SQLException e) {
            System.err.println("Erro ao conectar: " + e.getMessage());
            return null;
        }
    }

    public void desconectar() {
        try {
            if (conn != null && !conn.isClosed()) {
                // Desconectando do BD
                conn.close();
            }
        } catch (SQLException e) {
        }
    }
}
```

Classe Usuário

```
package model;

public class Usuario {
    private final String login;
    private final String senha;

    public Usuario(String login, String senha) {
        this.login = login;
        this.senha = senha;
    }

    public String getLogin() {
        return login;
    }

    public String getSenha() {
        return senha;
    }
}
```

Classe UsuárioDAO

```
package model;

import java.sql.SQLException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class UsuarioDAO {
    private final Connection conn;
    private PreparedStatement pstmt;
    private ResultSet rs;

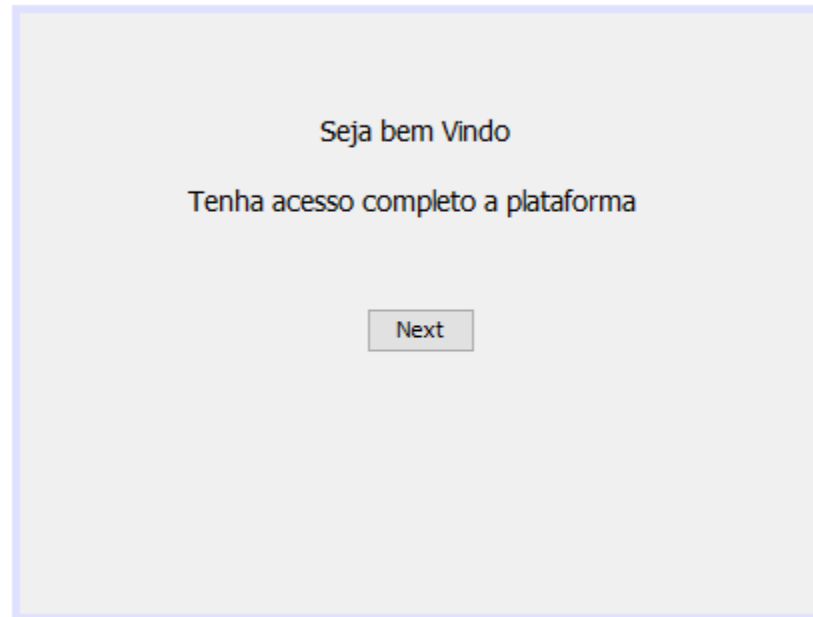
    public UsuarioDAO() {
        Conexao conexao = new Conexao();
        conn = conexao.conectar();
    }

    public boolean verificaLogin(Usuario usuario){
        String sql = "select * from usuarios where usuario = '" + usuario.getLogin()
            + "' and senha = '" + usuario.getSenha() + "'";

        try {
            pstmt = conn.prepareStatement(sql);
            // Executando o comando sql e armazenando no ResultSet
            rs = pstmt.executeQuery();
            //Retornando o ResultSet
            while(rs.next()){
                return true;
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
        return false;
    }
}
```

Tela Logado

- Não há nenhuma implementação, a Tela será apenas chamada quando o login e senha estiver correto.



Tela Login

- Se o login e senha estiver correto a Tela Logado será executada.
- Senão aparece uma mensagem de falha de login

```
private void btnEntrarActionPerformed(java.awt.event.ActionEvent evt) {  
    UsuarioDAO dao = new UsuarioDAO();  
  
    boolean encontrou = dao.verificaLogin(new Usuario(txtUsuario.getText(), txtSenha.getText()));  
  
    if(encontrou){  
        TelaLogado tela = new TelaLogado();  
        tela.setVisible(true);  
    }  
    else{  
        JOptionPane.showMessageDialog(null, "Login ou senha incorreto", "Falha no Login", JOptionPane.ERROR_MESSAGE);  
    }  
}
```

Banco de Dados

- No banco de Dados há dois usuários

The screenshot shows the pgAdmin 4 web interface in a browser. The address bar indicates the URL is `127.0.0.1:50982/browser/`. The interface includes a sidebar with a tree view of the database structure, a central query editor, and a results pane at the bottom.

Database Structure (Left Sidebar):

- Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - 1.3 Sequences
 - Tables (3)
 - agenda
 - dept
 - usuarios**
 - Trigger Functions
 - Types
 - Views
 - Login/Group Roles

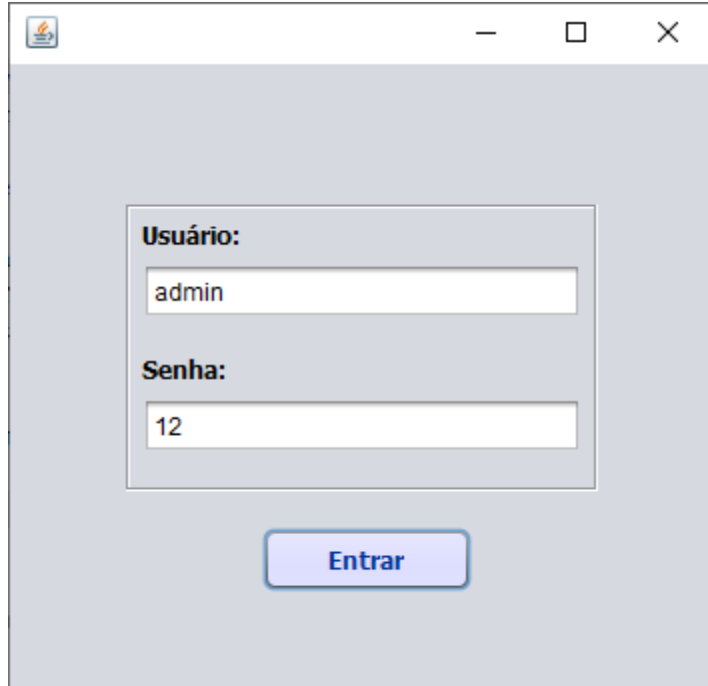
Query Editor (Center):

```
1 SELECT * FROM public.usuarios
2 ORDER BY usuario ASC
```

Data Output (Bottom):

| | usuario [PK] character varying | senha character varying |
|---|-----------------------------------|----------------------------|
| 1 | admin | 123456 |
| 2 | fulano | fulano1234 |

Teste com login ou senha incorreta

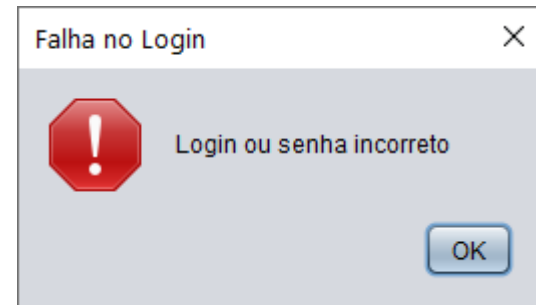


A screenshot of a login window with a light gray background. It features a white rectangular box containing two input fields. The first field is labeled "Usuário:" and contains the text "admin". The second field is labeled "Senha:" and contains the text "12". Below the input fields is a blue button with the text "Entrar". The window has a standard title bar with a minimize button, a maximize button, and a close button.

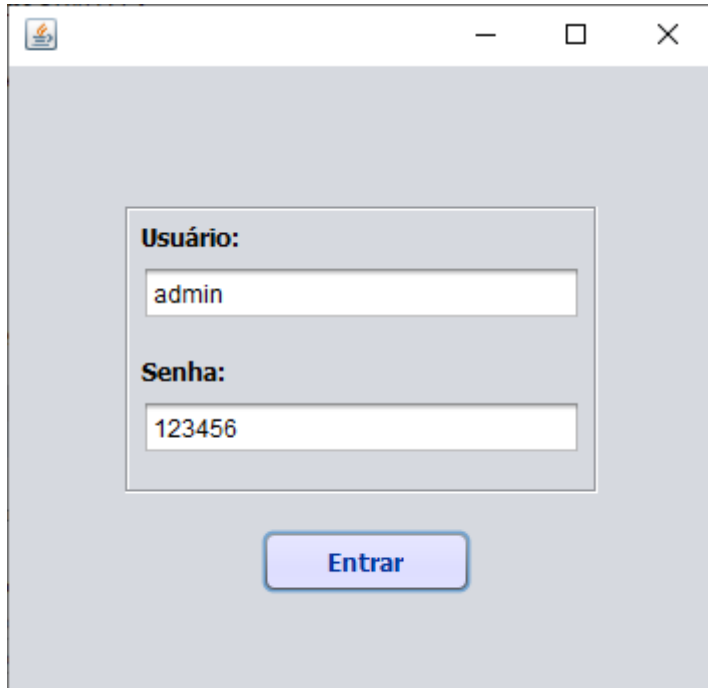
Usuário:
admin

Senha:
12

Entrar



Teste com login e senha correto

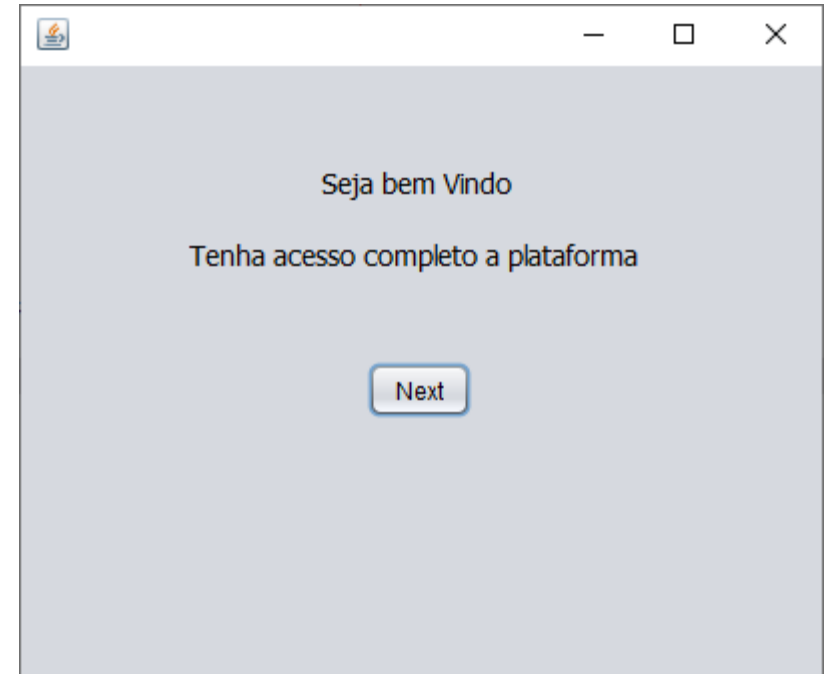


A screenshot of a login window with a light gray background. It features a white rectangular box containing two input fields. The first field is labeled 'Usuário:' and contains the text 'admin'. The second field is labeled 'Senha:' and contains the text '123456'. Below these fields is a blue button with the text 'Entrar' in white. The window has a standard title bar with a small icon on the left and minimize, maximize, and close buttons on the right.

Usuário:
admin

Senha:
123456

Entrar



SQL Injection

SQL Injection

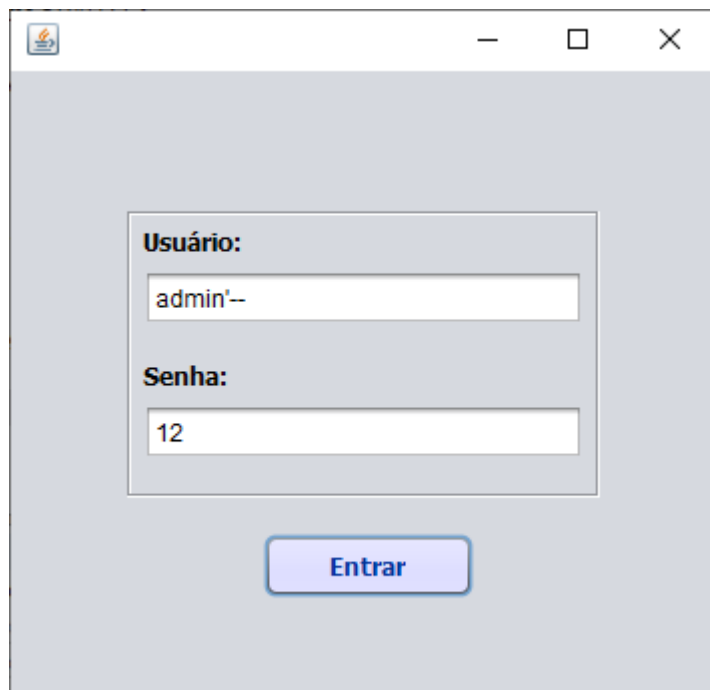
- Injeção de SQL é um tipo de ataque que se aproveita de falhas em sistemas com banco de dados através de comandos SQL, onde o atacante consegue inserir uma instrução SQL personalizada e indevida dentro de uma consulta (SQL query) através da entradas de dados de uma aplicação.
- Um usuário, por meio de ataques com injeção SQL, é possível obter qualquer tipo de dado sigiloso mantido no banco de dados de um computador servidor.

SQL Injection

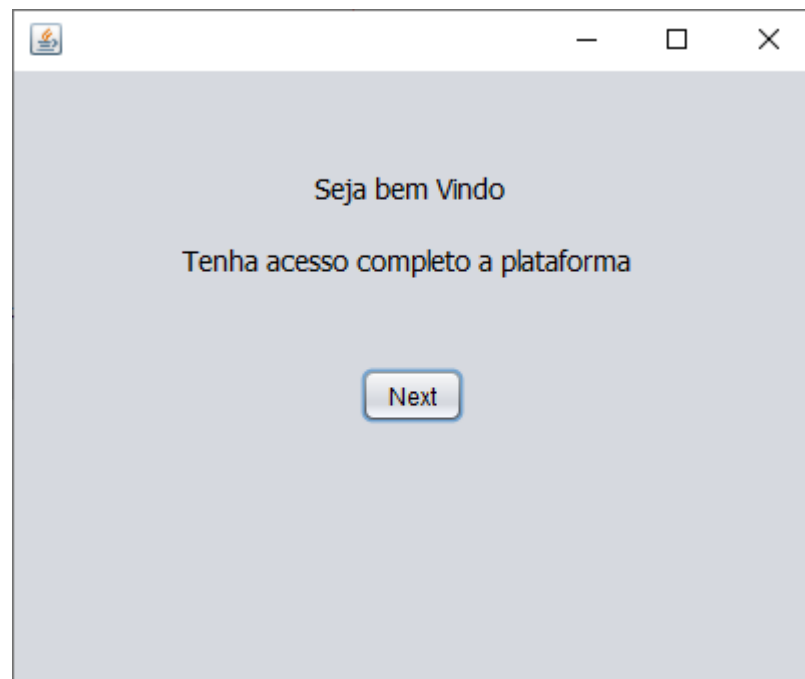
- Para evitar esse tipo de ataque o desenvolvedor precisa implementar a classe DAO da forma segura, evitando essa vulnerabilidade.
- Veremos como implementar corretamente, porém primeiro testaremos nossa Tela de Login.

Primeiro teste de SQL Injection

- Usando ' --



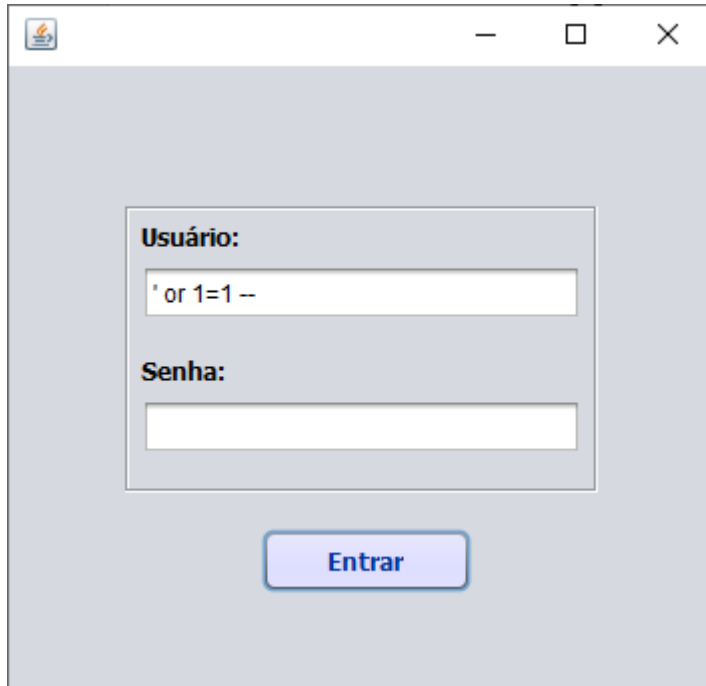
A screenshot of a web application login window. It features a light gray background with a central white box containing two input fields. The first field is labeled 'Usuário:' and contains the text 'admin'--'. The second field is labeled 'Senha:' and contains the text '12'. Below the input fields is a blue button with the text 'Entrar'.



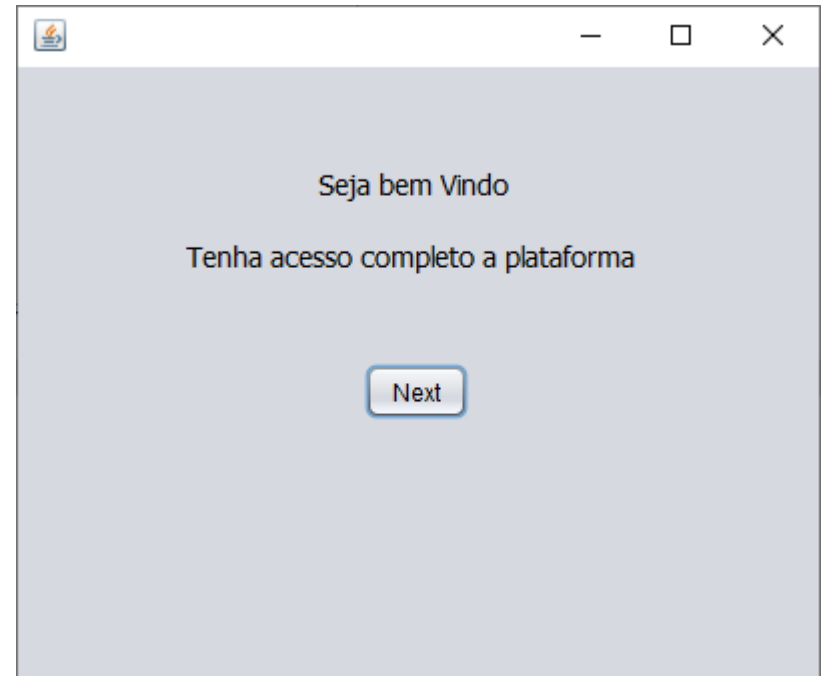
Se nenhuma verificação de dados (validação) for realizada, o usuário atacante conseguirá efetuar o login no sistema.

Segundo teste de SQL Injection

- Usando **' or 1 = 1 --**



A screenshot of a web application's login page. It features a light gray background with a central white box containing two input fields. The first field is labeled 'Usuário:' and contains the text **' or 1 = 1 --**. The second field is labeled 'Senha:' and is empty. Below the input fields is a blue button with the text 'Entrar'.



Se nenhuma verificação de dados (validação) for realizada, o usuário atacante conseguirá efetuar o login no sistema.

Terceiro teste de SQL Injection

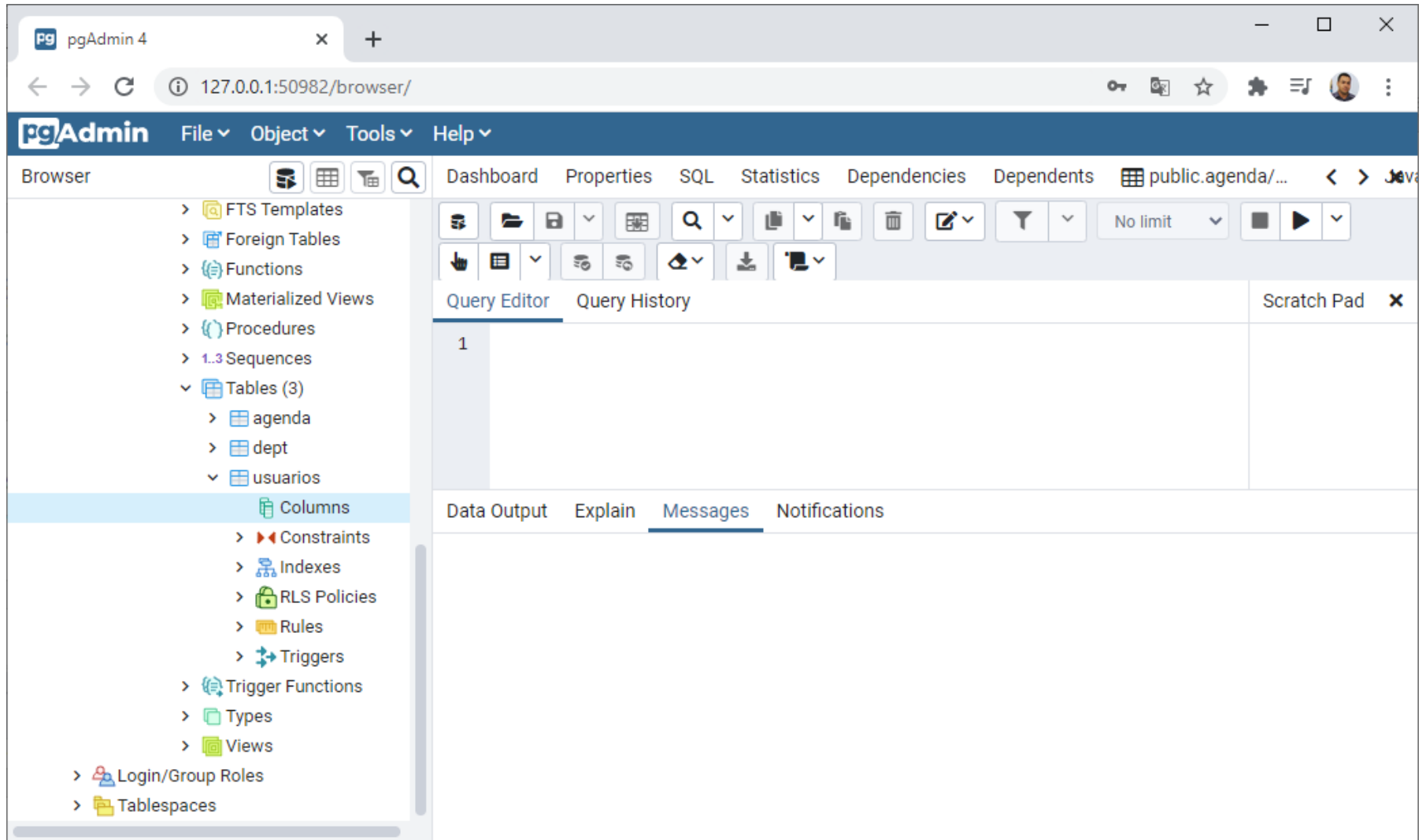
- Deletando a Tabela com **admin'; DROP TABLE usuarios ; --**



A screenshot of a web application login form. The form is titled "Usuário:" and "Senha:". The "Usuário:" field contains the text "admin'; DROP TABLE usuarios ; --". The "Senha:" field is empty. Below the fields is a blue button labeled "Entrar". The form is displayed in a window with standard Windows window controls (minimize, maximize, close) at the top.

A Tabela foi apagada.

Terceiro teste de SQL Injection



A Tabela foi apagada.

Evitando SQL Injection

Classe UsuárioDAO

```
package model;

import java.sql.SQLException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class UsuarioDAO {
    private final Connection conn;
    private PreparedStatement pstmt;
    private ResultSet rs;

    public UsuarioDAO() {
        Conexao conexao = new Conexao();
        conn = conexao.conectar();
    }

    public boolean verificaLogin(Usuario usuario){
        String sql = "select * from usuarios where usuario = '" + usuario.getLogin()
            + "' and senha = '" + usuario.getSenha() + "'";

        try {
            pstmt = conn.prepareStatement(sql);
            // Executando o comando sql e armazenando no ResultSet
            rs = pstmt.executeQuery();
            //Retornando o ResultSet
            while(rs.next()){
                return true;
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
        return false;
    }
}
```

Incorreto



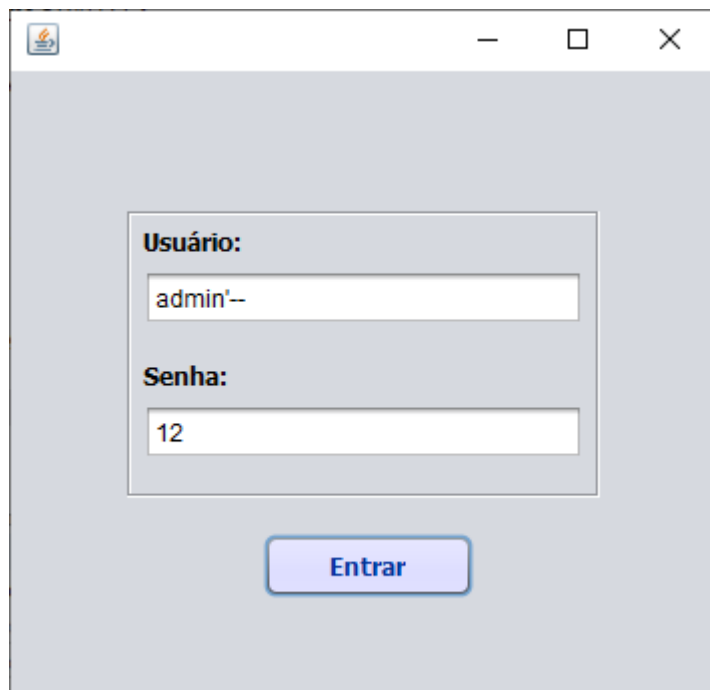
Classe UsuárioDAO

```
public boolean verificaLogin(Usuario usuario){  
    //String sql = "select * from usuarios where usuario = '" + usuario.getLogin()  
    //                + "' and senha = '" + usuario.getSenha() + "'";  
    String sql = "select * from usuarios where usuario = ? and senha = ? ";  
    try {  
        pstmt = conn.prepareStatement(sql);  
        pstmt.setString(1, usuario.getLogin());  
        pstmt.setString(1, usuario.getSenha());  
        // Executando o comando sql e armazenando no ResultSet  
        rs = pstmt.executeQuery();  
        //Retornando o ResultSet  
        while(rs.next()){  
            return true;  
        }  
    } catch (SQLException ex) {  
        System.out.println(ex.getMessage());  
    }  
    return false;  
}
```



Primeiro teste de SQL Injection

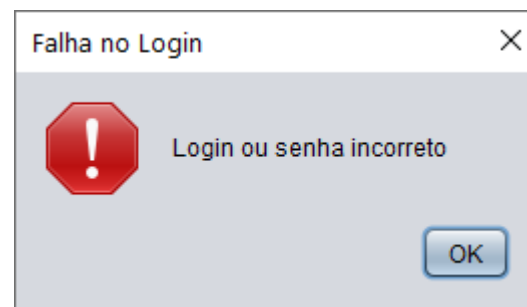
- Usando ' --



Usuário:
admin'--

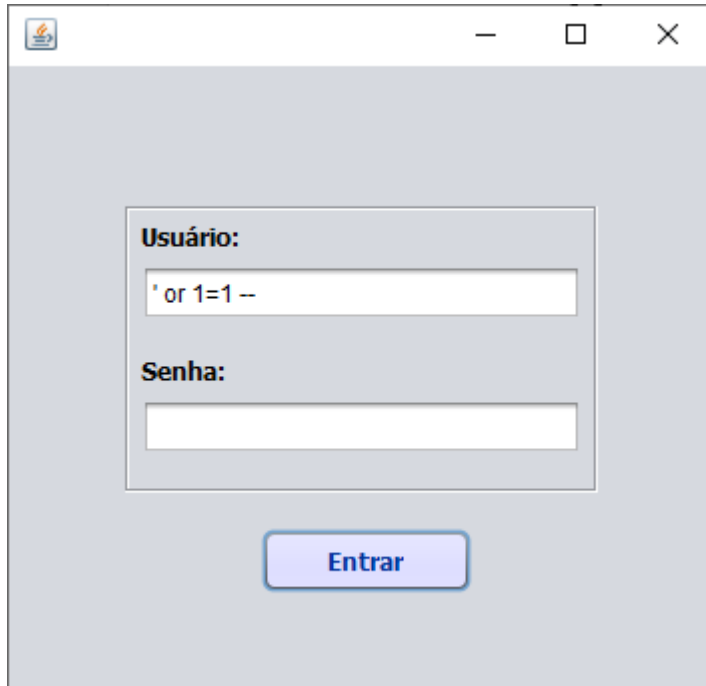
Senha:
12

Entrar



Segundo teste de SQL Injection

- Usando **' or 1 = 1 --**

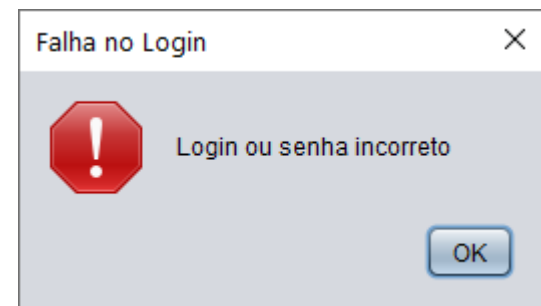


Usuário:

' or 1=1 --

Senha:

Entrar



Terceiro teste de SQL Injection



- Deletando a Tabela com **admin'; DROP TABLE usuarios ; --**

Usuário:

admin'; DROP TABLE usuarios ; --

Senha:

Entrar

Terceiro teste de SQL Injection



pgAdmin 4

127.0.0.1:50982/browser/

pgAdmin File Object Tools Help

Browser

- > Aa FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Procedures
- > 1.3 Sequences
- ▼ Tables (3)
 - > agenda
 - > dept
 - ▼ usuarios
 - ▼ Columns (2)
 - usuario
 - senha
 - > Constraints
 - > Indexes
 - > RLS Policies
 - > Rules
 - > Triggers
 - > Trigger Functions
 - > Types

Dashboard Properties SQL Statistics Dependencies Dependents public.agenda/...

Query Editor Query History Scratch Pad

```
1 SELECT * FROM public.usuarios
2 ORDER BY usuario ASC
```

Data Output Explain Messages Notifications

| | usuario [PK] character varying | senha character varying |
|---|-----------------------------------|----------------------------|
| 1 | admin | 123456 |
| 2 | fulano | fulano1234 |

A Tabela intacta.