DESENVOLVIMENTO DE APLICAÇÕES COM BANCO DE DADOS

2016.1

PROF ANDERSON

APRESENTAÇÃO

contato

asantos@unicarioca.edu.br

DESENVOLVIMENTO DE APLICAÇÕES COM BANCO DE DADOS

CONTEÚDO PROGRAMÁTICO

UNIDADE 1 INTRODUÇÃO

UNIDADE 2 FUNDAMENTOS SQL

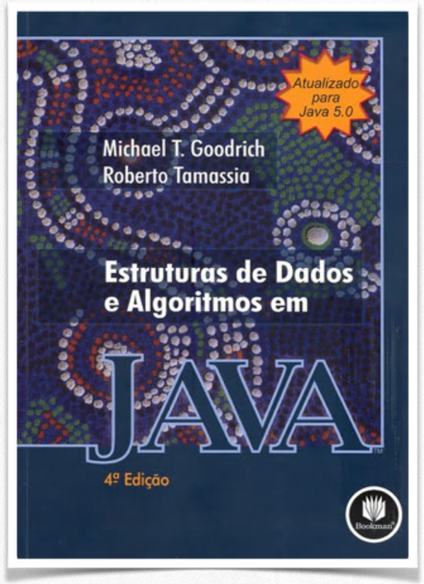
UNIDADE 3 JDBC

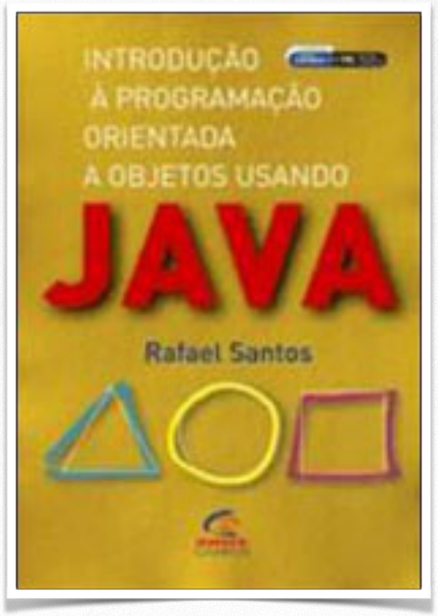
UNIDADE 4 DAO

UNIDADE 5 Hibernate

DESENVOLVIMENTO DE APLICAÇÕES COM BANCO DE DADOS

BIBLIOGRAFIA



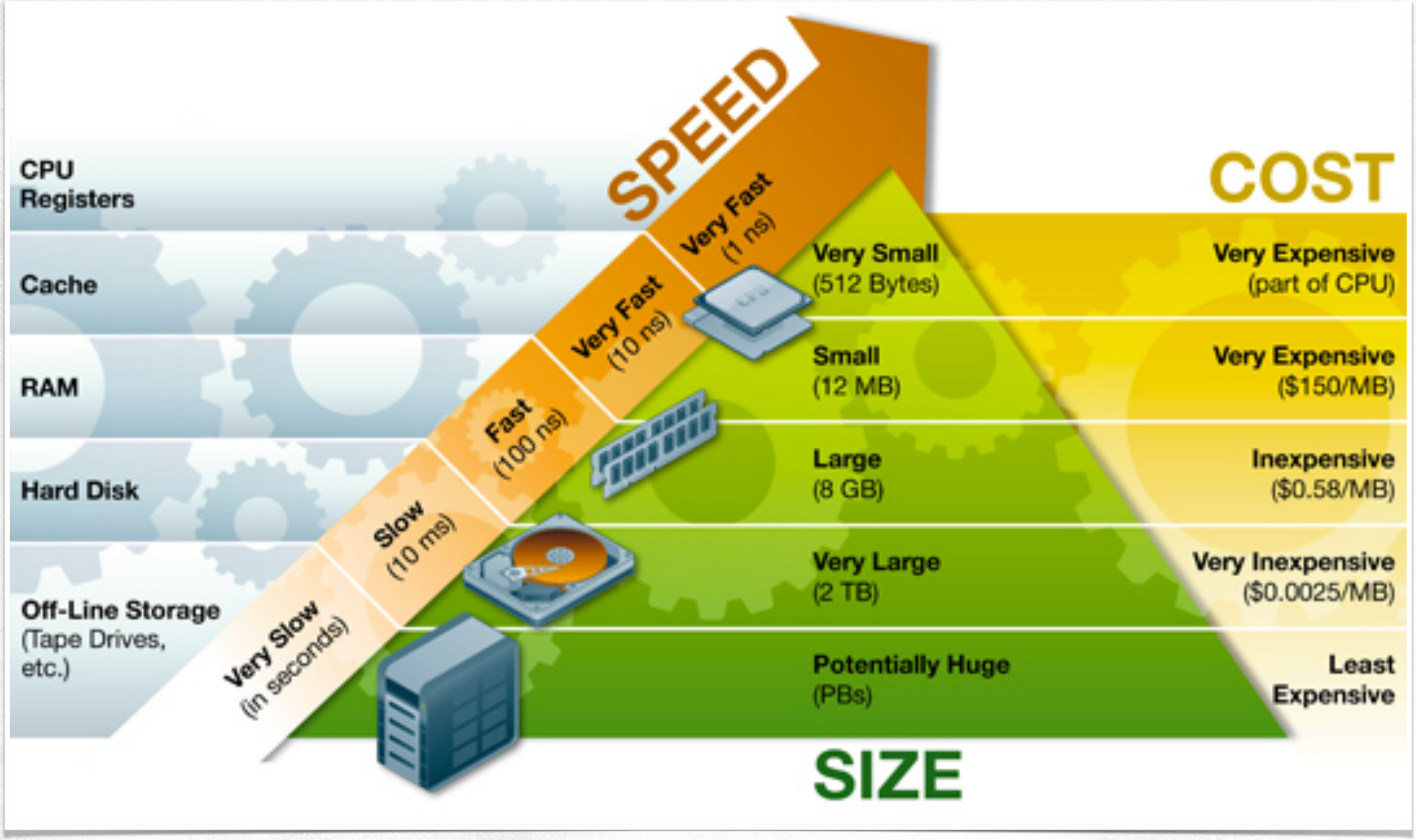


INTRODUÇÃO

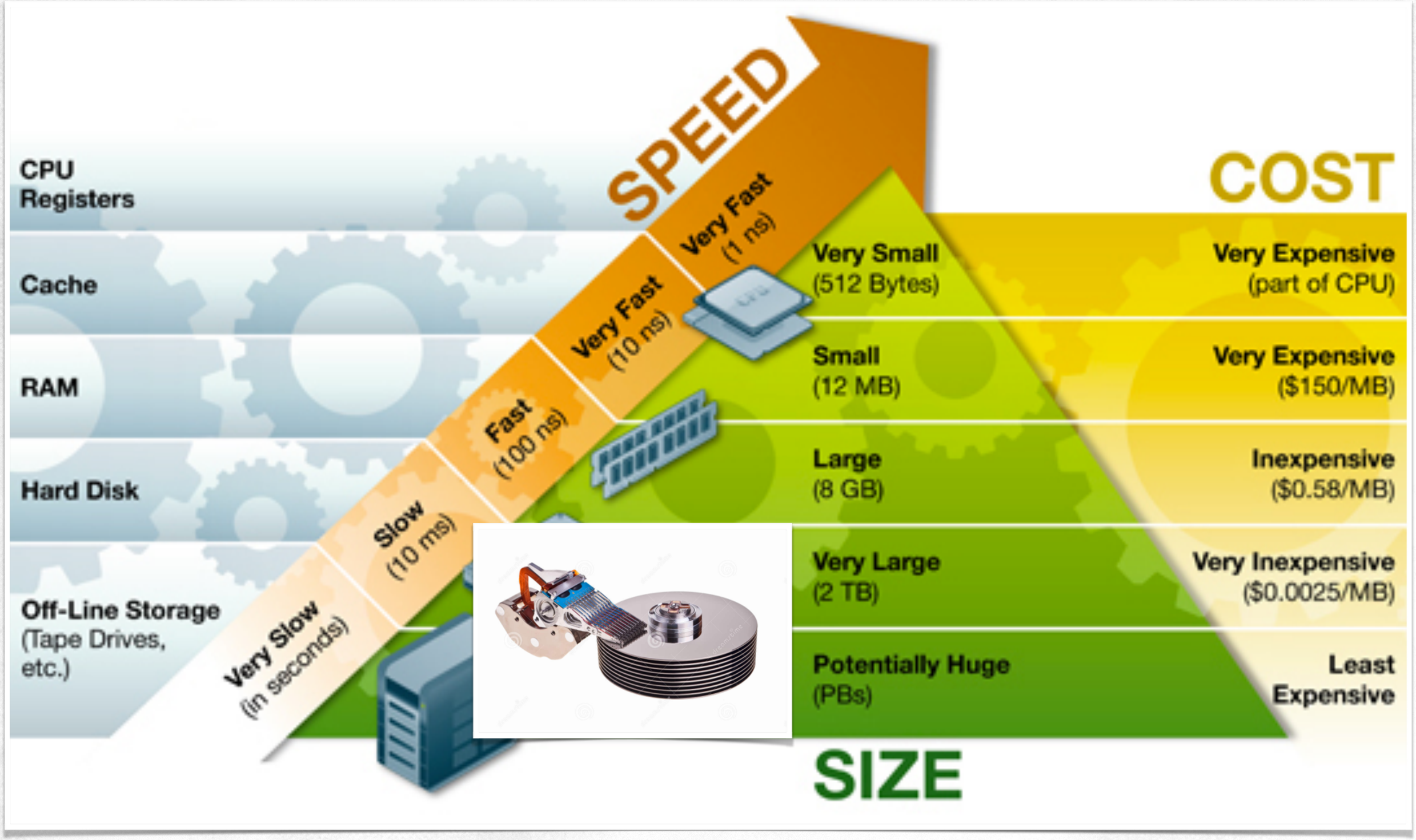
I

- Introdução
- Arquitetura
- Ambiente de Programação

INTRODUÇÃO


SPEED
COST
SIZE
CPU Registers
Cache
RAM
Hard Disk
Off-Line Storage (Tape Drives, etc.)
Very Fast (1 ns)
Very Fast (10 ns)
Fast (100 ns)
Slow (10 ms)
Very Slow (in seconds)
Very Small (512 Bytes)
Small (12 MB)
Large (8 GB)
Very Large (2 TB)
Potentially Huge (PBs)
Very Expensive (part of CPU)
Very Expensive ($150/MB)
Inexpensive ($0.58/MB)
Very Inexpensive ($0.0025/MB)
Least Expensive

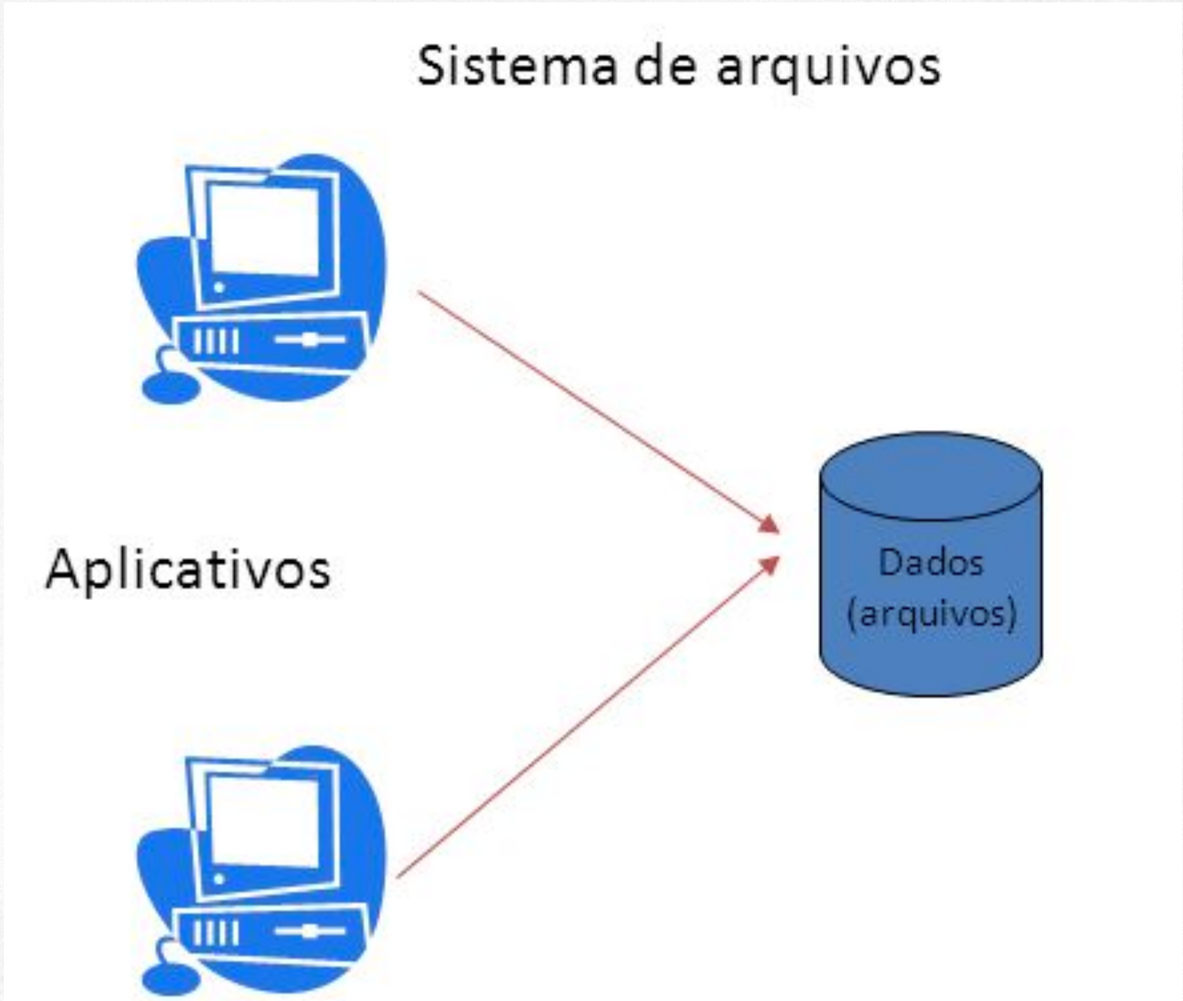
INTRODUÇÃO


SPEED
COST
SIZE
CPU Registers
Cache
RAM
Hard Disk
Off-Line Storage (Tape Drives, etc.)
Very Fast (1 ns)
Very Fast (10 ns)
Fast (100 ns)
Slow (10 ms)
Very Slow (in seconds)
Very Small (512 Bytes)
Small (12 MB)
Large (8 GB)
Very Large (2 TB)
Potentially Huge (PBs)
Very Expensive (part of CPU)
Very Expensive ($150/MB)
Inexpensive ($0.58/MB)
Very Inexpensive ($0.0025/MB)
Least Expensive

INTRODUÇÃO

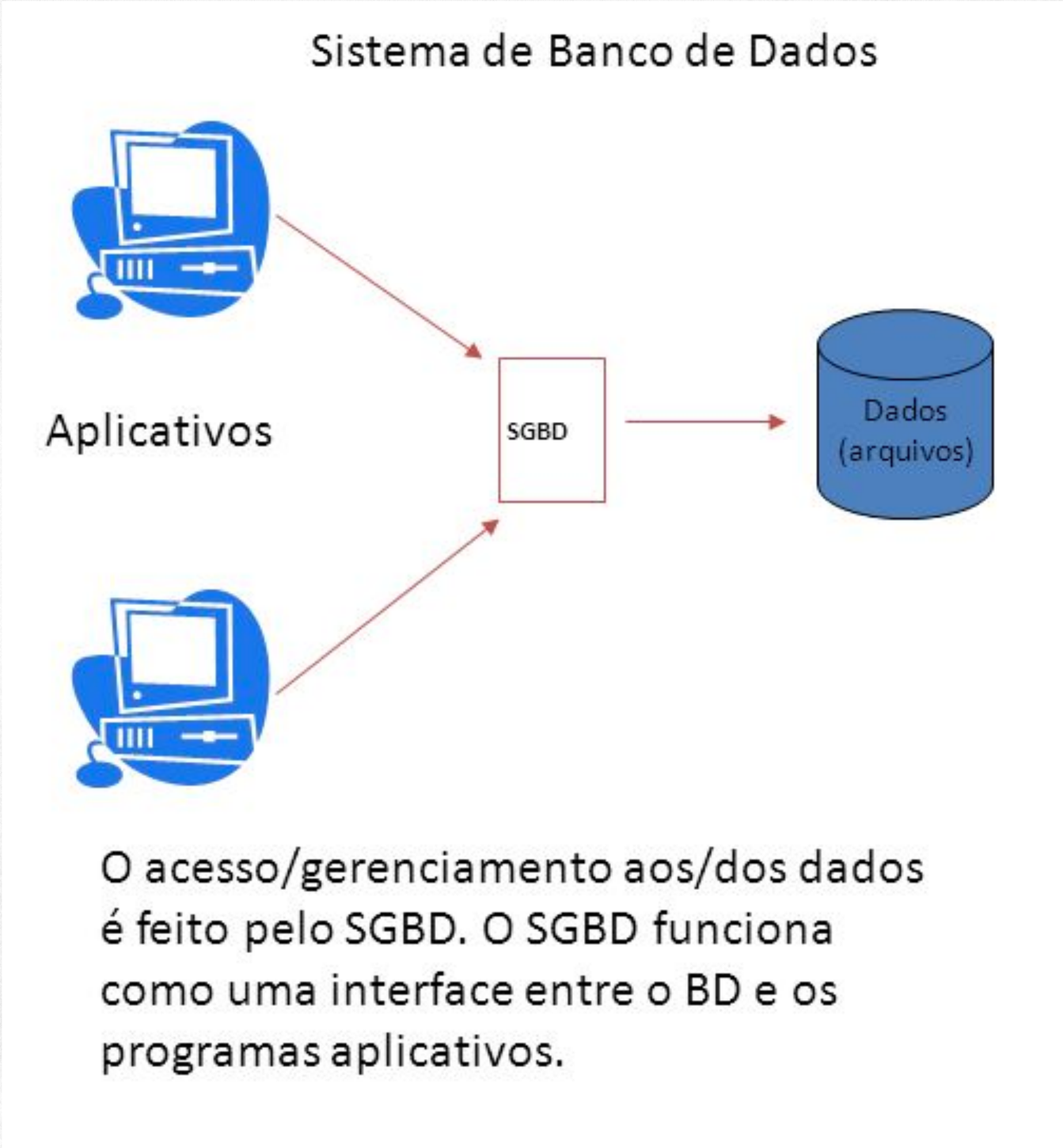
CPU
Registers
Cache
RAM
Hard Disk
Off-Line Sto
(Tape Drives,
etc.)

INTRODUÇÃO

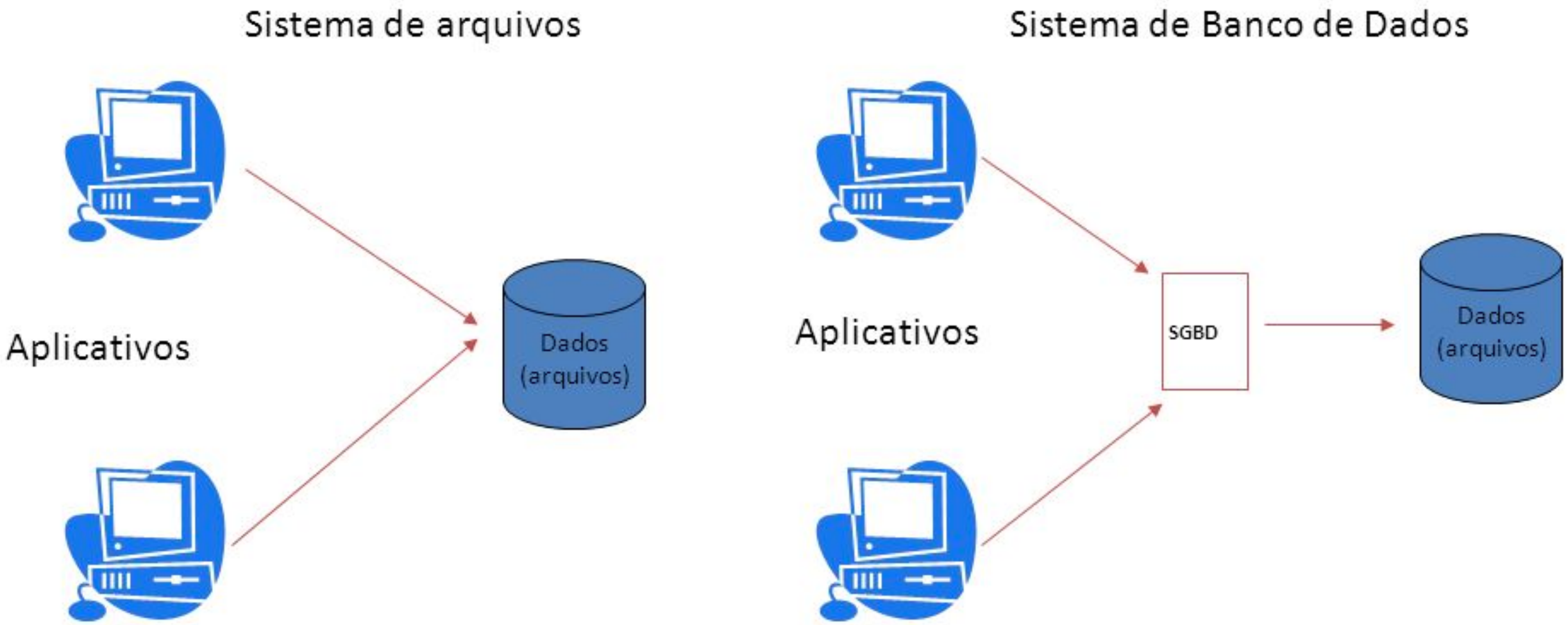


O acesso/gerenciamento aos/dos dados é feito diretamente pelos programas aplicativos.

INTRODUÇÃO


Sistema de Banco de Dados
Aplicativos
SGBD
Dados
(arquivos)
O acesso/gerenciamento aos/dos dados é feito pelo SGBD. O SGBD funciona como uma interface entre o BD e os programas aplicativos.

INTRODUÇÃO

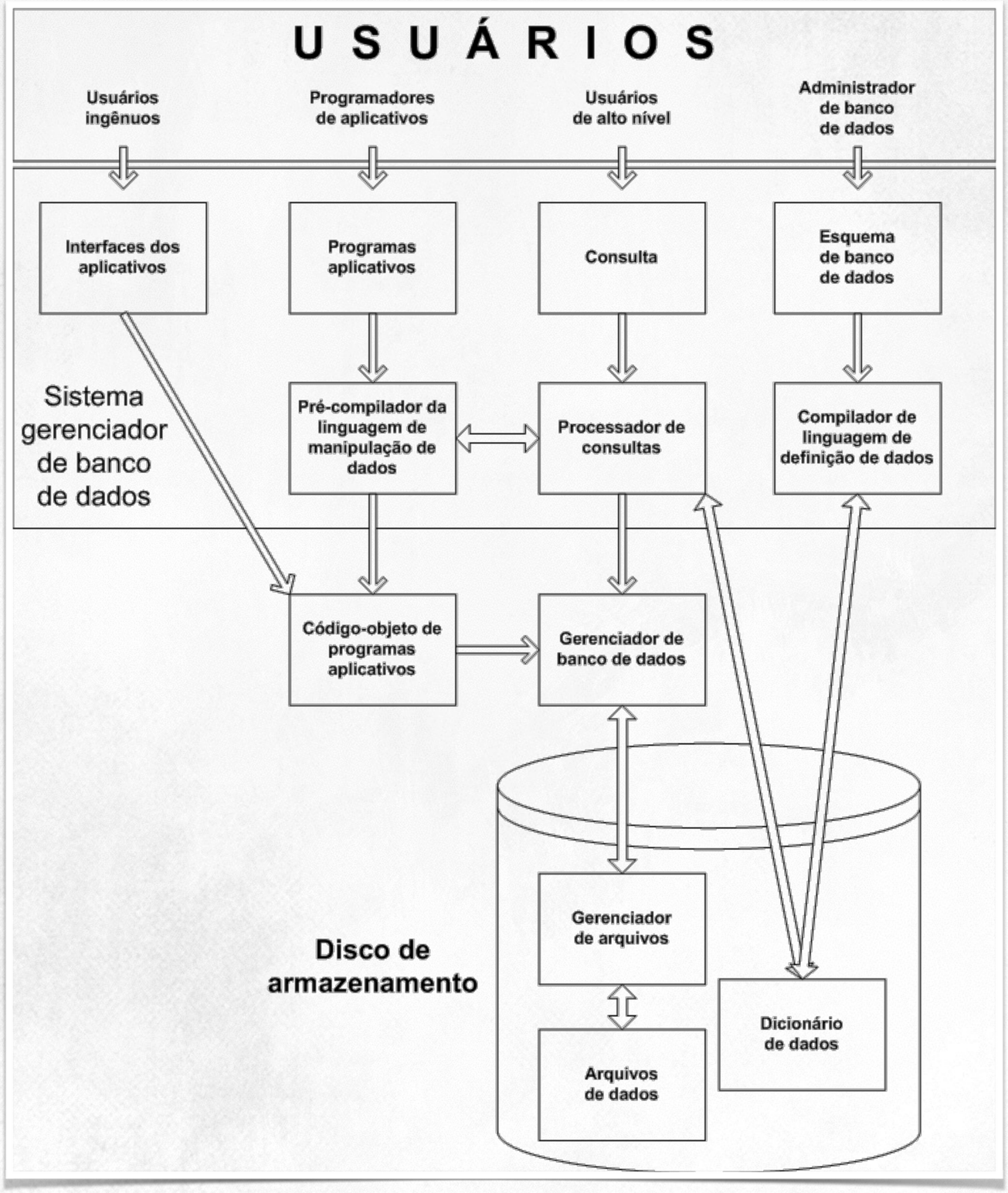


O acesso/gerenciamento aos/dos dados é feito diretamente pelos programas aplicativos.

O acesso/gerenciamento aos/dos dados é feito pelo SGBD. O SGBD funciona como uma interface entre o BD e os programas aplicativos.

INTRODUÇÃO

ARQUITETURA

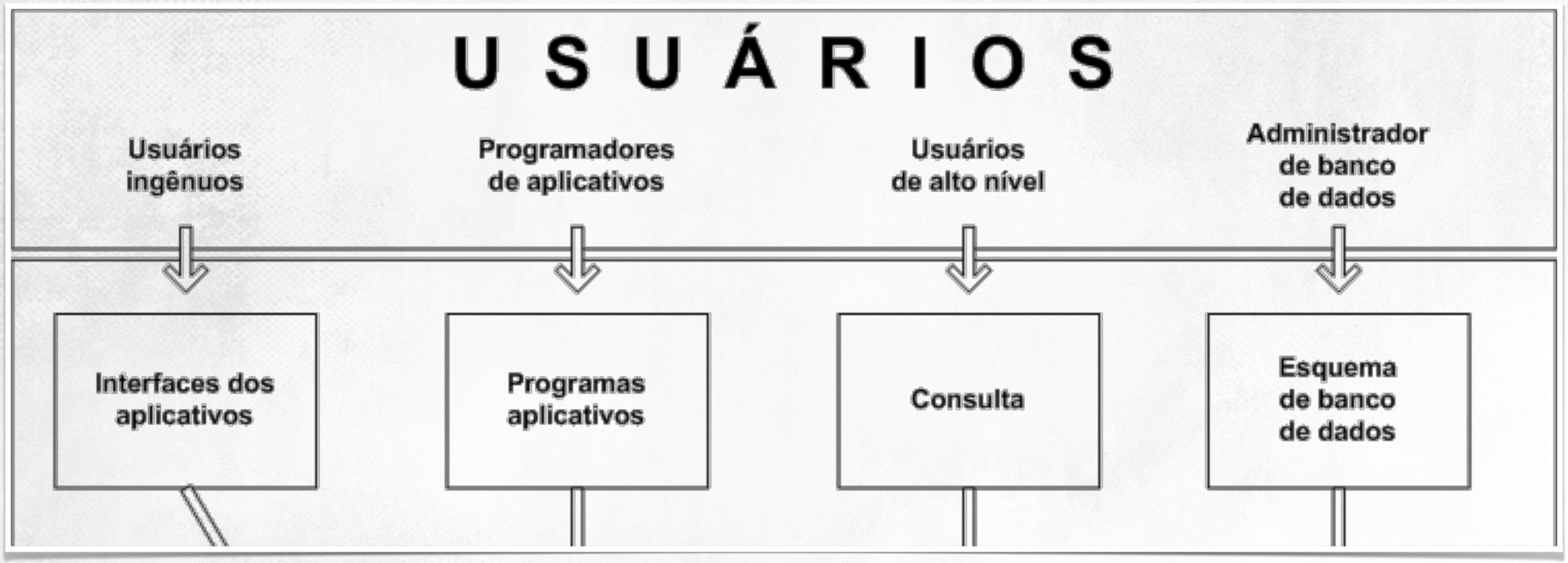


INTRODUÇÃO

ARQUITETURA

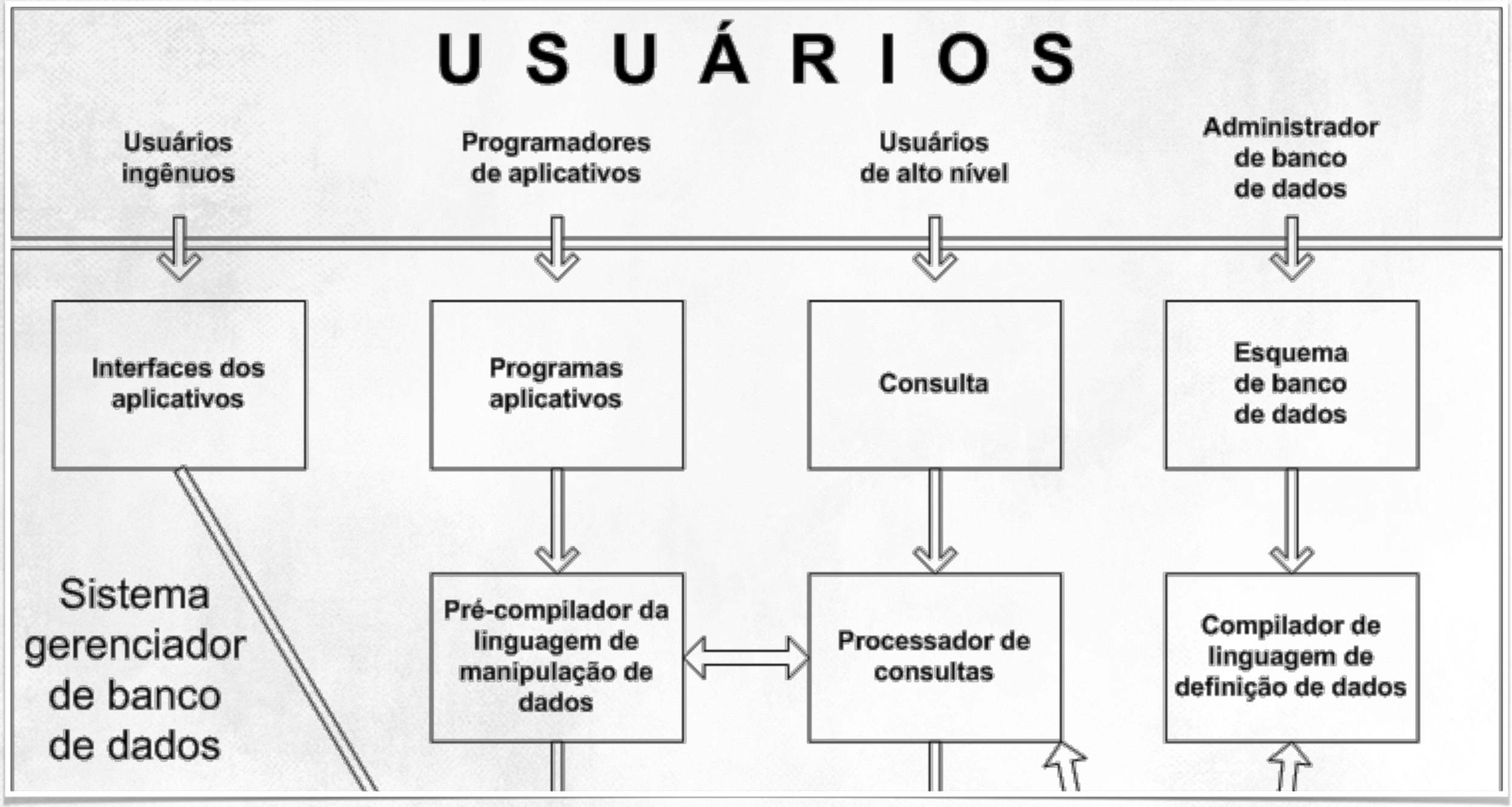
INTRODUÇÃO

ARQUITETURA



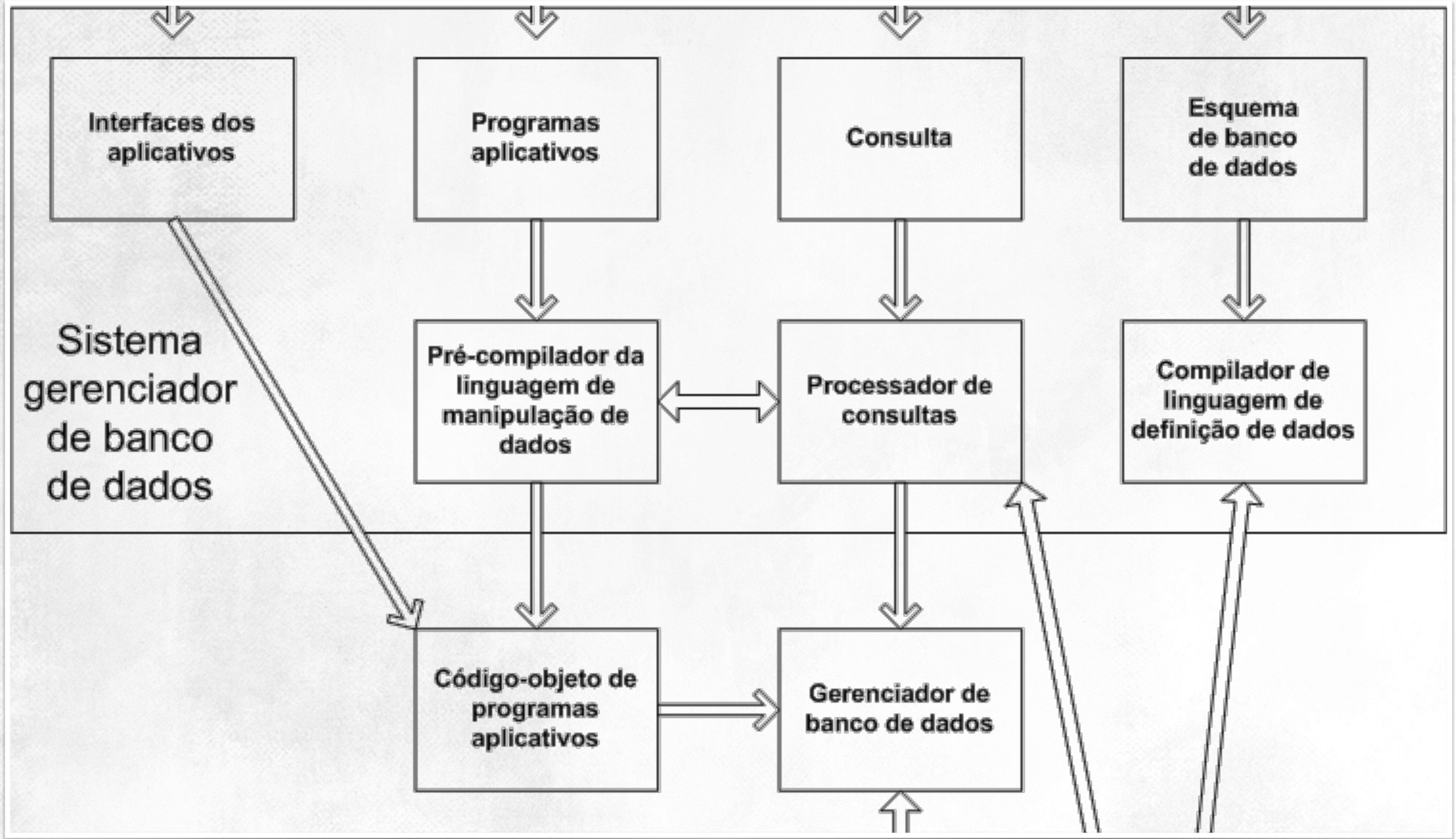
INTRODUÇÃO

ARQUITETURA



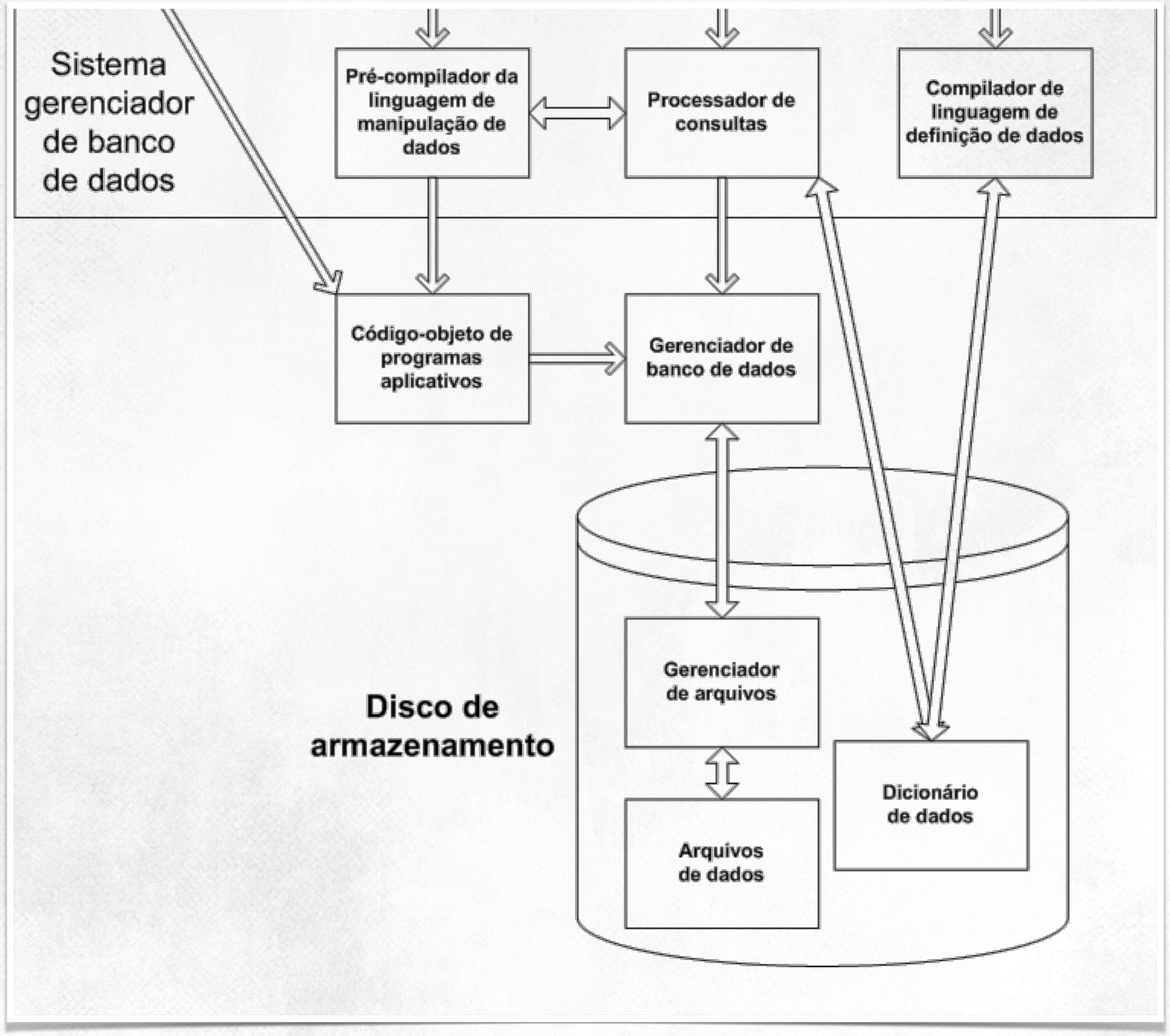
INTRODUÇÃO

ARQUITETURA



INTRODUÇÃO

ARQUITETURA



FUNDAMENTOS DE SQL

II

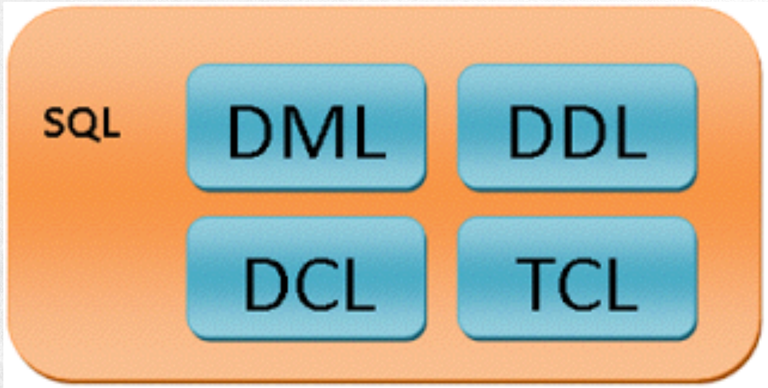
- Estrutura
- Revisão DDL
- Revisão DML
- Revisão DQL

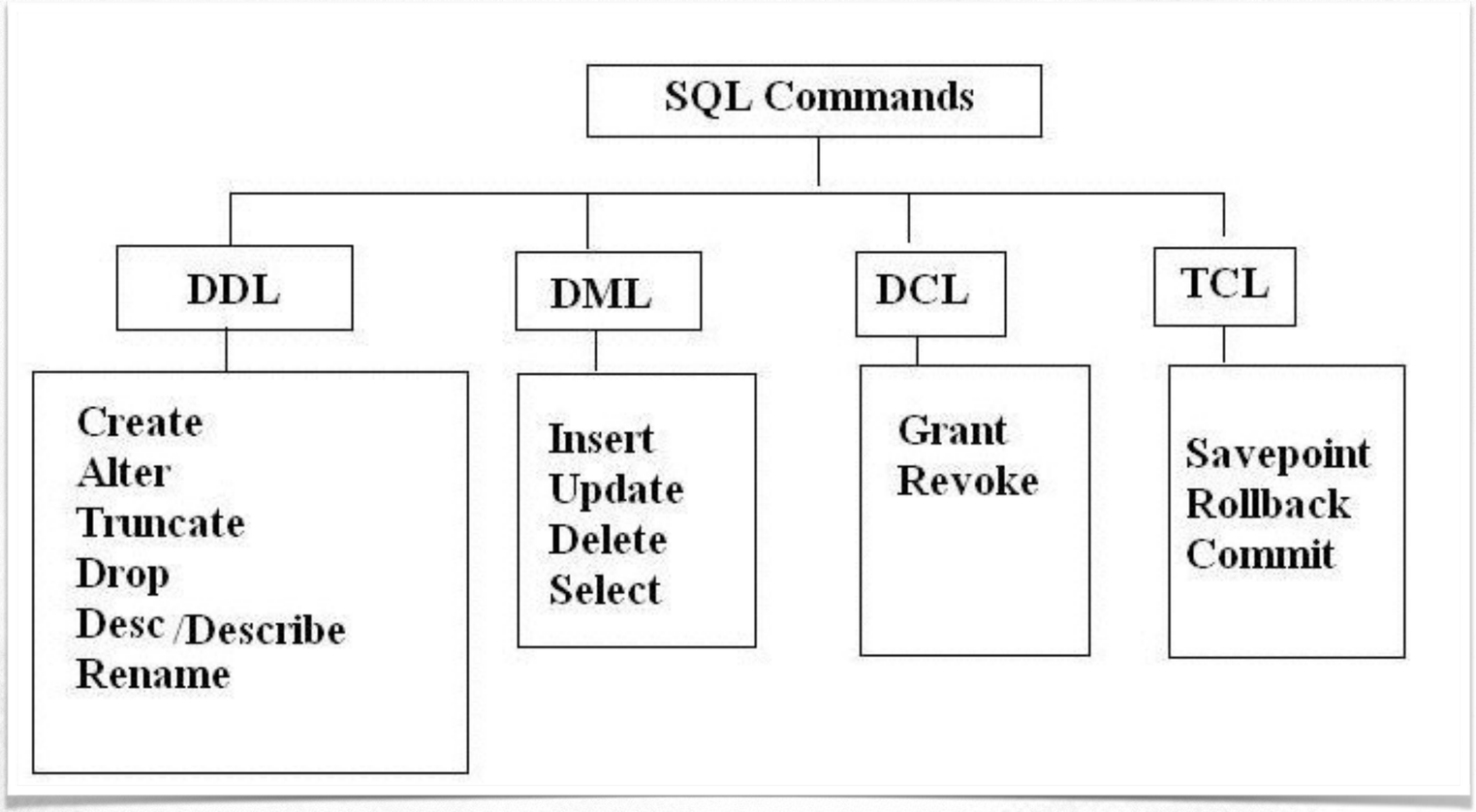
FUNDAMENTOS DE SQL

SQL

SQL: STRUCTURED QUERY LANGUAGE

Linguagem de Consulta Estruturada





FUNDAMENTOS DE SQL

SQL

SQL

DDL: Data Definition Language - Linguagem de Definição de Dados.

- ALTER, CREATE, DROP, RENAME, TRUNCATE.

FUNDAMENTOS DE SQL

DDL

SQL>>DDL>>CREATE

-Permite criar uma base de dados, função/procedimento, índice, tabela, trigger, view.

CREATE TABLE t (c CHAR(20) CHARACTER SET utf8 COLLATE utf8_bin);

```
mysql> create table filmes (
    -> id int(10) unsigned not null auto_increment,
    -> titulo varchar(80) not null,
    -> ano int(4) unsigned not null,
    -> diretor varchar(80) not null,
    -> primary key (id));
Query OK, 0 rows affected (0.14 sec)

mysql>
```

FUNDAMENTOS DE SQL

DDL

SQL>>DDL>>ALTER

-Permite alterar as características de uma base de dados, função/procedimento, tabela e view.

ALTER TABLE t2 **DROP COLUMN** c, **DROP COLUMN** d;

```
mysql> alter table filmes add column roterista varchar(80);
Query OK, 0 rows affected (0.07 sec)

mysql>
```

FUNDAMENTOS DE SQL

DDL

SQL>>DDL>>RENAME

- Renomeia uma ou mais tabelas.

RENAME TABLE T1 **TO** T2;

```
mysql> rename table filmes to filme;
```

FUNDAMENTOS DE SQL

DDL

SQL>>DDL>>TRUNCATE

- Apaga os registros de uma tabela;

TRUNCATE TABLE T1;

```
mysql> truncate table filme;
```

FUNDAMENTOS DE SQL

DDL

SQL>>DDL>>DROP

- Apaga uma base de dados, tabela, função/procedimento, índice, trigger e índice;

DROP TABLE T1;

```
mysql> drop table filme;
```

FUNDAMENTOS DE SQL

DML

SQL

DML: Data Manipulation Language – Linguagem de Manipulação de Dados.

– INSERT, SELECT, UPDATE e DELETE.

FUNDAMENTOS DE SQL

DML

SQL>>DML>>INSERT

- Insere novas linhas em uma tabela já existente.

INSERT INTO tbl_name () **VALUES**();

```
mysql> show tables;
+--------------------+
| Tables_in_locadora |
+--------------------+
| filmes             |
+--------------------+
1 row in set (0.05 sec)
mysql>
```

```
mysql> insert into filmes (id,titulo,ano,diretor)
    -> values (0,'King Kong',2006,'Peter Jackson');
Query OK, 1 row affected (0.08 sec)
mysql>
```

FUNDAMENTOS DE SQL

DML

SQL>>DML>>SELECT

- É usado para recuperar linhas selecionadas de uma ou mais tabelas.

```
mysql> show tables;
+--------------------+
| Tables_in_locadora |
+--------------------+
| filmes             |
+--------------------+
1 row in set (0.05 sec)
mysql>
```

SELECT * FROM t1 WHERE ...

```
mysql> select * from filmes;
+----+--------------------+------+------------------+
| id | titulo             | ano  | diretor          |
+----+--------------------+------+------------------+
|  1 | King Kong          | 2006 | Peter Jackson    |
|  2 | Guerra dos Mundos  | 2005 | Steven Spielberg |
+----+--------------------+------+------------------+
2 rows in set (0.01 sec)
mysql>
```

FUNDAMENTOS DE SQL

DML

SQL>>DML>>UPDATE

- Atualiza o valor dos campos de uma tabela.

UPDATE t1 **SET** col1 = col1 + 1 **WHERE...**

```
mysql> show tables;
+--------------------+
| Tables_in_locadora |
+--------------------+
| filmes             |
+--------------------+
1 row in set (0.05 sec)
mysql>
```

```
mysql> UPDATE filmes SET ano=2006;
Query OK, 1 row affected (0.17 sec)
Linhas que combinaram: 3 - Alteradas: 1 - Avisos: 0
```

FUNDAMENTOS DE SQL

DML

SQL>>DML>>DELETE

- Apaga as linhas de uma tabela.

DELETE FROM somelog **WHERE** user = 'jcole'

```
mysql> show tables;
+--------------------+
| Tables_in_locadora |
+--------------------+
| filmes             |
+--------------------+
1 row in set (0.05 sec)
mysql>
```

```
mysql> DELETE FROM filmes WHERE ano<2006;
Query OK, 1 row affected (0.08 sec)
```

FUNDAMENTOS DE SQL

SQL

Gerenciamento de uma coleção de CDs.

1. Tabela

- nome do disco;
- nome do cantor(a)/grupo; e
- estilo musical

FUNDAMENTOS DE SQL

SQL

Gerenciamento de uma coleção de CDs.

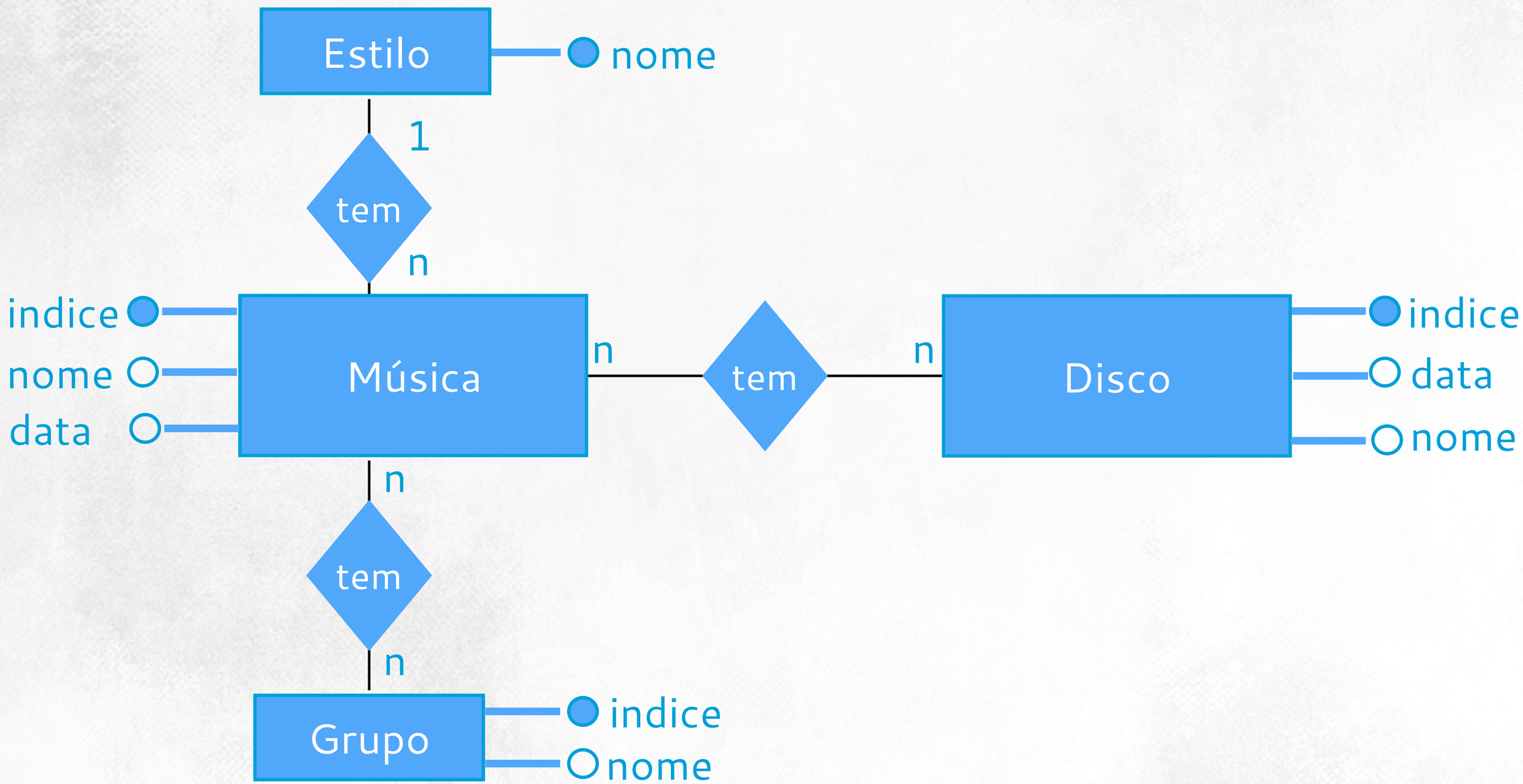
1. Tabela

- nome do disco;
- nome do cantor(a)/grupo; e
- estilo musical

chave primária com a tupla acima.

FUNDAMENTOS DE SQL

Gerenciamento de uma coleção de CDs.



FUNDAMENTOS DE SQL

Gerenciamento de uma coleção de CDs.

1. Criação da base

```
create database colecaocd;
```

2. Seleção da base

```
use colecaocd
```

3. Criação da tabela Grupo

```
create table estilo (nome varchar(80) not null, primary
key(nome));
```

FUNDAMENTOS DE SQL

Gerenciamento de uma coleção de CDs.

4. Criação da tabela grupo

```
create table grupo(indice int(4) not null
auto_increment, nome varchar(80) not null, primary
key(indice));
```

5. Criação da tabela musica

```
create table musica (indice int(10) not null
auto_increment, nome varchar(80) not null, data
date, primary key(indice));
```

FUNDAMENTOS DE SQL

Gerenciamento de uma coleção de CDs.

6. Criação da tabela disco

```
create table musica (indice int(10) not null auto_increment,
 nome varchar(80) not null, data date, primary key(indice));
```

7. Relacionamento Grupo Música

```
create table grupomusica (grupo_indice int(4) not null,
musica_indice int(10) not null, primary key(grupo_indice,
musica_indice), foreign key (grupo_indice) references
grupo(indice) on delete restrict, foreign key
(musica_indice) references musica(indice) on delete
cascade);
```

FUNDAMENTOS DE SQL

Gerenciamento de uma coleção de CDs.

8. Relacionamento Disco Música

```
create table discomusica (disco_indice int(10) not null,
musica_indice int(10) not null, primary key(disco_indice,
musica_indice), foreign key (disco_indice) references
disco(indice) on delete cascade, foreign key (musica_indice)
references musica(indice) on delete restrict);
```

FUNDAMENTOS DE SQL

Gerenciamento de uma coleção de CDs.

9. Relacionamento Disco Música

```
create table discomusica (disco_indice int(10) not null,
musica_indice int(10) not null, primary key(disco_indice,
musica_indice), foreign key (disco_indice) references
disco(indice) on delete cascade, foreign key (musica_indice)
references musica(indice) on delete restrict);
```

FUNDAMENTOS DE SQL

Gerenciamento de uma coleção de CDs.

10. Faça download da base em excel e insira na base do mysql.

JDBC

III

- Introdução
- Estabelecendo uma conexão
- Conectando com um DSO
- Tratando Exceções
- Manipulando tabelas
- Prepared Statement

JDBC

INTRODUÇÃO

- É uma biblioteca que permite estabelecer uma “conversa" com um SGBD;
- driver

JDBC

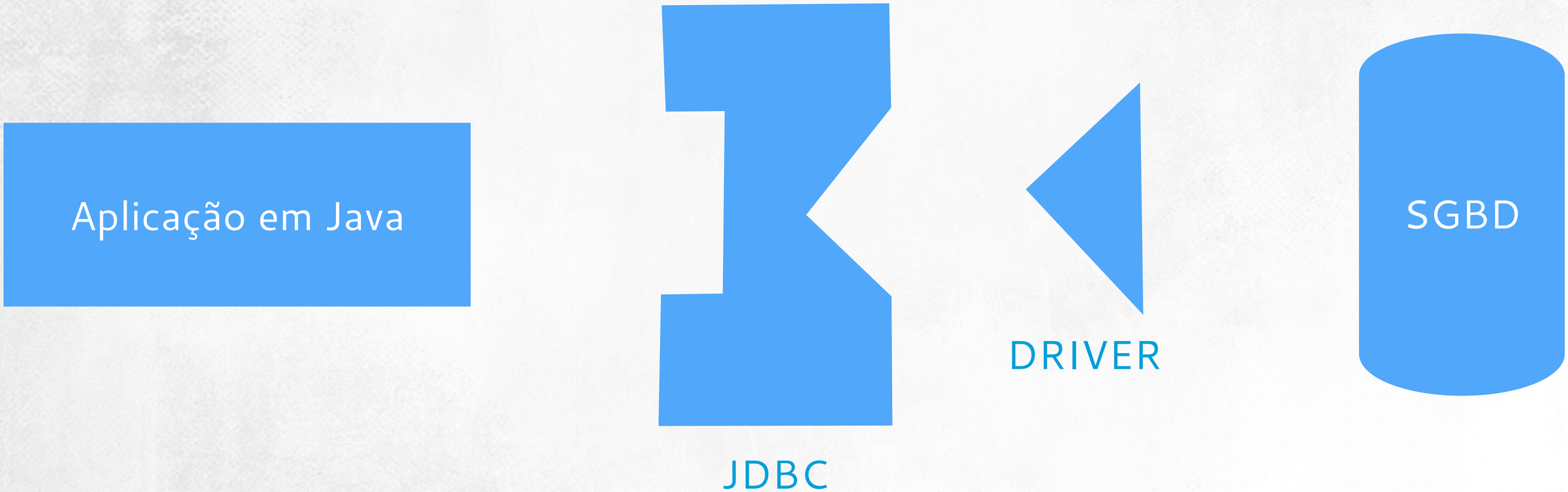
INTRODUÇÃO

JDBC

INTRODUÇÃO

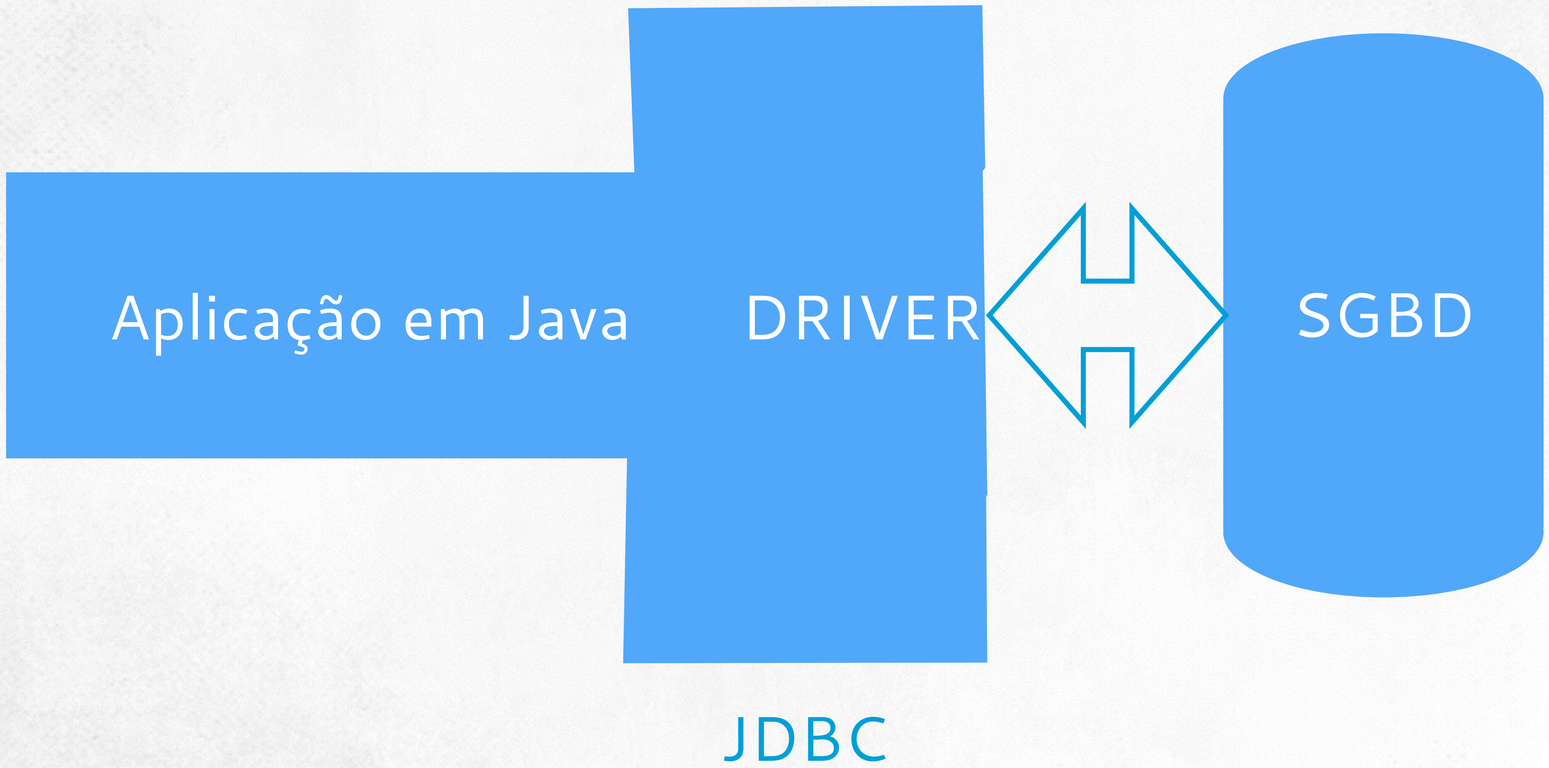
JDBC

INTRODUÇÃO



JDBC

INTRODUÇÃO



JDBC

INTRODUÇÃO

- A aplicação chama a biblioteca JDBC
- A biblioteca carrega o driver que “entende” o SGDB
- Após, a aplicação pode se conectar e enviar requisições ao SGBD
- Pacote principal: java.sql

JDBC

INTRODUÇÃO

- Java Database Connectivity;
- Padrão de acesso a BD's relacionais através de Java:
  - API comum;
  - Os fabricantes de drivers JDBC implementam aspectos específicos;
- Qualquer aplicação Java pode acessar um SGBD através do JDBC;

JDBC

PASSO-A-PASSO

DRIVER
Instanciar o driver

1

CONEXÃO
Est Conexão SGBD

2

STMT
"Canal Conversa"

3

DML
SELECT

4

RESULT
Resultado SELECT

5

JDBC

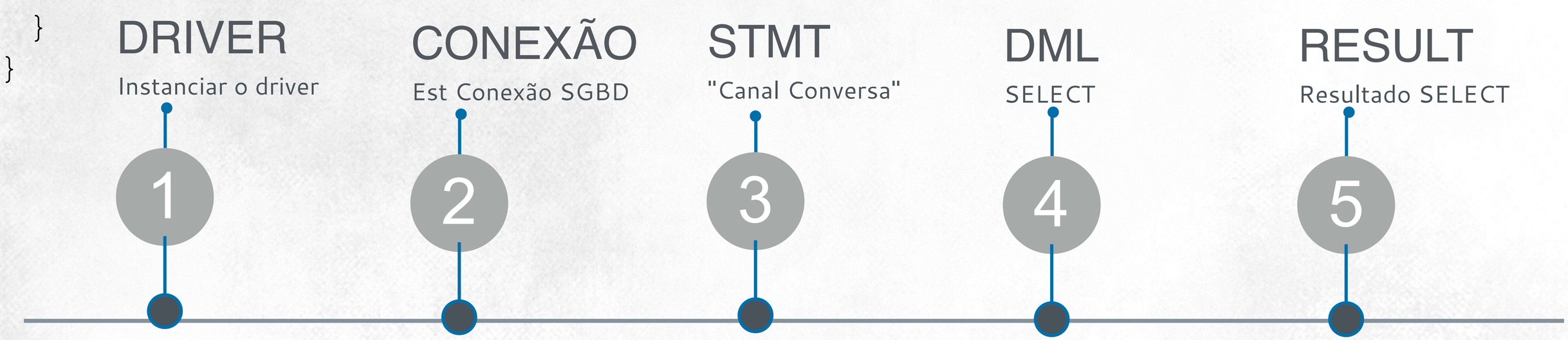
PASSO-A-PASSO

```
public class UniCarioca {
 public static void main(String[] args) {

       Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colecaocd","aluno","unicarioca");
                Statement stmt = con.createStatement()) {
            ResultSet rs = stmt.executeQuery("select * from
estilo;");
            while (rs.next()) {
                System.out.println("Estilo: " +
rs.getString("nome"));
            }
        }



  }
}
```

JDBC

PASSO-A-PASSO

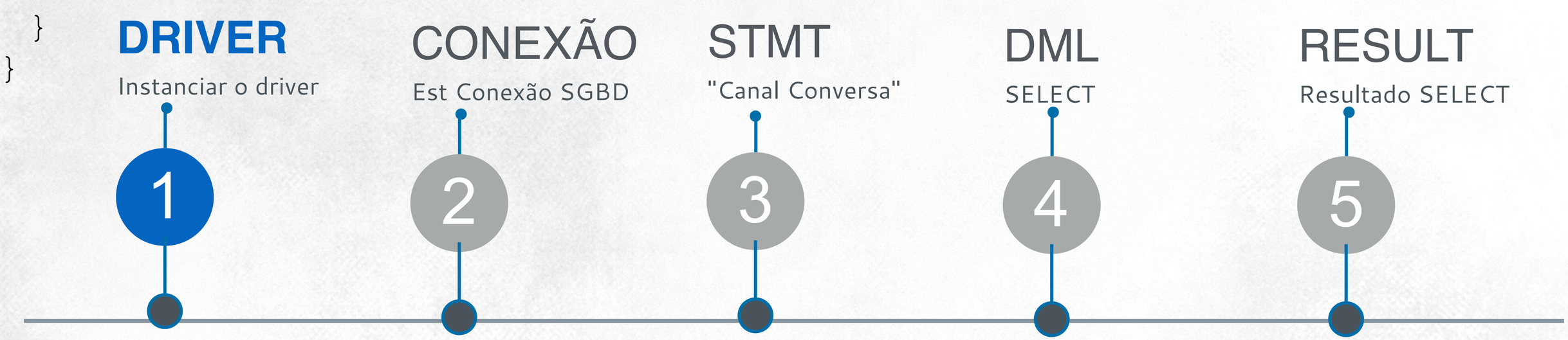
```
public class UniCarioca {
 public static void main(String[] args) {

       Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colecaocd","aluno","unicarioca");
                Statement stmt = con.createStatement()) {
            ResultSet rs = stmt.executeQuery("select * from estilo;");
            while (rs.next()) {
                System.out.println("Estilo: " + rs.getString("nome"));
            }
        }


  }
}
```



JDBC

PASSO-A-PASSO

```
public class UniCarioca {
 public static void main(String[] args) {

      Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colecaocd","aluno","unicarioca");
                Statement stmt = con.createStatement()) {
            ResultSet rs = stmt.executeQuery("select * from estilo;");
            while (rs.next()) {
                System.out.println("Estilo: " + rs.getString("nome"));
            }
        }


  }
}
```



JDBC

PASSO-A-PASSO

```
public class UniCarioca {
 public static void main(String[] args) {

      Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colec
```

Até a versão 3 do JDBC era necessário instanciar o driver do banco. A partir do JDBC 4 (Java 6) este passo não é mais necessário.

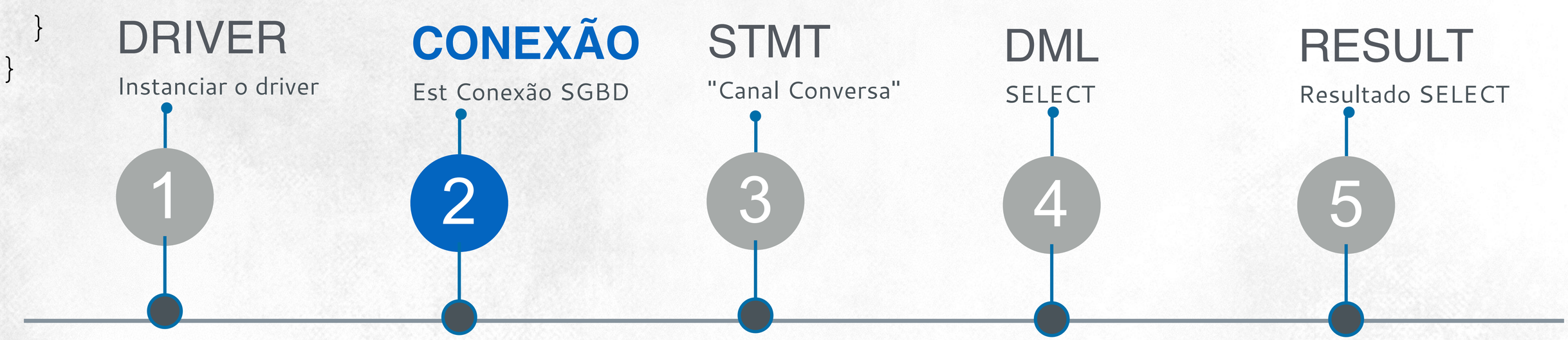
No caso de aplicações Web, veja a documentação dos servidor de aplicação, como o Tomcat 7.

1. **DRIVER** — Instanciar o driver
2. CONEXÃO — Est Conexão SGBD
3. STMT — "Canal Conversa"
4. DML — SELECT
5. RESULT — Resultado SELECT

JDBC

PASSO-A-PASSO

```
public class UniCarioca {
 public static void main(String[] args) {

       Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colecaocd","aluno","unicarioca");
                Statement stmt = con.createStatement()) {
            ResultSet rs = stmt.executeQuery("select * from estilo;");
            while (rs.next()) {
                System.out.println("Estilo: " + rs.getString("nome"));
            }
        }


  }
}
```



JDBC

DriverManager.getConnection("jdbc:mysql://localhost:8889/colecaocd","aluno","unicarioca")

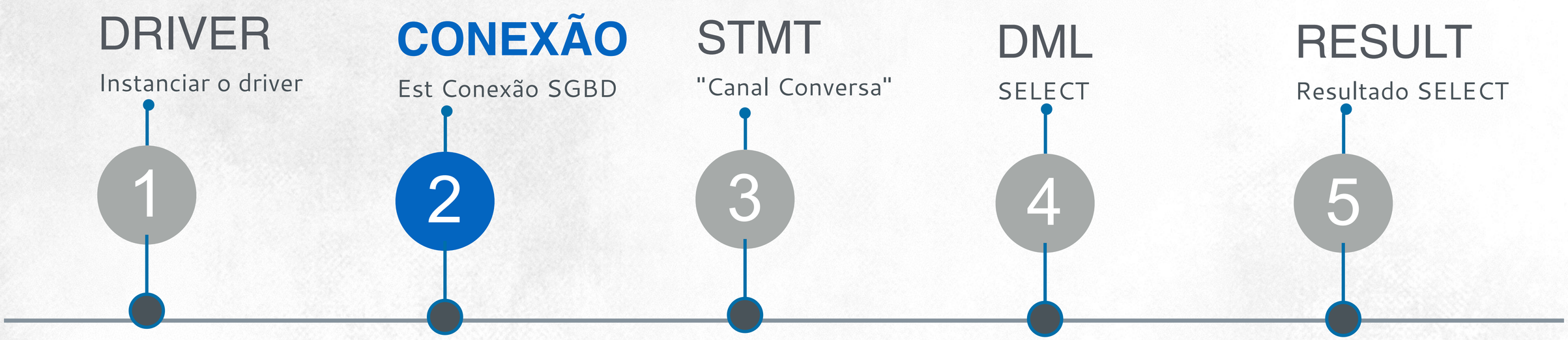
1. **ENDEREÇO IP DO SERVIDOR DO BANCO**
   DriverManager.getConnection("jdbc:mysql://**localhost**:8889/colecaocd","aluno","unicarioca")

2. **PORTA DO SERVIDOR DO BANCO (3306)**
   DriverManager.getConnection("jdbc:mysql://localhost:**8889**/colecaocd","aluno","unicarioca")

3. **BASE DE DADOS**
   DriverManager.getConnection("jdbc:mysql://localhost:8889/**colecaocd**","aluno","unicarioca")

4. **USUÁRIO**
   DriverManager.getConnection("jdbc:mysql://localhost:8889/colecaocd","**aluno**","unicarioca")

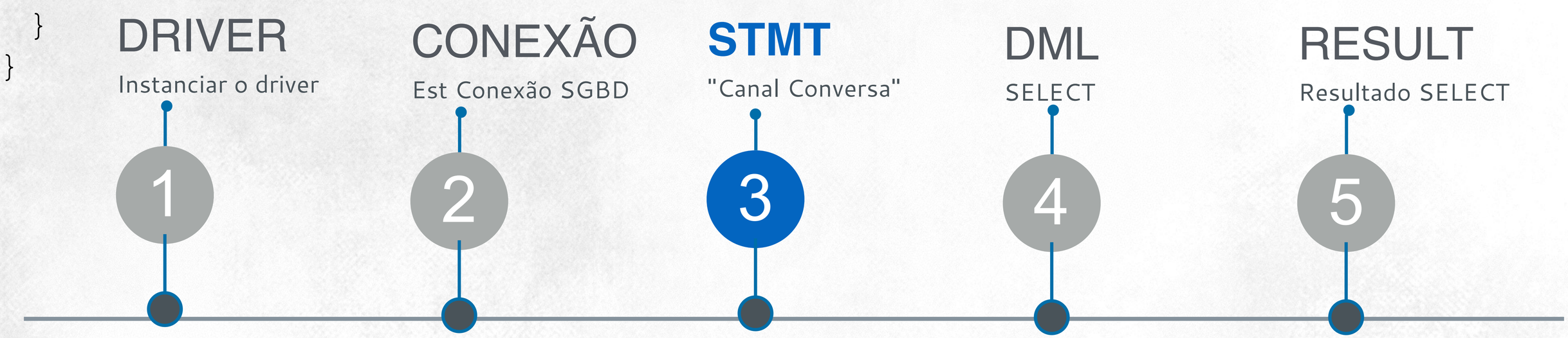
5. **SENHA**
   DriverManager.getConnection("jdbc:mysql://localhost:8889/colecaocd","aluno","**unicarioca**")



JDBC

PASSO-A-PASSO

```
public class UniCarioca {
 public static void main(String[] args) {

       Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colecaocd","aluno","unicarioca");
                Statement stmt = con.createStatement()) {
            ResultSet rs = stmt.executeQuery("select * from estilo;");
            while (rs.next()) {
                System.out.println("Estilo: " + rs.getString("nome"));
            }
        }


  }
}
```



JDBC

PASSO-A-PASSO

```
public class UniCarioca {
 public static void main(String[] args) {

       Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colecaocd","aluno","unicarioca");
                Statement stmt = con.createStatement()) {
            ResultSet rs = stmt.executeQuery("select * from estilo;");
            while (rs.next()) {
                System.out.println("Estilo: " + rs.getString("nome"));
            }
        }


  }
}
```

DRIVER
Instanciar o driver
1

CONEXÃO
Est Conexão SGBD
2

STMT
"Canal Conversa"
3

DML
SELECT
4

RESULT
Resultado SELECT
5

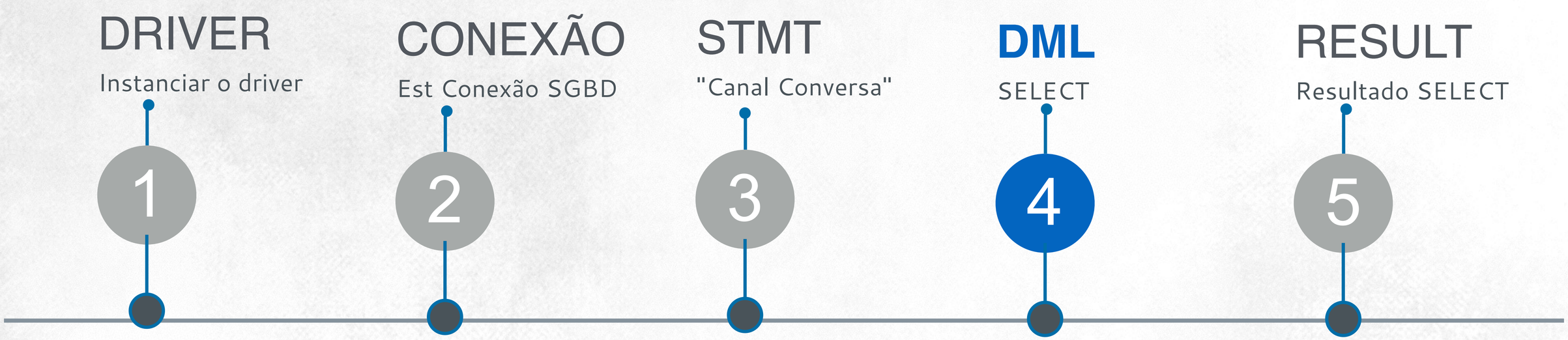
JDBC

PASSO-A-PASSO

1) **executeQuery**
   a) SELECT
   b) Retorna o conjunto de dados recuperados no banco

2) **executeUpdate**
   a) INSERT, DELETE e UPDATE
   b) Retorna a quantidade de linhas afetadas pelo SQL

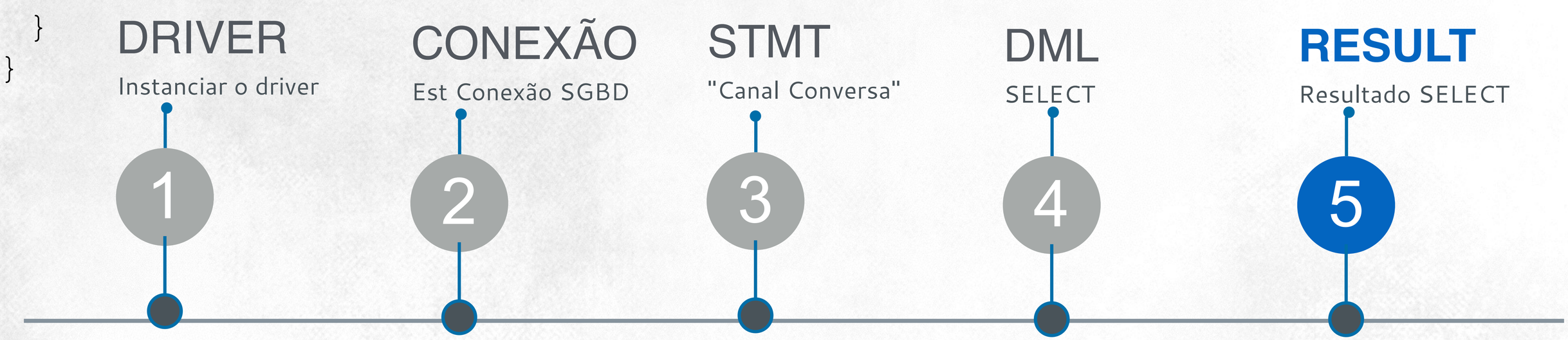
3) **execute**
   a) Retorna um verdadeiro se um conjunto de dados do banco pode ser retornado



JDBC

PASSO-A-PASSO

```
public class UniCarioca {
 public static void main(String[] args) {

       Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colecaocd","aluno","unicarioca");
                Statement stmt = con.createStatement()) {
            ResultSet rs = stmt.executeQuery("select * from estilo;");
            while (rs.next()) {
                System.out.println("Estilo: " + rs.getString("nome"));
            }
        }


    }
}
```

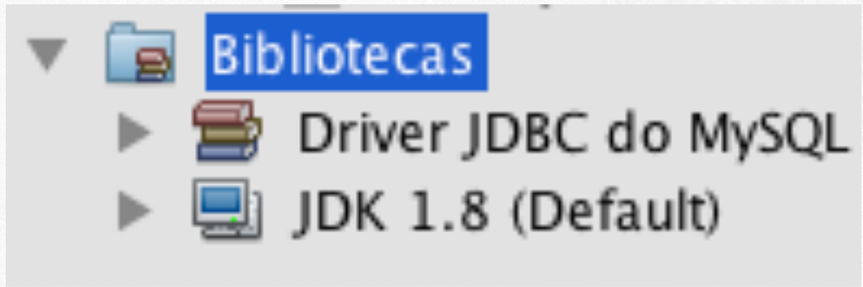


JDBC

ERROS MAIS COMUNS

- Exception in thread "main" java.lang.ClassNotFoundException:

  com.mysql.jdbc.Driver



Exception in thread "main" java.sql.SQLException: Access denied for user 'aluno'@'localhost' (using password: YES)

☐	aluno	%	Yes	SELECT	No	Edit Privileges	Export
☐	aluno	localhost	Yes	SELECT	No	Edit Privileges	Export

JDBC

EXEMPLOS

```
public class Aula5 {
 public static void main(String[] args) {

       Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colecaocd","aluno","unicarioca");
            Statement stmt = con.createStatement()) {
            int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");
            System.out.println("Foram alterados " + regAlterados + "
registros");
        }
 }
}
```

JDBC

EXEMPLOS

```
public class Aula5 {
 public static void main(String[] args) {

       Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colecaocd","aluno","unicarioca");
            Statement stmt = con.createStatement()) {
            int regAlterados = stmt.executeUpdate("delete from
estilo;");
            System.out.println("Foram alterados " + regAlterados + "
registros");
        }
 }
}
```

JDBC

EXEMPLOS

```
public class Aula5 {
 public static void main(String[] args) {

       Class.forName("com.mysql.jdbc.Driver");
        try (Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:8889/
colecaocd","aluno","unicarioca");
            Statement stmt = con.createStatement()) {
            int regAlterados = stmt.executeUpdate("update estilo set
nome='samba';");
            System.out.println("Foram alterados " + regAlterados + "
registros");
        }
 }
}
```

PREPAREDSTATEMENT

DESCRIÇÃO

- possui todos os recursos da interface Statement;

- métodos da interface PreparedStatement são:
  - *execute()*; (true: ResultSet; false: número ou não tem resultado)
  - executeQuery(); (ResultSet)
  - executeUpdate(); (Número linhas alteradas ou 0)
  - *setString():* atribuir valores do tipo Alfa Numéricos.

PREPARED STATEMENT

CONSULTA

```
String queryStr = "SELECT * FROM Program WHERE Name = ? and Cost < ?";

PreparedStatement pstmt = con.prepareStatement(queryStr);

pstmt.setString(1, “Unfogging the Future”);
pstmt.setInt(2, 1000);

ResultSet rs = pstmt.executeQuery();
```

PREPARED STATEMENT

CONSULTA

```
String queryStr = "SELECT * FROM Program WHERE Name = ? and Cost < ?”;

PreparedStatement pstmt = con.prepareStatement(queryStr);

pstmt.setString(1, “Unfogging the Future”);
pstmt.setInt(2, 1000);

ResultSet rs = pstmt.executeQuery();
```

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
```

PREPARED STATEMENT

CONSULTA

```
String queryStr = "SELECT * FROM Program WHERE Name = ? and Cost < ?";

PreparedStatement pstmt =     con.prepareStatement(queryStr);

pstmt.setString(1, “Unfogging the Future”);
pstmt.setInt(2, 1000);

ResultSet rs = pstmt.executeQuery();
```

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
```

PREPARED STATEMENT

CONSULTA

```
String queryStr = "SELECT * FROM Program WHERE Name = ? and Cost < ?";

PreparedStatement pstmt =     con.prepareStatement(queryStr);

pstmt.setString(1, “Unfogging the Future”);
pstmt.setInt(2, 1000);

ResultSet rs = pstmt.executeQu
```

Passagem de Parâmetros

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
```

PREPARED STATEMENT

CONSULTA

```
String queryStr = "SELECT * FROM Program WHERE Name = ? and Cost < ?";

PreparedStatement pstmt =     con.prepareStatement(queryStr);

pstmt.setString(1, “Unfogging the Future”);
pstmt.setInt(2, 1000);

ResultSet rs = pstmt.executeQu
```

Passagem de Parâmetros

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
```

PREPARED STATEMENT

CONSULTA

```
String queryStr = "SELECT * FROM Program WHERE Name = ? and Cost < ?";

PreparedStatement pstmt =     con.prepareStatement(queryStr);

pstmt.setString(1, “Unfogging the Future”);
pstmt.setInt(2, 1000);

ResultSet rs = pstmt.executeQu
```

Passagem de Parâmetros

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
```

PREPARED STATEMENT

CONSULTA

```
String queryStr = "SELECT * FROM Program WHERE Name = ? and Cost < ?";

PreparedStatement pstmt = con.prepareStatement(queryStr);

pstmt.setString(1, “Unfogging the Future”);
pstmt.setInt(2, 1000);

ResultSet rs = pstmt.executeQuery();
```

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
```

PREPARED STATEMENT

CONSULTA

```
String queryStr = "SELECT * FROM Program WHERE Name = ? and Cost < ?";

PreparedStatement pstmt = con.prepareStatement(queryStr);

pstmt.setString(1, “Unfogging the Future”);
pstmt.setInt(2, 1000);

ResultSet rs = pstmt.executeQuery();
```

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
```

PREPARED STATEMENT

CONSULTA

```
String queryStr = "SELECT * FROM Program WHERE Name = ? and Cost < ?";

PreparedStatement pstmt = con.prepareStatement(queryStr);

pstmt.setString(1, “Unfogging the Future”);
pstmt.setInt(2, 1000);

ResultSet rs = pstmt.executeQuery();
```

Passagem de Parâmetros

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
```

PREPARED STATEMENT

CONSULTA

```
String queryStr = "SELECT * FROM Program WHERE Name = ? and Cost < ?";

PreparedStatement pstmt = con.prepareStatement(queryStr);

pstmt.setString(1, “Unfogging the Future”);
pstmt.setInt(2, 1000);

ResultSet rs = pstmt.executeQuery();
```

Passagem de Parâmetros

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
```

PREPARED STATEMENT

EXERCÍCIOS

```
CREATE TABLE pessoa (
    nome VARCHAR(40),
    email VARCHAR(80),
    cpf VARCHAR(14),
    telefone VARCHAR(30),
    sexo CHAR(1),
    datanascimento DATE);
```

Implemente um projeto em Java para fazer Inserção, Remoção, Atualização e Listagem.

DAO

IV

4.1 Descrição
4.2 Estrutura
4.3 Implementação

DAO

INTRODUÇÃO

- Simplificar e desacoplar a interação das aplicações Java com SGBD;

- Todas as comunicações com o mecanismo de persistência são mediadas por um objeto: o DAO.

DAO

INTRODUÇÃO

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");

     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
}
```

DAO

INTRODUÇÃO

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");

     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
}
```

DAO

INTRODUÇÃO

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");

     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
}
```

Fábrica de Conexões

Acesso ao Dado

Retorno do Dado

DAO

INTRODUÇÃO

Aula5.java

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");


     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
}
```

DAO

INTRODUÇÃO

Aula5.java

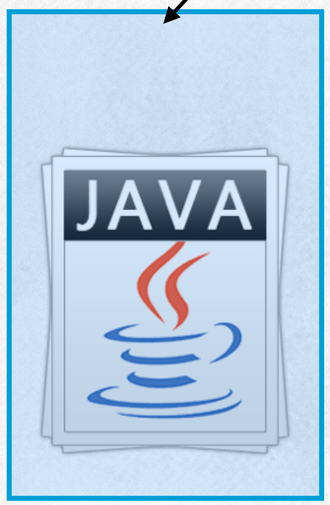
```
public class Aula5 {
 public static void main(String[] args) {

    try{
      Connection con = (Connection)
      DriverManager.getConnection("jdbc:mysql://localhost:3306/
      colecaocd","aluno","unicarioca");

      Statement stmt = con.createStatement();

      int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

      System.out.println("Foram alterados " + regAlterados + "
registros");
      }catch(Exception e){
       e.printStackTrace();
      }
}
```

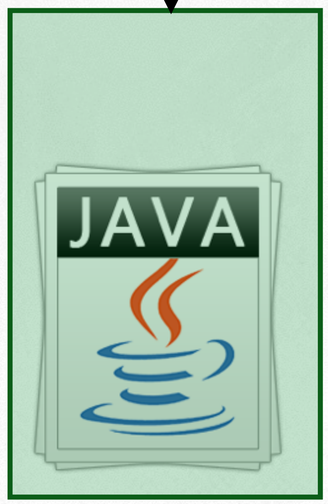
DAO

INTRODUÇÃO

Aula5.java

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");

     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
 }
```



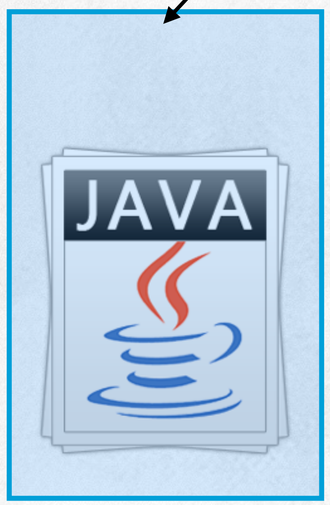


DAO

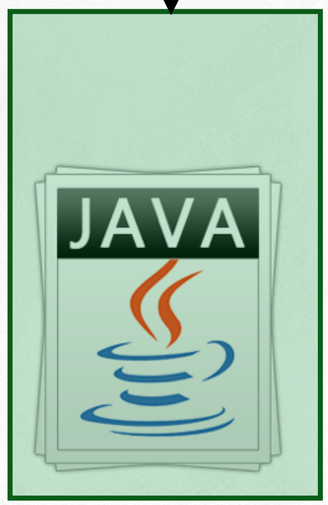
INTRODUÇÃO

Aula5.java

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");

     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
 }
```



FabricaConexoes.java



Estilo.java

EstiloDAO.java

DAO

fabricaconexoes.java

```
public class FabricaConexoes {
    private String servidor, base, login, senha;

    public FabricaConexoes()
    {
        servidor = "localhost";
        base = "base";
        login = "aluno";
        senha ="unicarioca";
    }

   public Connection Conexao() throws ClassNotFoundException,
  SQLException
    {
        Class.forName("com.mysql.jdbc.Driver");
        return (Connection)
  DriverManager.getConnection("jdbc:mysql://" +getServidor()+ ":
  3306/" + getBase(), getLogin(),    getSenha());
    }
}
```

DAO

fabricaconexoes.java

```
public class FabricaConexoes {
    private String servidor, base, login, senha;

    public FabricaConexoes()
    {
        servidor = "localhost";
        base = "base";
        login = "aluno";
        senha ="unicarioca";
    }

   public Connection Conexao() throws ClassNotFoundException,
  SQLException
    {
        Class.forName("com.mysql.jdbc.Driver");
        return (Connection)
  DriverManager.getConnection("jdbc:mysql://" +getServidor()+ ":
  3306/" + getBase(), getLogin(),    getSenha());
    }
}
```

DAO

fabricaconexoes.java

```
public class FabricaConexoes {
    private String servidor, base, login, senha;

    public FabricaConexoes()
    {
        servidor = "localhost";
        base = "base";
        login = "aluno";
        senha ="unicarioca";
    }

    public Connection Conexao() throws ClassNotFoundException,
  SQLException
    {
        Class.forName("com.mysql.jdbc.Driver");
        return (Connection)
  DriverManager.getConnection("jdbc:mysql://" +getServidor()+ ":
  3306/" + getBase(), getLogin(),    getSenha());
    }
}
```

getters & setters

DAO

fabricaconexoes.java

```
public class FabricaConexoes {
    private String servidor, base, login, senha;

    public FabricaConexoes()
    {
        servidor = "localhost";
        base = "base";
        login = "aluno";
        senha ="unicarioca";
    }

    public Connection Conexao() throws ClassNotFoundException,
  SQLException
    {
        Class.forName("com.mysql.jdbc.Driver");
        return (Connection)
  DriverManager.getConnection("jdbc:mysql://" +getServidor()+ ":
  3306/" + getBase(), getLogin(),    getSenha());
    }
}
```

DAO

fabricaconexoes.java

```
public class FabricaConexoes {
    private String servidor, base, login, senha;

    public FabricaConexoes()
    {
        servidor = "localhost";
        base = "base";
        login = "aluno";
        senha ="unicarioca";
    }

  public Connection Conexao() throws ClassNotFoundException,
 SQLException
    {
        Class.forName("com.mysql.jdbc.Driver");
        return (Connection)
 DriverManager.getConnection("jdbc:mysql://" +getServidor()+ ":
 3306/" + getBase(), getLogin(),    getSenha());
    }
}
```

DAO

INTRODUÇÃO

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");

     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
}
```

Fábrica de Conexões

Acesso ao Dado

Retorno do Dado

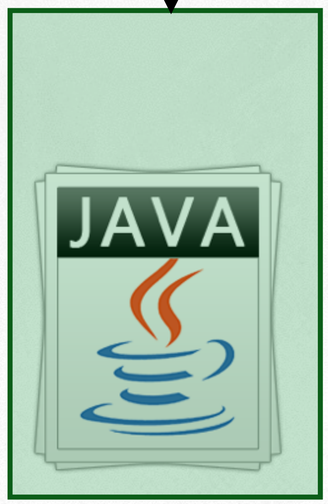
DAO

INTRODUÇÃO

Aula5.java

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");

     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
 }
```

FabricaConexoes.java



Estilo.java

EstiloDAO.java

DAO

estilo.java

```
public class Estilo{
     private String nome;

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

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

}
```

DAO

estilo.java

```
public class Estilo{
     private String nome;

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

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

}
```

DAO

estilo.java

```
public class Estilo{
     private String nome;

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

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

}
```

DAO

INTRODUÇÃO

Aula5.java

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");

     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
 }
```

FabricaConexoes.java

Estilo.java

EstiloDAO.java

DAO

INTRODUÇÃO

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");

     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
}
```

Fábrica de Conexões

Acesso ao Dado

Retorno do Dado

DAO

estilodao.java

```
public boolean inserir(Estilo e) throws
 ClassNotFoundException, SQLException
    {
        FabricaConexoes fabrica = new
 FabricaConexoes();
        PreparedStatement stmt =
fabrica.Conexao().prepareStatement("INSERT INTO agenda
(pessoa) VALUES (?)");

        stmt.setString(1, e.getNome());

        return stmt.execute();
    }
```

DAO

estilodao.java

```
public boolean inserir(Estilo e) throws
 ClassNotFoundException, SQLException
    {
        FabricaConexoes fabrica = new
 FabricaConexoes();
        PreparedStatement stmt =
fabrica.Conexao().prepareStatement("INSERT INTO agenda
(pessoa) VALUES (?)");

        stmt.setString(1, e.getNome());

        return stmt.execute();
    }
```

DAO

estilodao.java

```
public boolean inserir(Estilo e) throws
 ClassNotFoundException, SQLException
    {
        FabricaConexoes fabrica = new
 FabricaConexoes();
        PreparedStatement stmt =
fabrica.Conexao().prepareStatement("INSERT INTO agenda
(pessoa) VALUES (?)");

        stmt.setString(1, e.getNome());

        return stmt.execute();
    }
```

DAO

estilodao.java

```
public boolean inserir(Estilo e) throws
 ClassNotFoundException, SQLException
    {
        FabricaConexoes fabrica = new
 FabricaConexoes();
        PreparedStatement stmt =
fabrica.Conexao().prepareStatement("INSERT INTO agenda
(pessoa) VALUES (?)");

        stmt.setString(1, e.getNome());

        return stmt.execute();
    }
```

DAO

estilodao.java

```
public boolean inserir(Estilo e) throws
 ClassNotFoundException, SQLException
    {
        FabricaConexoes fabrica = new
 FabricaConexoes();
        PreparedStatement stmt =
fabrica.Conexao().prepareStatement("INSERT INTO agenda
(pessoa) VALUES (?)");

        stmt.setString(1, e.getNome());

        return stmt.execute();
    }
```

DAO

EXERCÍCIOS

```
CREATE TABLE pessoa (
    nome VARCHAR(40),
    email VARCHAR(80),
    cpf VARCHAR(14),
    telefone VARCHAR(30),
    sexo CHAR(1),
    datanascimento DATE);
```

Implemente um projeto em Java para fazer Inserção, Remoção, Atualização e Listagem, usando DAO.

DAO

resposta

Aula5.java

```
public class Aula5 {
 public static void main(String[] args) {

    try{
     Connection con = (Connection)
     DriverManager.getConnection("jdbc:mysql://localhost:3306/
     colecaocd","aluno","unicarioca");

     Statement stmt = con.createStatement();

     int regAlterados = stmt.executeUpdate("insert into estilo
('samba');");

     System.out.println("Foram alterados " + regAlterados + "
registros");
     }catch(Exception e){
      e.printStackTrace();
     }
 }
```

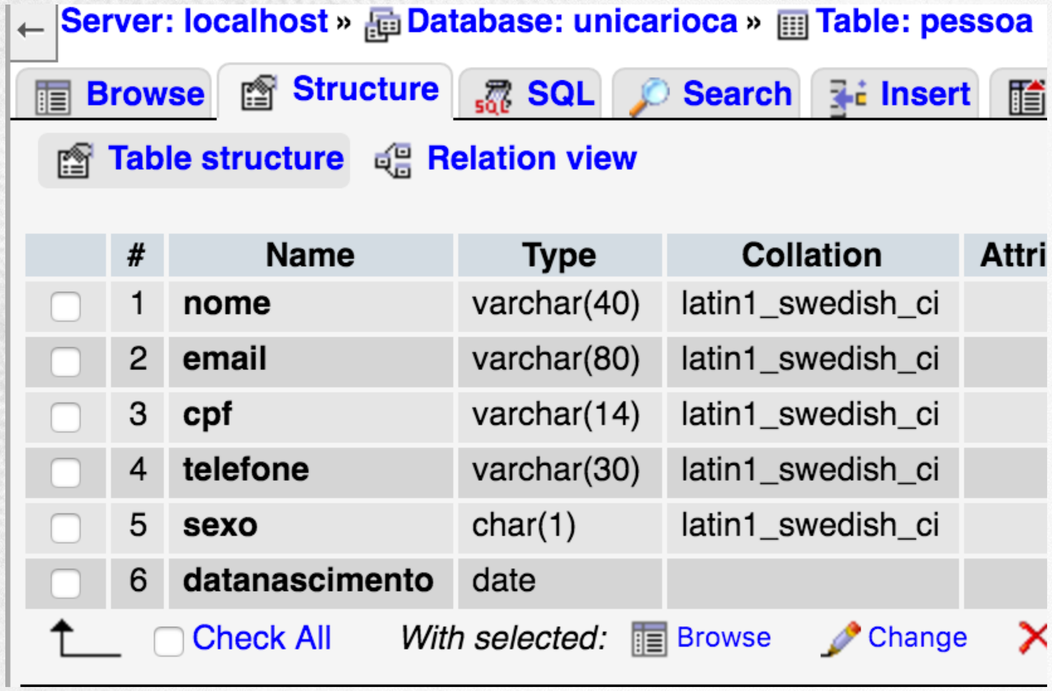
FabricaConexoes.java

Estilo.java

EstiloDAO.java

DAO

resposta: base de dados

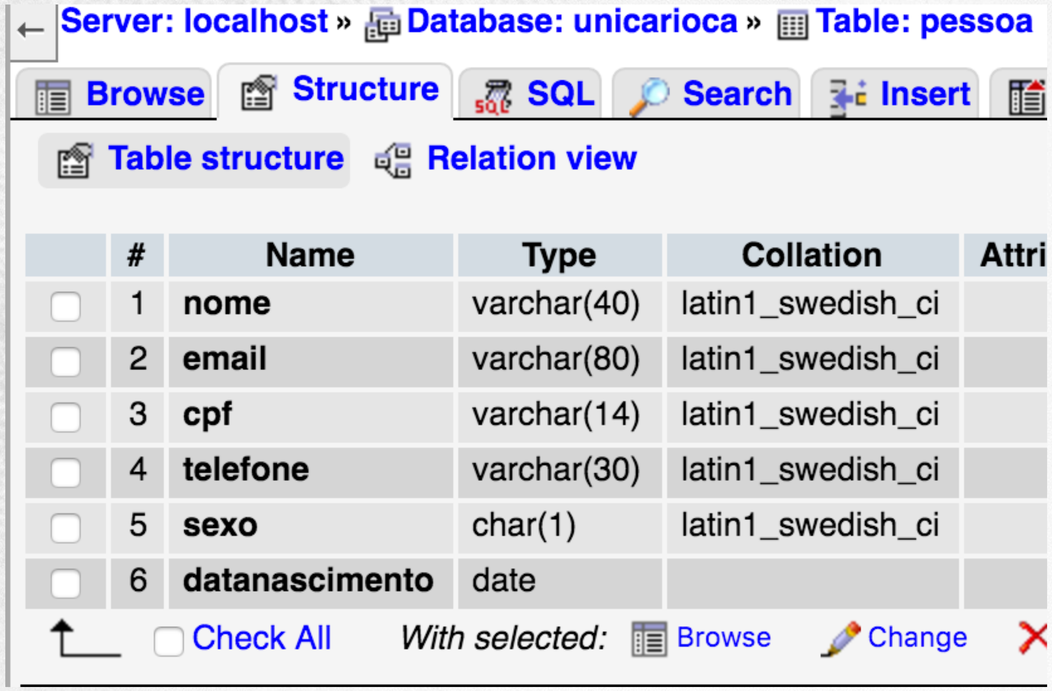


#	Name	Type	Collation	Attri
1	nome	varchar(40)	latin1_swedish_ci	
2	email	varchar(80)	latin1_swedish_ci	
3	cpf	varchar(14)	latin1_swedish_ci	
4	telefone	varchar(30)	latin1_swedish_ci	
5	sexo	char(1)	latin1_swedish_ci	
6	datanascimento	date		

```
CREATE TABLE pessoa (
    nome VARCHAR(40),
    email VARCHAR(80),
    cpf VARCHAR(14),
    telefone VARCHAR(30),
    sexo CHAR(1),
    datanascimento DATE);
```

DAO

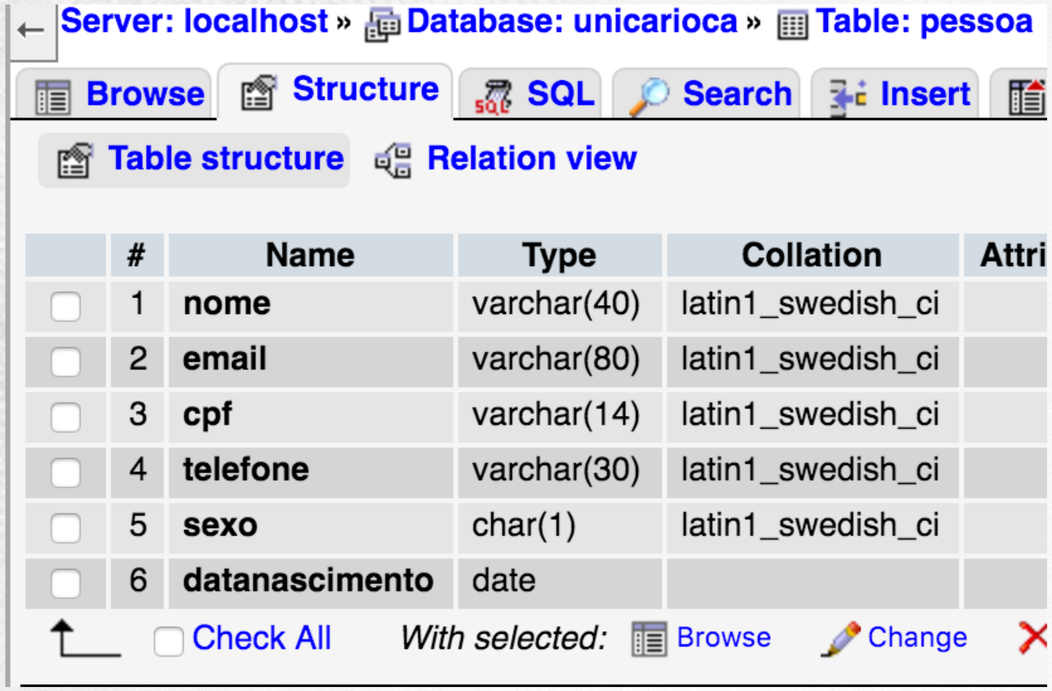
resposta: base de dados




```
CREATE TABLE pessoa (
    nome VARCHAR(40),
    email VARCHAR(80),
    cpf VARCHAR(14),
    telefone VARCHAR(30),
    sexo CHAR(1),
    datanascimento DATE);
```

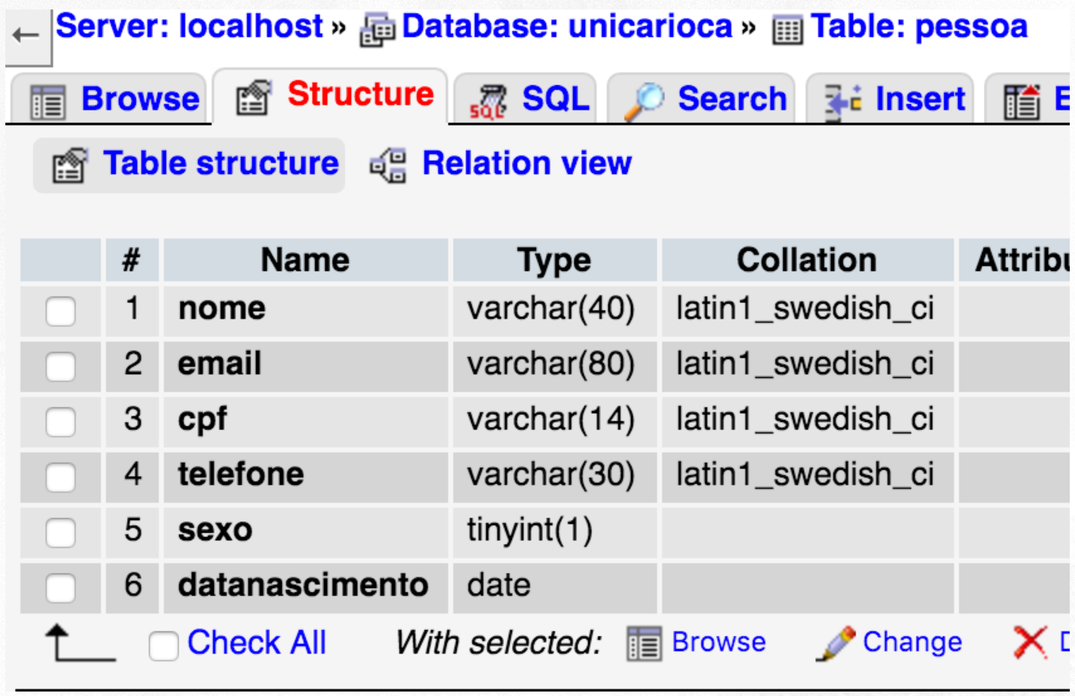
DAO

resposta: base de dados



#	Name	Type	Collation	Attri
1	nome	varchar(40)	latin1_swedish_ci	
2	email	varchar(80)	latin1_swedish_ci	
3	cpf	varchar(14)	latin1_swedish_ci	
4	telefone	varchar(30)	latin1_swedish_ci	
5	sexo	char(1)	latin1_swedish_ci	
6	datanascimento	date		

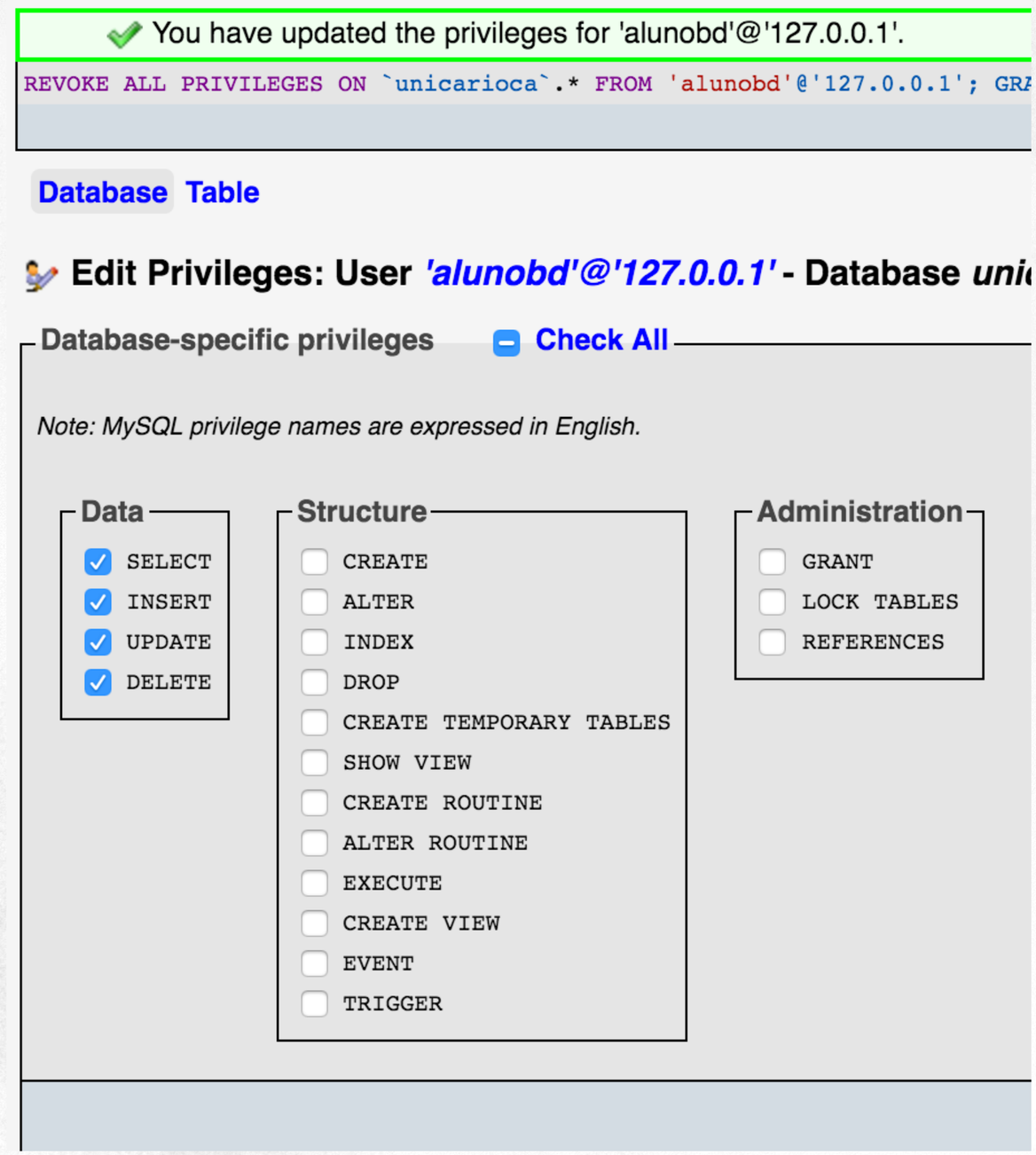
```
CREATE TABLE pessoa (
    nome VARCHAR(40),
    email VARCHAR(80),
    cpf VARCHAR(14),
    telefone VARCHAR(30),
    sexo CHAR(1),
    datanascimento DATE);
```



#	Name	Type	Collation	Attribu
1	nome	varchar(40)	latin1_swedish_ci	
2	email	varchar(80)	latin1_swedish_ci	
3	cpf	varchar(14)	latin1_swedish_ci	
4	telefone	varchar(30)	latin1_swedish_ci	
5	sexo	tinyint(1)		
6	datanascimento	date		

DAO

resposta: base de dados



DAO

resposta: FabricaConexoes.java

```java
public class FabricaConexoes {
    private String servidor, porta, base, login, senha;

    public FabricaConexoes(){
        this.servidor = "127.0.0.1";
        this.porta = "8889";
        this.base = "unicarioca";
        this.login = "alunobd";
        this.senha = "unicarioca";
    }
```

DAO

resposta: FabricaConexoes.java

```
public Connection Conexao() throws ClassNotFoundException, SQLException
{
    return (Connection) DriverManager.getConnection("jdbc:mysql://" +getServidor()+ ":"+
            getPorta() +"/" + getBase(), getLogin(), getSenha());
}
```

DAO

resposta: Pessoa.java

```
 */
public class Pessoa {
    private String nome, email, cpf, telefone;
    private boolean sexo;
    private Date  datanascimento;

    /**
     * @return the nome
     */
    public String getNome() {
        return nome;
    }

    /**
     * @param nome the nome to set
     */
    public void setNome(String nome) {
        this.nome = nome;
    }
```

DAO

resposta: Pessoa.java

```
public boolean isSexo() {
    return sexo;
}

public String getSexo(){
    return sexo?"M":"F";
}

/**
 * @param sexo the sexo to set
 */
public void setSexo(boolean sexo) {
    this.sexo = sexo;
}
```

DAO

resposta: Pessoa.java

```java
public Date getDatanascimento() {
    return datanascimento;
}

/**
 * @return the datanascimento
 */
public String getDatanascimentoString() {
    String data = new SimpleDateFormat("dd/MM/yyyy").format(this.datanascimento);

    return data;
}
```

DAO

resposta: Pessoa.java

```
@Override
public String toString(){
    String PessoaString = "Nome: " + this.nome;
    PessoaString += "\n";
    PessoaString += "CPF: " + this.cpf;
    PessoaString += "\n";
    PessoaString += "Telefone: " + this.telefone;
    PessoaString += "\n";
    PessoaString += "Sexo: " + this.getSexo();
    PessoaString += "\n";
    PessoaString += "Data de Nascimento: " + this.getDatanascimentoString();
    PessoaString += "\n";
    PessoaString += "E-mail: " + this.email;
    PessoaString += "\n";

    return PessoaString;
}
```

DAO

resposta: PessoaDAO.java

```java
public class PessoaDAO {

    Connection banco;

    public PessoaDAO() throws ClassNotFoundException, SQLException {
        FabricaConexoes fabrica = new FabricaConexoes();
        banco = fabrica.Conexao();
    }

    @Override
    @SuppressWarnings("FinalizeDeclaration")
    protected void finalize() throws SQLException, Throwable
    {
        try {
            banco.close();
        } finally {
            super.finalize();
        }
    }
```

DAO

resposta: PessoaDAO.java

```
public int inserir(Pessoa p) throws ClassNotFoundException, SQLException {
    PreparedStatement stmt = banco.prepareStatement("INSERT INTO pessoa (nome, "
            + "email, cpf, telefone, sexo, datanascimento) VALUES (?, ?, ?, ?, ?, ?)");

    java.util.Date data = p.getDatanascimento();
    java.sql.Date databanco = new java.sql.Date(data.getTime());

    stmt.setString(1, p.getNome());
    stmt.setString(2, p.getEmail());
    stmt.setString(3, p.getCpf());
    stmt.setString(4, p.getTelefone());
    stmt.setBoolean(5, p.isSexo());
    stmt.setDate(6, databanco);
    return stmt.executeUpdate();
}
```

DAO

resposta: PessoaDAO.java

```
public class Inserir {
    public static void main(String[] args) throws ParseException, ClassNotFoundException, SQLException{
        Pessoa p = new Pessoa();
        p.setNome("Anderson");
        p.setEmail("asantos@unicarioca.edu.br");
        p.setCpf("000.000.000-00");
        p.setTelefone("9999-8888");
        p.setSexo(true);
        Date datanascimento = new SimpleDateFormat("yyyy-MM-dd").parse("1974-01-01");
        p.setDatanascimento(datanascimento);


        PessoaDAO operacao = new PessoaDAO();

        int resultado = operacao.inserir(p);

        System.out.println(resultado==1?"INSERIU":"NÃO INSERIU");

    }
}
```

DAO

resposta: PessoaDAO.java

```
public int apagarPeloNome(Pessoa p) throws SQLException{
    PreparedStatement stmt = banco.prepareStatement("DELETE FROM pessoa WHERE nome LIKE ?");
    stmt.setString(1, "%"+p.getNome()+"%");
    return stmt.executeUpdate();
}
```

DAO

resposta: PessoaDAO.java

```
public class Apagar {
    public static void main(String[] args) throws ClassNotFoundException, SQLException{
        Pessoa p = new Pessoa();
        p.setNome("Anderson");
        PessoaDAO operacao = new PessoaDAO();
        int resultado = operacao.apagarPeloNome(p);
        System.out.println(resultado>0?"APAGOU":"NÃO APAGOU");
    }
}
```

DAO

resposta: PessoaDAO.java

```
public int atualizarTelefone(Pessoa p, String telefone) throws SQLException{
    PreparedStatement stmt = banco.prepareStatement("UPDATE pessoa SET telefone=? WHERE nome LIKE
    stmt.setString(1, telefone);
    stmt.setString(2, "%"+p.getNome()+"%");
    return stmt.executeUpdate();
}
```

DAO

resposta: PessoaDAO.java

```
public class Atualizar {
    public static void main(String[] args) throws ClassNotFoundException, SQLException
    {
        Pessoa p = new Pessoa();
        p.setNome("Anderson");

        String telefone = "8888-9999";

        PessoaDAO operacao = new PessoaDAO();
        int resultado = operacao.atualizarTelefone(p, telefone);

        System.out.println(resultado>0?"ATUALIZOU":"NÃO ATUALIZOU");
    }
}
```

DAO

resposta: PessoaDAO.java

```java
public ArrayList listar() throws SQLException{
    PreparedStatement stmt = banco.prepareStatement("SELECT * FROM pessoa");
    ArrayList<Pessoa> lista = new ArrayList<Pessoa>();

    ResultSet resultado = stmt.executeQuery();
    Pessoa p;
    while(resultado.next()){
        p = new Pessoa();
        p.setNome(resultado.getString("nome"));
        p.setDatanascimento(resultado.getDate("datanascimento"));
        p.setCpf(resultado.getString("cpf"));
        p.setEmail(resultado.getString("email"));
        p.setSexo(resultado.getBoolean("sexo"));
        p.setTelefone(resultado.getString("telefone"));
        lista.add(p);
    }

    return lista;
}
```

DAO

resposta: PessoaDAO.java

```
public class Listar {
    public static void main(String[] args) throws ClassNotFoundException, SQLException
    {
        PessoaDAO operacao = new PessoaDAO();
        ArrayList lista = operacao.listar();
        Iterator indice = lista.iterator();
        Pessoa p = new Pessoa();
        int i=0;
        while(indice.hasNext())
        {
            System.out.println((Pessoa) lista.get(i++));
            indice.next();
        }
    }
}
```

Exercícios de Revisão

EXERCÍCIOS

Correlacione as colunas:

(1) ALTER () Apaga um índice de uma tabela

(2) CREATE () Cria um novo gatilho

(3) DELETE () Atualiza um registro

(4) DROP () Insere linhas em uma tabela

(5) INSERT () Altera a estrutura de uma tabela

(6) RENAME () Recupera um conjunto de dados de uma tabela

(7) SELECT () Renomeia uma tabela

(8) TRUNCATE () Apaga linhas de uma tabela

(9) UPDATE () Apaga a estrutura de uma tabela

EXERCÍCIOS

Correlacione as colunas:

(1) ALTER	(1, 4 e 8) Apaga um índice de uma tabela
(2) CREATE	(2) Cria um novo gatilho
(3) DELETE	(9) Atualiza um registro
(4) DROP	(5) Insere linhas em uma tabela
(5) INSERT	(1) Altera a estrutura de uma tabela
(6) RENAME	(7) Recupera um conjunto de dados de uma tabela
(7) SELECT	(6) Renomeia uma tabela
(8) TRUNCATE	(3) Apaga linhas de uma tabela
(9) UPDATE	(8) Apaga a estrutura de uma tabela

EXERCÍCIOS

Considere:

```
CREATE TABLE `livro` (
  `idlivro` int(11) NOT NULL AUTO_INCREMENT,
  `titulo` varchar(100) NOT NULL,
  `preco` float DEFAULT NULL,
  `estoque` int(11) DEFAULT '0',
  `idgenero` int(11) NOT NULL,
  `ideditora` int(11) NOT NULL,
  PRIMARY KEY (`idlivro`,`idgenero`,`ideditora`)
) ENGINE=MyISAM AUTO_INCREMENT=10 DEFAULT CHARSET=latin1;
```

1) Java - Como Programar **Valor: R$ 369,00**

Servidor: 216.58.222.78 Porta: 3306 Base: saraiva
Usuario: JeffersonPenteado Senha: f1@m3nG0

Considere:

```
public class UniCarioca {
  public static void main(String[] args) {
        String conexao = “jdbc:mysql://216.58.222.78:3306/saraiva”;
        String usuario = "JeffersonPenteado";
        String senha = "f1@m3nGO";
        String driver = “com.mysql.jdbc.Driver”;
        String SQL = “SELECT titulo, preco FROM livro ORDER BY titulo";
        Connection conexao = null;
        int i=0;
        DecimalFormat df = new DecimalFormat("0.00");
       try{
             Class.forName(driver);
            con = (Connection) DriverManager.getConnection(conexao, usuario,
       senha);
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(SQL);
             while (rs.next()) {
                  System.out.println(String.valueOf(++i) + ") " + rs.getString("titulo") + "   " + "R
$ " + String.valueOf(df.format(Float.valueOf(rs.getString("preco")))));
             }
       }catch(Exception e){
           e.printStackTrace();
        }
   }
}
```

Considere:

```
public class UniCarioca {
  public static void main(String[] args) {
        String conexao = “jdbc:mysql://216.58.222.78:3306/saraiva”;
        String usuario = "JeffersonPenteado";
        String senha = "f1@m3nGO";
        String driver = “com.mysql.jdbc.Driver”;
        String SQL = “SELECT titulo, preco FROM livro ORDER BY titulo";
        Connection conexao = null;
        int i=0;
        DecimalFormat df = new DecimalFormat("0.00");
       try{
             Class.forName(driver);
            con = (Connection) DriverManager.getConnection(conexao, usuario,
       senha);
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(SQL);
             while (rs.next()) {
                 System.out.println(String.valueOf(++i) + ") " + rs.getString("titulo") + "   " + "R
$ " + String.valueOf(df.format(Float.valueOf(rs.getString("preco")))));
             }
       }catch(Exception e){
           e.printStackTrace();
        }
   }
}
```

Considere:

```
public class UniCarioca {
  public static void main(String[] args) {
        String conexao = “jdbc:mysql://216.58.222.78:3306/saraiva”;
        String usuario = "JeffersonPenteado";
        String senha = "f1@m3nGO";
        String driver = “com.mysql.jdbc.Driver”;
        String SQL = “SELECT titulo, preco FROM livro ORDER BY titulo";
        Connection conexao = null;
        int i=0;
        DecimalFormat df = new DecimalFormat("0.00");
       try{
             Class.forName(driver);
            con = (Connection) DriverManager.getConnection(conexao, usuario,
       senha);
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(SQL);
             while (rs.next()) {
                 System.out.println(String.valueOf(++i) + ") " + rs.getString("titulo") + "   " + "R
$ " + String.valueOf(df.format(Float.valueOf(rs.getString("preco")))));
             }
       }catch(Exception e){
           e.printStackTrace();
        }
   }
}
```

Troque por Prepare Statement

Hibernate

V

HIBERNATE

INTRODUÇÃO

- Ferramenta para mapeamento O/R em Java
  - Uma das mais difundidas
- Separação entre a aplicação e o acesso ao banco
- Facilidade de uso
  - Controle de transações
  - Independência quanto ao tipo de base de dados
- Consulta de dados
  - HQL

HIBERNATE

CLASSES PERSISTENTES

Implementam entidades lógica do negócio

POJOs (Plain Old Java Object)

- Construtor padrão
- Possui métodos de acesso – gets/sets (opcional)

Possui um atributo identificador (opcional)

HIBERNATE

MAPEAMENTO

Relaciona o modelo de objetos com o modelo relacional

Arquivos de mapeamento

- Descrevem como são feitos os mapeamentos

HIBERNATE

ARQUIVO DE MAPEAMENTO

Arquivo XML

Declaração de DTD

–Útil para auto-completion

Declaração das classes

Usualmente possuem extensão .hbm.xml

```
<hibernate-mapping>
    <class name="hibernate01.Pessoa" table="pessoa" catalog="unicarioca" optimistic-
lock="version">
        <id name="indice" type="java.lang.Integer">
            <column name="indice" />
            <generator class="identity" />
        </id>
        <property name="nome" type="string">
            <column name="nome" length="40" />
        </property>
    ….
    </class>
</hibernate-mapping>
```

HIBERNATE

XML

Recomendação da W3C para gerar linguagens de marcação para necessidades especiais

- Separação do conteúdo e da formatação
- Legibilidade tanto por humanos quanto por máquinas
- Possibilidade de criação de tags sem limitação
- Criação de arquivos para validação de estrutura
- Com seu uso pode-se interligar bancos de dados distintos
- Simplicidade

HIBERNATE

ARQUIVO DE MAPEAMENTO

Declaração de classe usando o elemento *class*

- Indicação do nome da classe
- Indicação do nome da Tabela (opcional)

```
<hibernate-mapping>
    <class name="hibernate01.Pessoa" table="pessoa"
catalog="unicarioca" optimistic-lock="version">

</hibernate-mapping>
```

HIBERNATE

MAPEAMENTO DE IDENTIFICADORES

Indicação da chave primária

- nome do atributo
- tipo

Escolha do gerador da Chave

```
<hibernate-mapping>
….
        <id name="indice" type="java.lang.Integer">
            <column name="indice" />
            <generator class="identity" />
        </id>

….
</hibernate-mapping>
```

HIBERNATE

MAPEAMENTO DE PROPRIEDADES

Elemento *property*

- Indicação de nome e tipo
  - Se não explicitar coluna considera o nome da propriedade

```
<hibernate-mapping>
    …
        <property name="nome" type="string">
            <column name="nome" length="40" />
        </property>
    …
</hibernate-mapping>
```

HIBERNATE

MAPEAMENTO DE PROPRIEDADES

Tipos

- **nomes de tipos básicos do Hibernate**
  - integer, string, character, date, timestamp, float, binary, serializable, object, blob.
- **nome de tipos básicos Java**
  - int, float, char, java.lang.String, java.util.Date, java.lang.Integer, java.sql.Clob, java.util.Calendar.
- **nome de uma classe Java Serializable**

Outras propriedades

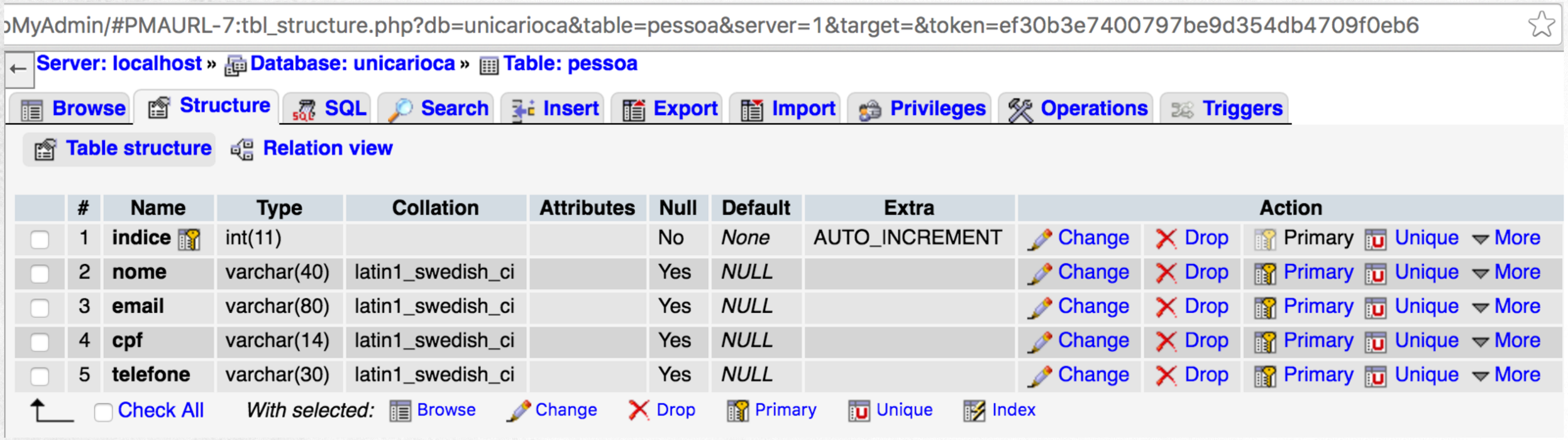
- ***Access*** *(*property | field*)*: controla como o Hibernate vai acessar os atributos em tempo de execução (default: property)
- ***column*** nome da coluna na tabela do banco (default: nome do atributo)
- ***not-null*** (true | false): determina se a propriedade pode ser nula (default: false)

HIBERNATE

EXERCÍCIOS

BANCO DE DADOS

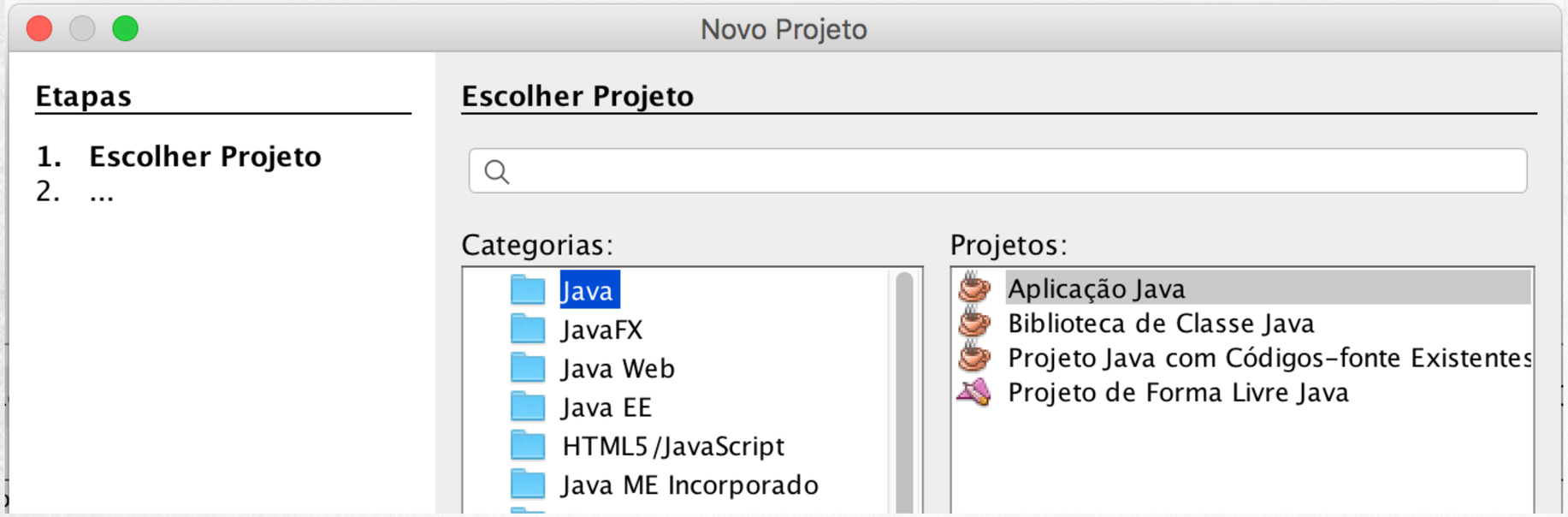
banco=**pessoa** servidor=**localhost** porta=**3306** usuário=**alunobd** senha=**unicarioca**



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action				
1	**indice**	int(11)			No	*None*	AUTO_INCREMENT	Change	Drop	Primary	Unique	More
2	**nome**	varchar(40)	latin1_swedish_ci		Yes	*NULL*		Change	Drop	Primary	Unique	More
3	**email**	varchar(80)	latin1_swedish_ci		Yes	*NULL*		Change	Drop	Primary	Unique	More
4	**cpf**	varchar(14)	latin1_swedish_ci		Yes	*NULL*		Change	Drop	Primary	Unique	More
5	**telefone**	varchar(30)	latin1_swedish_ci		Yes	*NULL*		Change	Drop	Primary	Unique	More

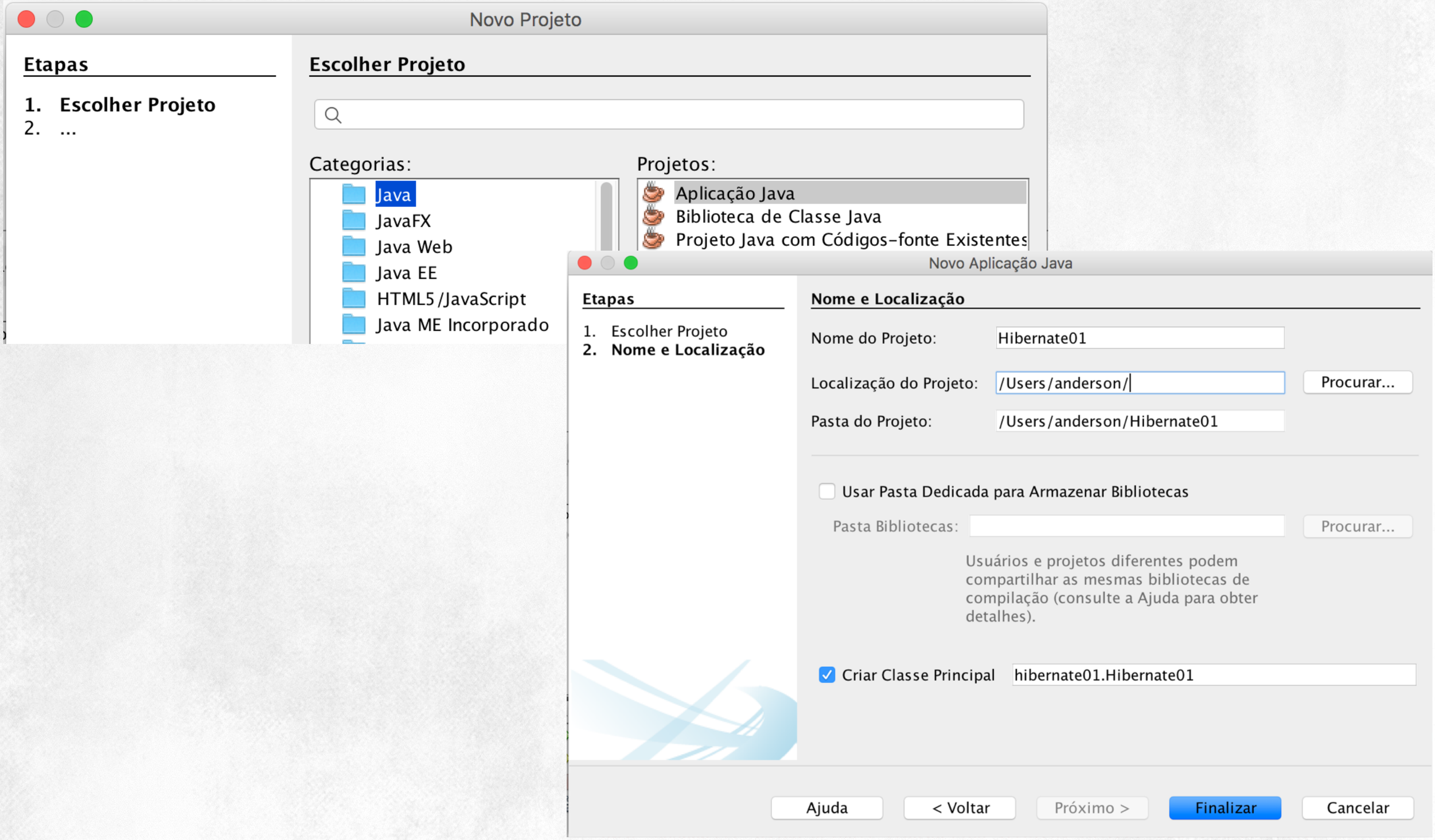
HIBERNATE

EXERCÍCIOS



HIBERNATE

EXERCÍCIOS



HIBERNATE

EXERCÍCIOS

1. **ASSISTENTE DE CONFIGURAÇÃO DO HIBERNATE**
   Informações de conexões com SGBD

2. **ASSISTENTE DE ENGENHARIA REVERSA DO HIBERNATE**
   Informações de tabelas do BD para engenharia reversa

3. **ARQUIVOS DE MAPEAMENTO DO HIBERNATE E POJOS DE BANCO DE DADOS**
   Criação das classes POJO através de engenharia reversa

4. **ASSISTENTE DE MAPEAMENTO DO HIBERNATE**
   Informações de mapeamento

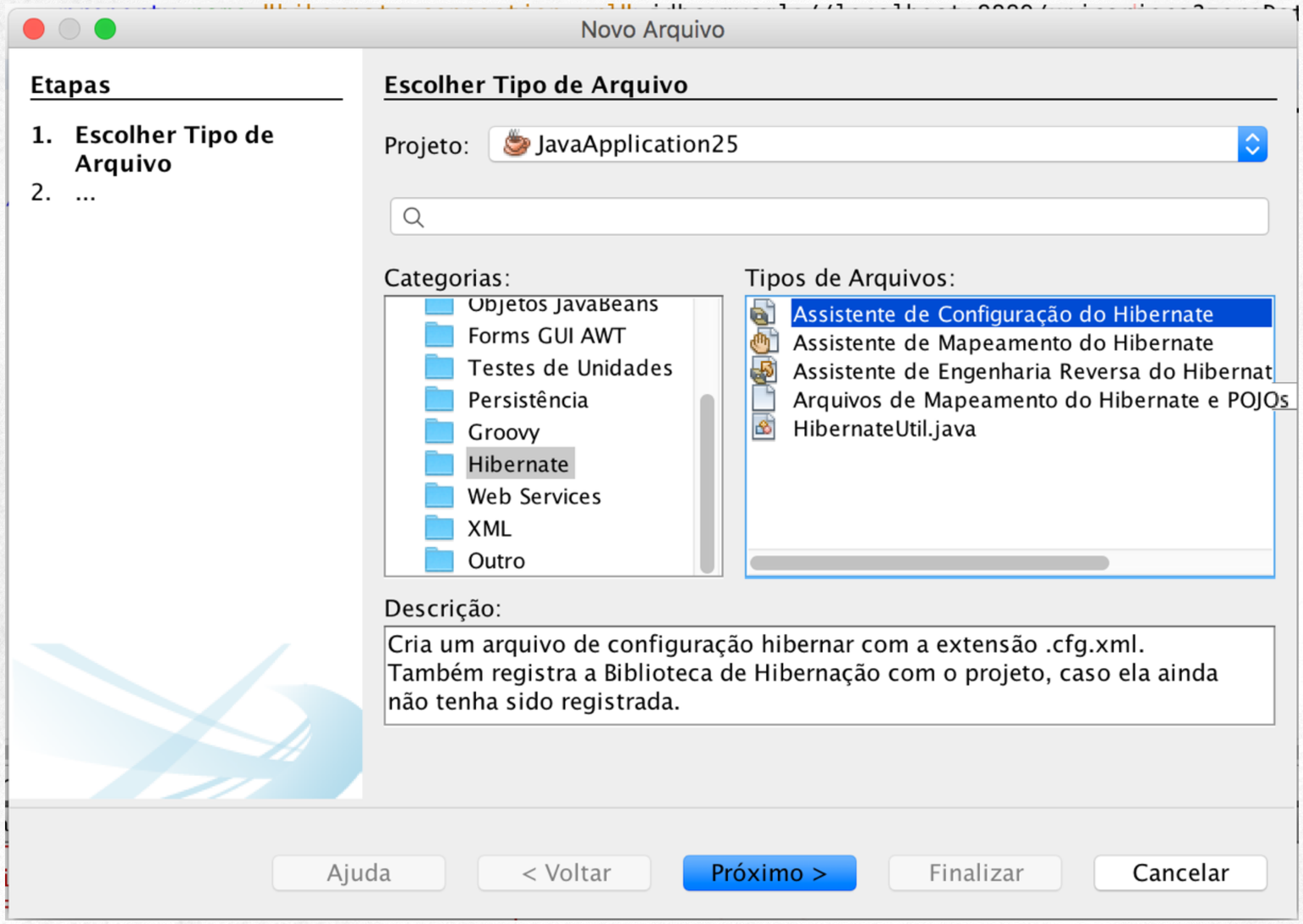
5. **HIBERNATEUTIL.JAVA**
   Arquivo para execução do hibernate

HIBERNATE

EXERCÍCIOS

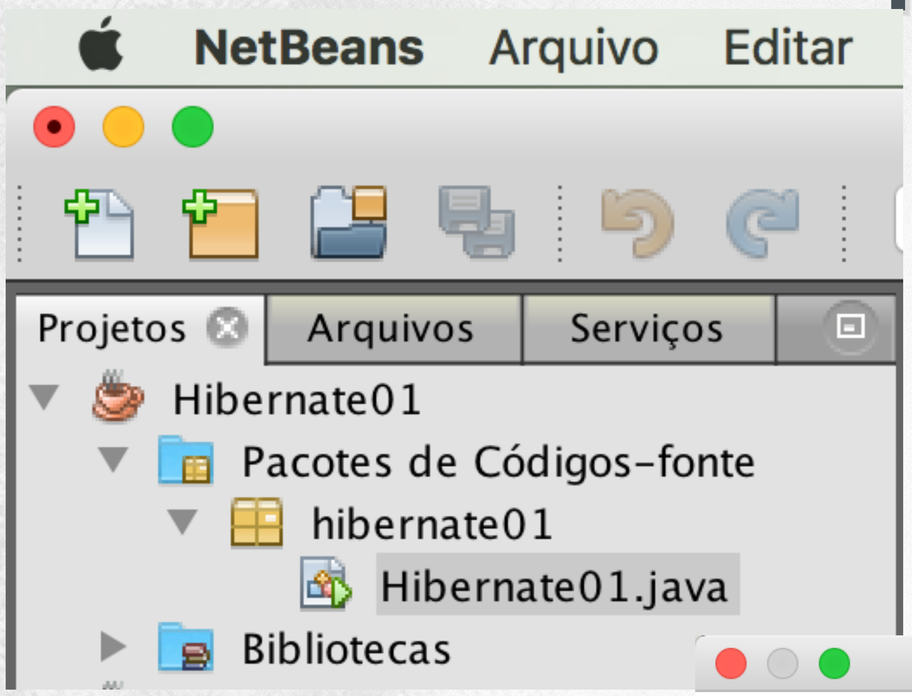
1 ASSISTENTE DE CONFIGURAÇÃO DO HIBERNATE

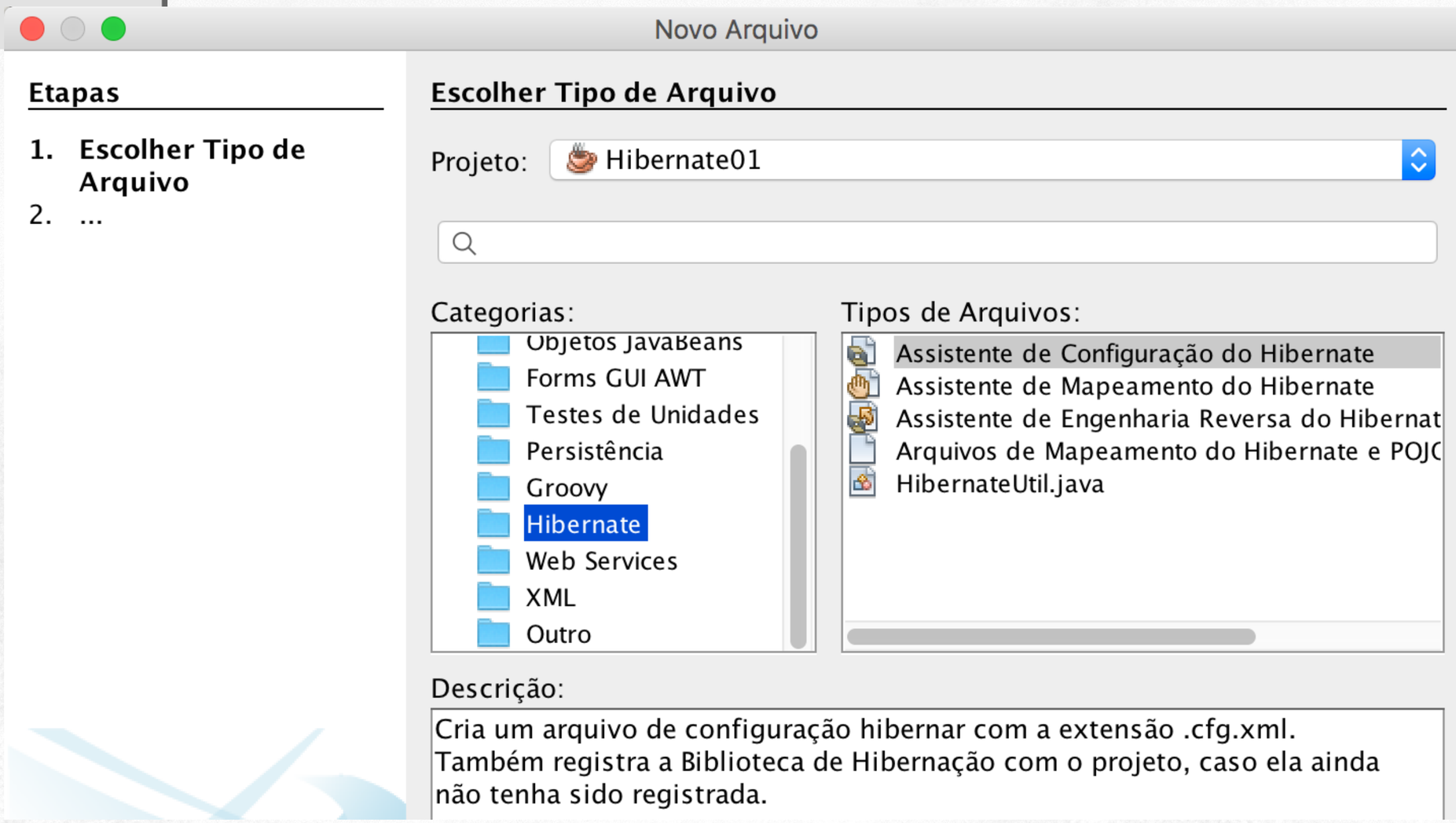
Informações de conexões com SGBD



HIBERNATE

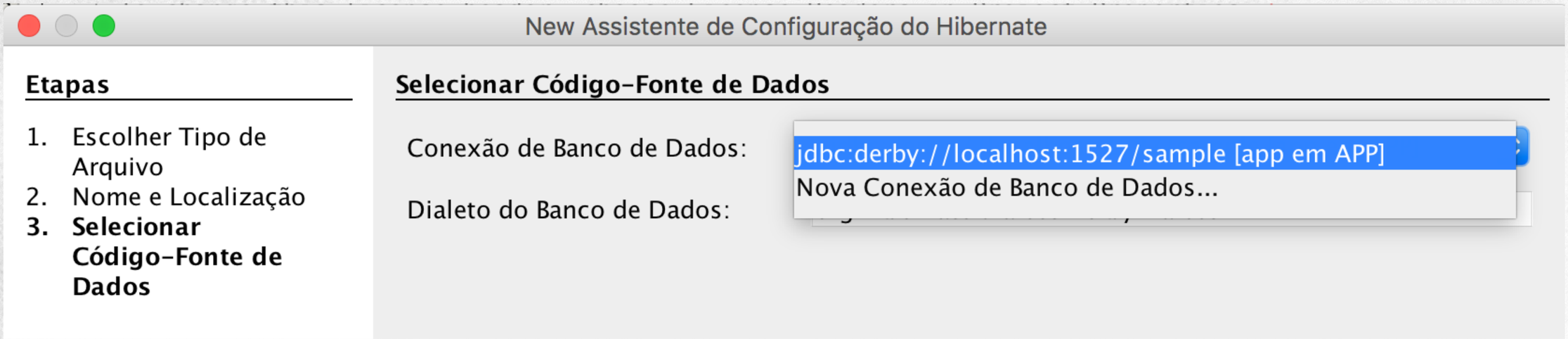
EXERCÍCIOS

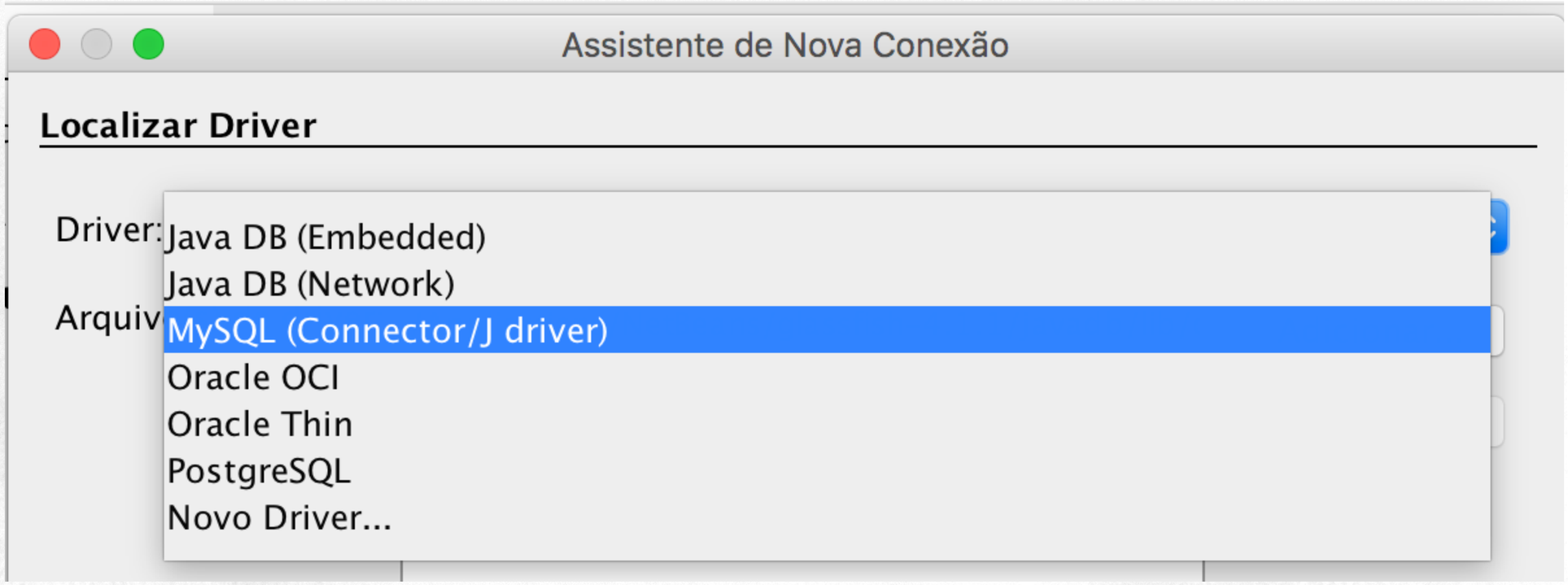




HIBERNATE

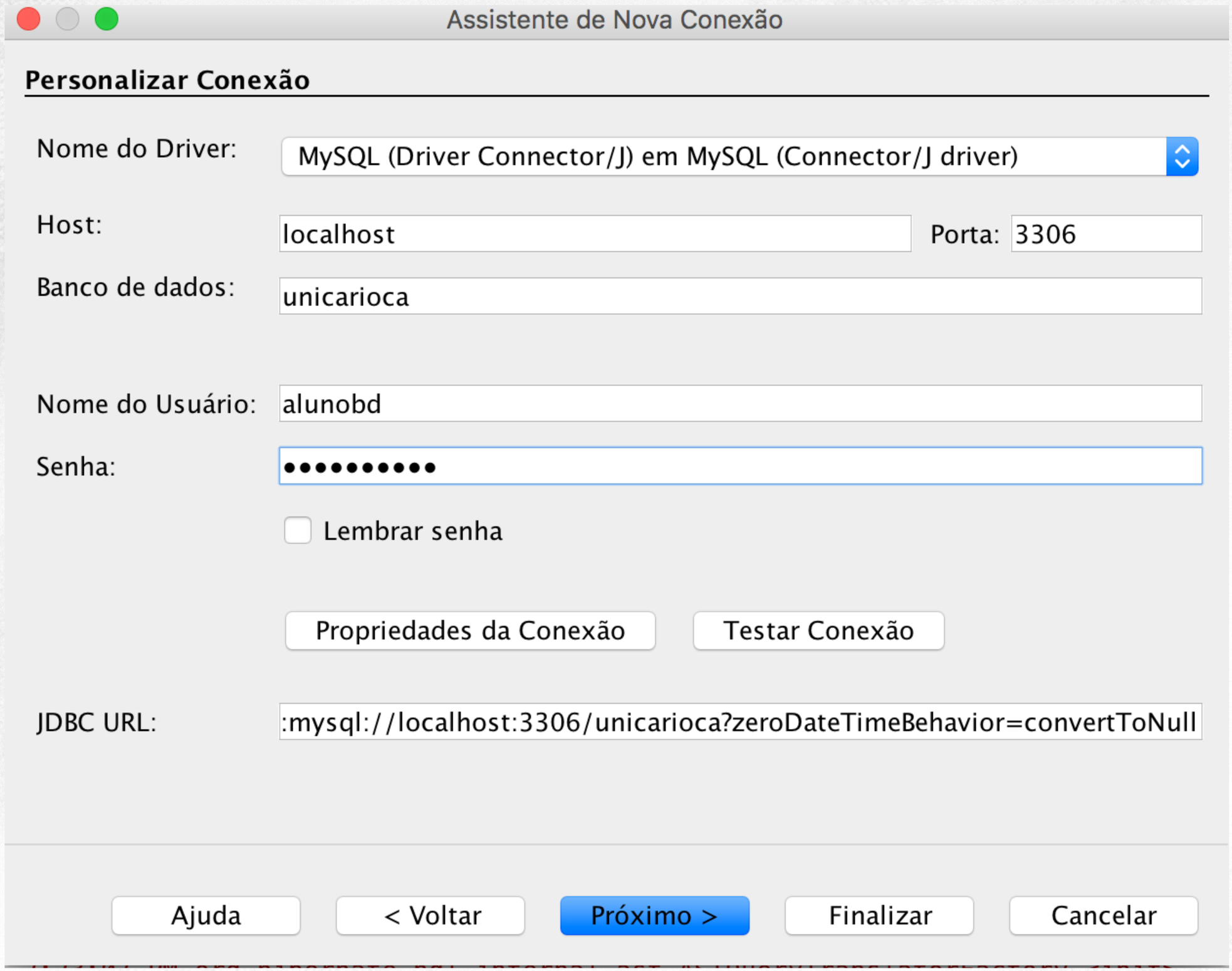
EXERCÍCIOS





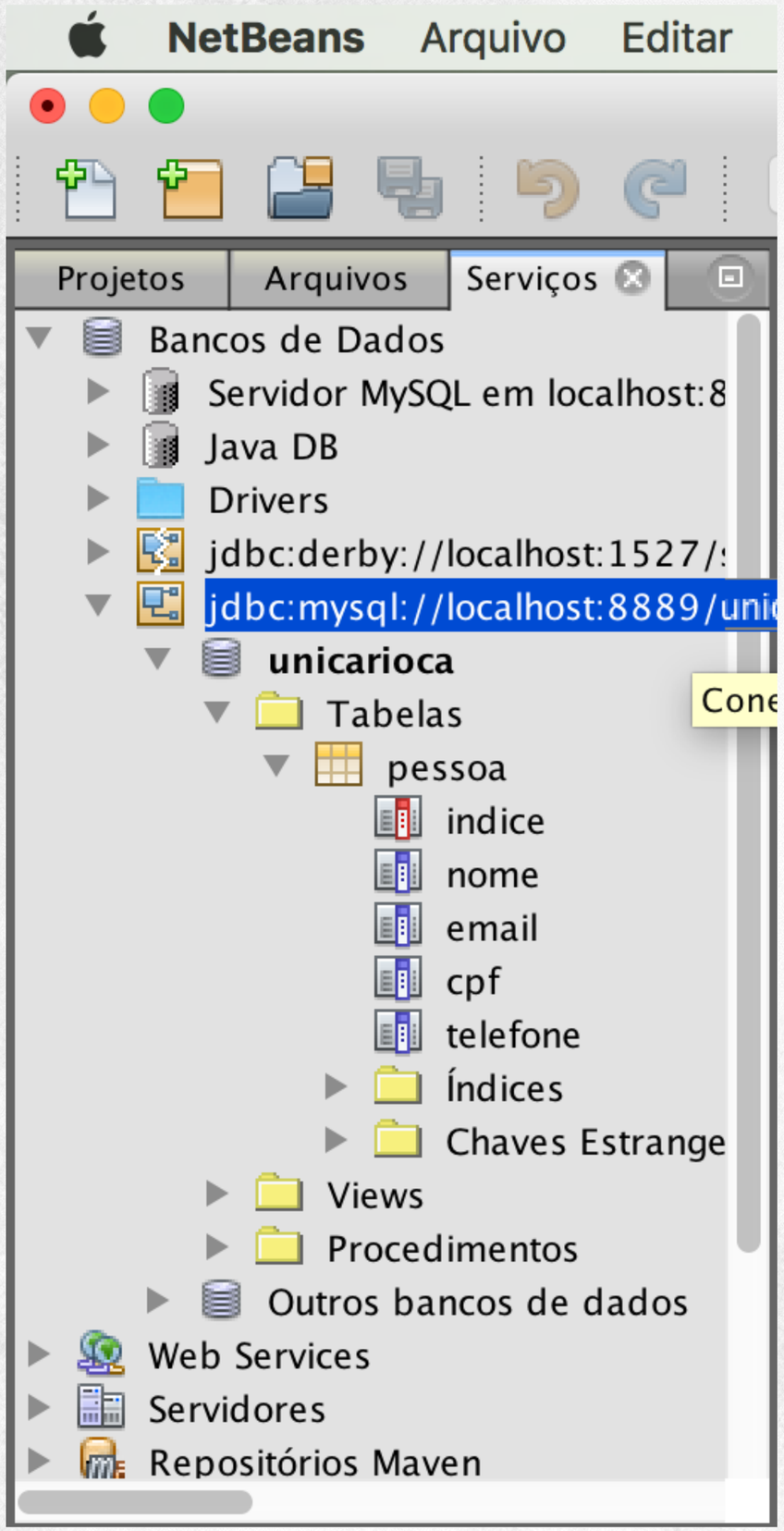
HIBERNATE

EXERCÍCIOS



HIBERNATE

EXERCÍCIOS



HIBERNATE

EXERCÍCIOS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" “http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">


<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</
property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:8889/unicarioca?
zeroDateTimeBehavior=convertToNull</property>
    <property name="hibernate.connection.username">alunobd</property>
    <property name="hibernate.connection.password">unicarioca</property>
    <property
name="hibernate.current_session_context_class">org.hibernate.context.internal.ThreadLo
calSessionContext</property>
    <mapping resource="hibernate01/Pessoa.hbm.xml"/>
  </session-factory>



</hibernate-configuration>
```

HIBERNATE

EXERCÍCIOS

1. **ASSISTENTE DE CONFIGURAÇÃO DO HIBERNATE**
   Informações de conexões com SGBD

2. **ASSISTENTE DE ENGENHARIA REVERSA DO HIBERNATE**
   Informações de tabelas do BD para engenharia reversa

3. **ARQUIVOS DE MAPEAMENTO DO HIBERNATE E POJOS DE BANCO DE DADOS**
   Criação das classes POJO através de engenharia reversa

4. **ASSISTENTE DE MAPEAMENTO DO HIBERNATE**
   Informações de mapeamento

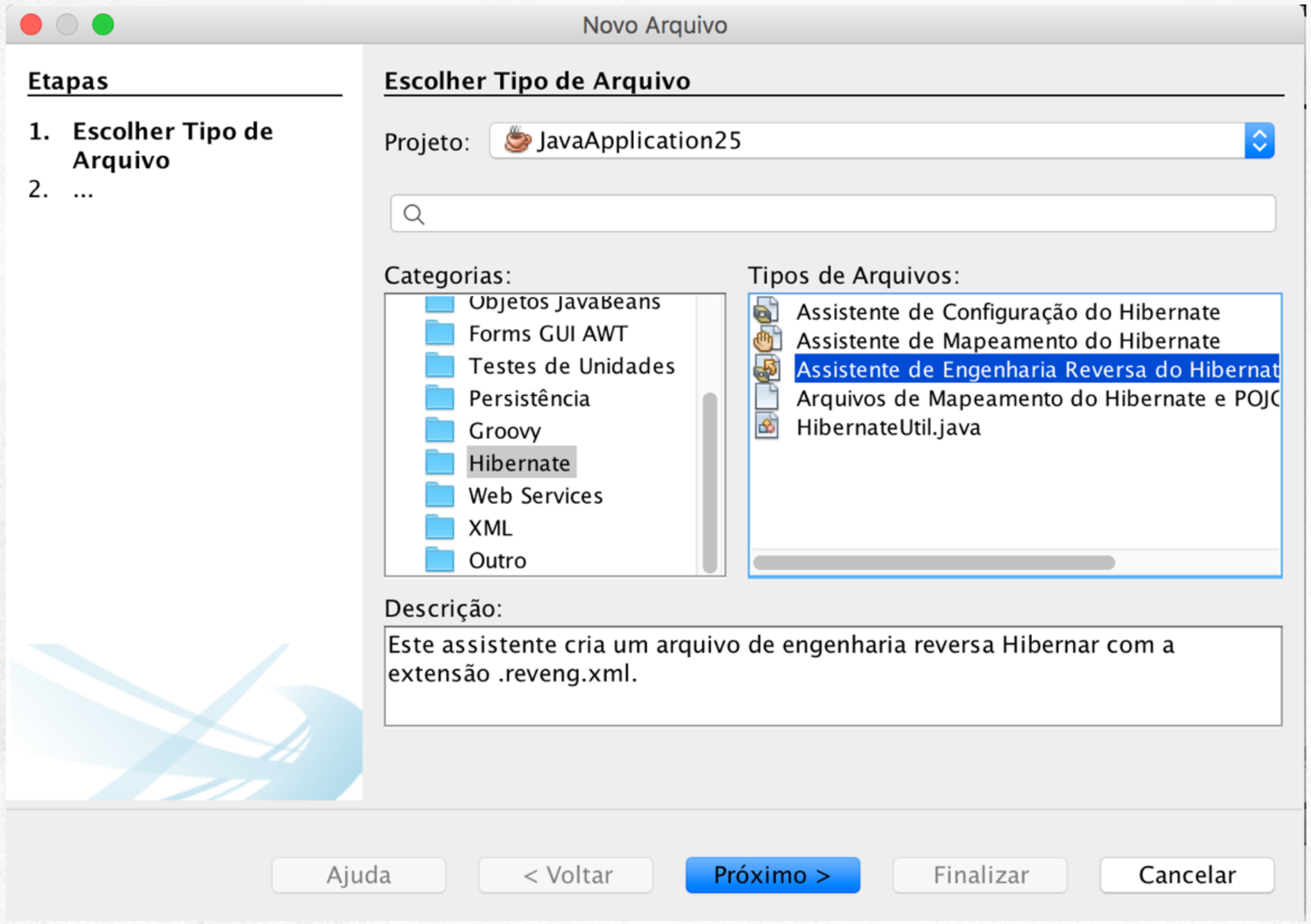
5. **HIBERNATEUTIL.JAVA**
   Arquivo para execução do hibernate

HIBERNATE

EXERCÍCIOS

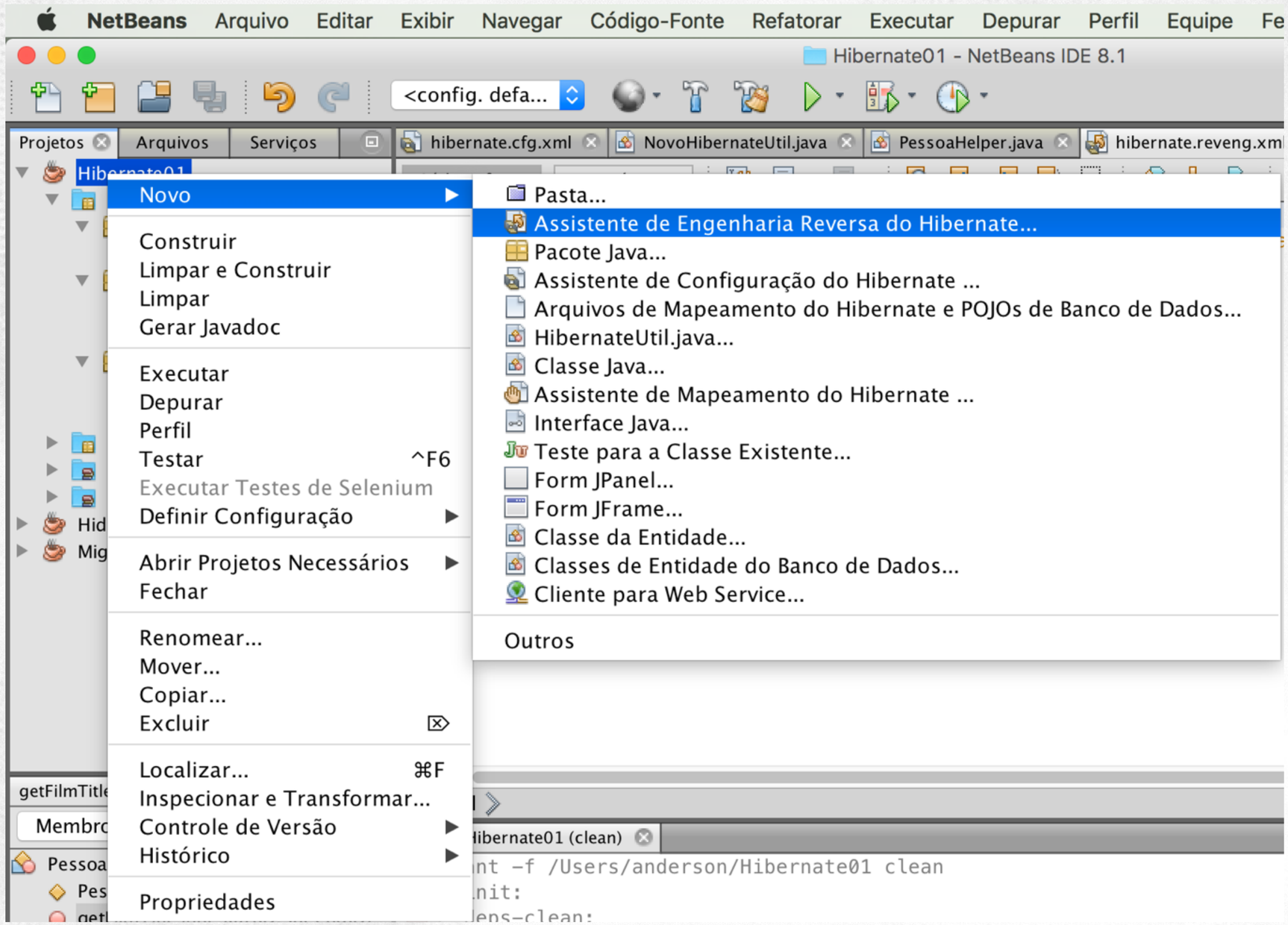
2 ASSISTENTE DE ENGENHARIA REVERSA DO HIBERNATE

Informações de tabelas do BD para engenharia reversa



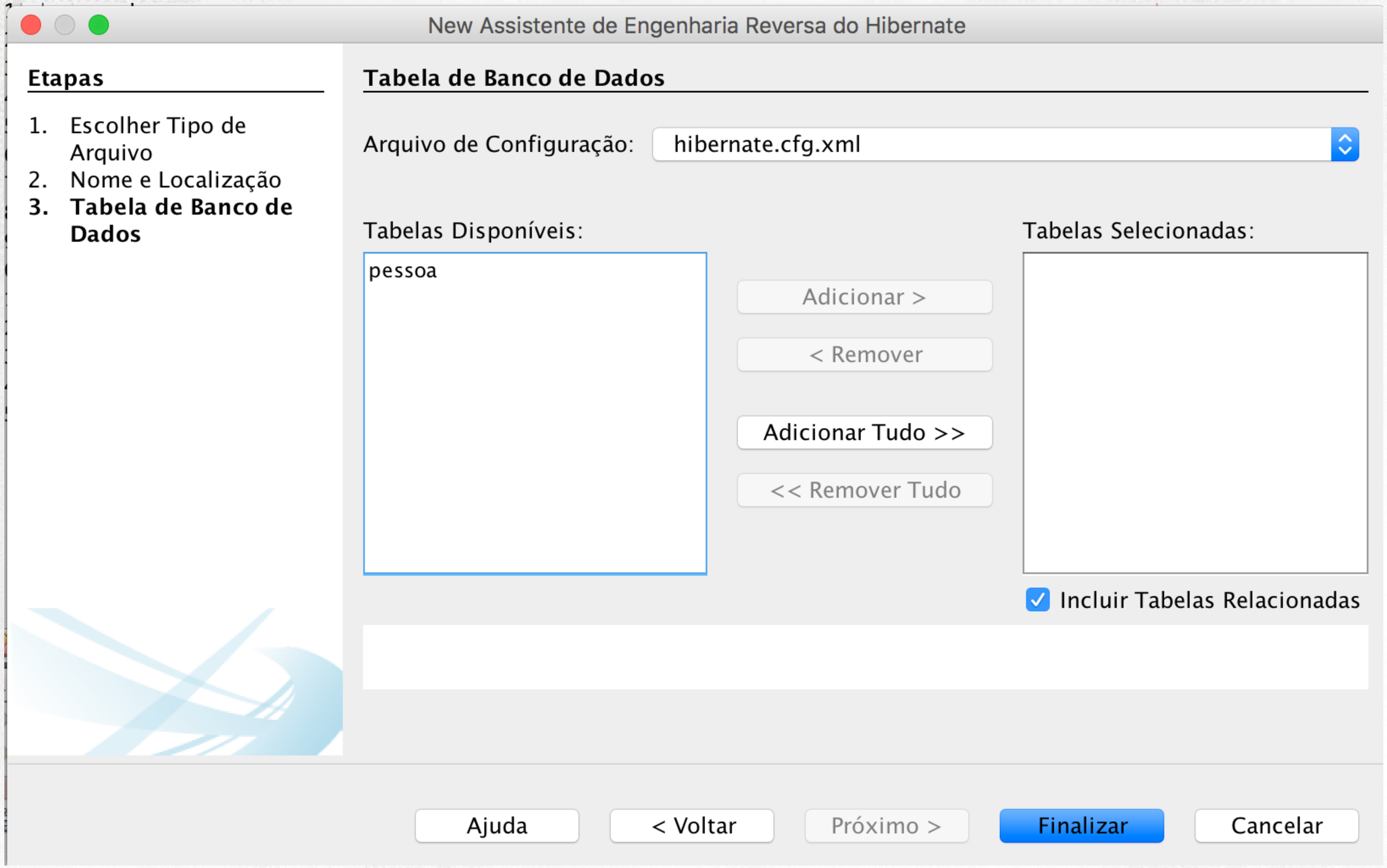
HIBERNATE

EXERCÍCIOS



HIBERNATE

EXERCÍCIOS



HIBERNATE

EXERCÍCIOS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse
Engineering DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-reverse-
engineering-3.0.dtd">
<hibernate-reverse-engineering>
  <schema-selection match-catalog="unicarioca"/>
  <table-filter match-name="pessoa"/>
</hibernate-reverse-engineering>
```

HIBERNATE

EXERCÍCIOS

1. **ASSISTENTE DE CONFIGURAÇÃO DO HIBERNATE**
   Informações de conexões com SGBD

2. **ASSISTENTE DE ENGENHARIA REVERSA DO HIBERNATE**
   Informações de tabelas do BD para engenharia reversa

3. **ARQUIVOS DE MAPEAMENTO DO HIBERNATE E POJOS DE BANCO DE DADOS**
   Criação das classes POJO através de engenharia reversa

4. **ASSISTENTE DE MAPEAMENTO DO HIBERNATE**
   Informações de mapeamento

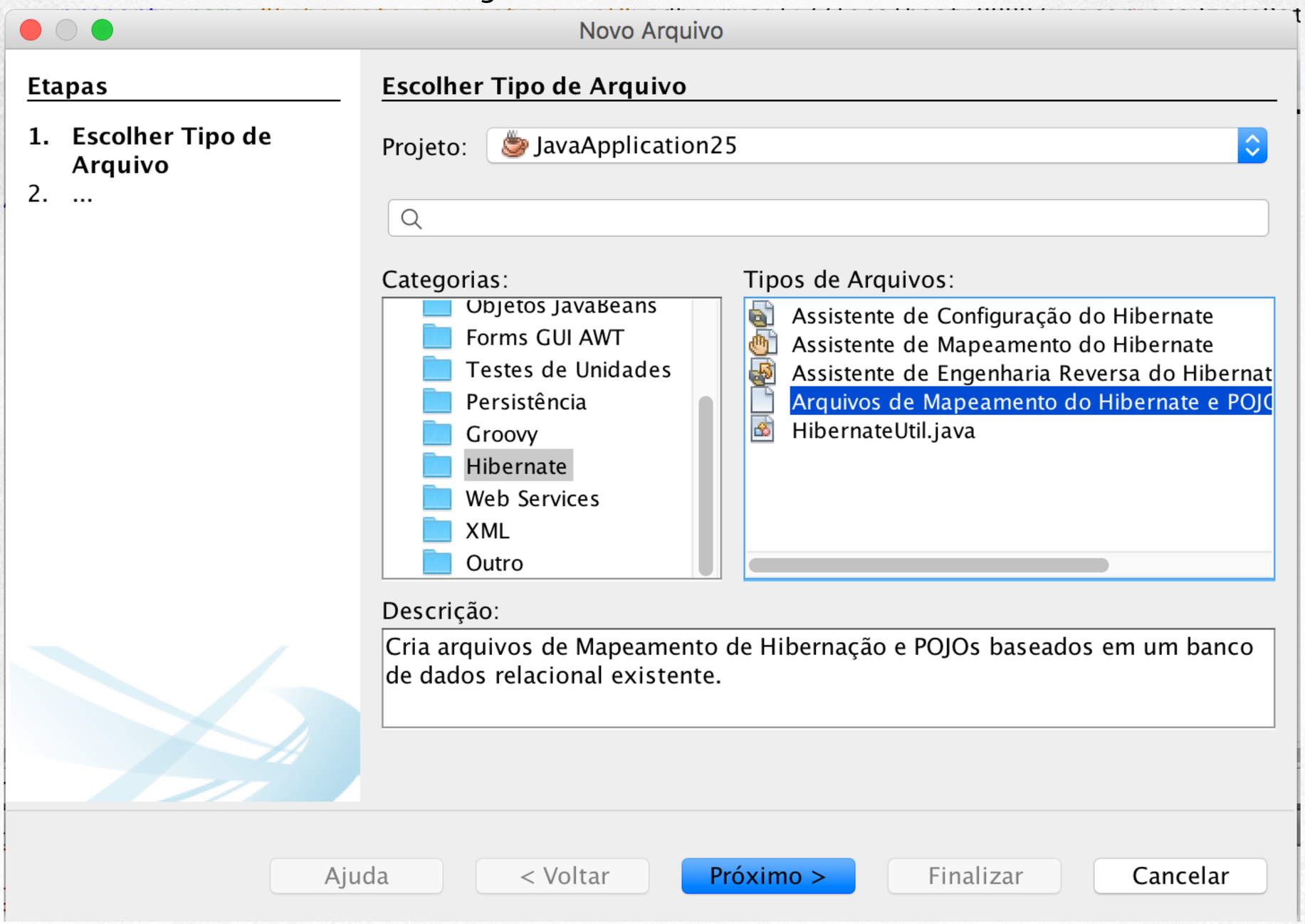
5. **HIBERNATEUTIL.JAVA**
   Arquivo para execução do hibernate

HIBERNATE

EXERCÍCIOS

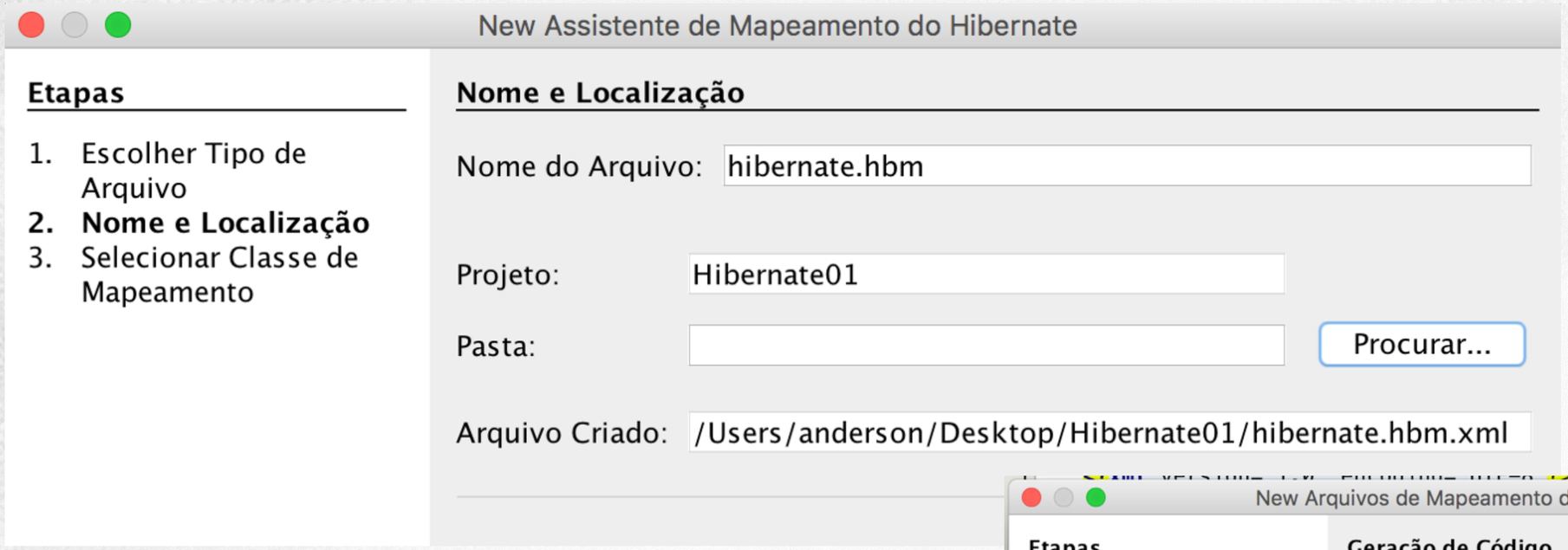
3 ARQUIVOS DE MAPEAMENTO DO HIBERNATE E POJOS DE BANCO DE DADOS

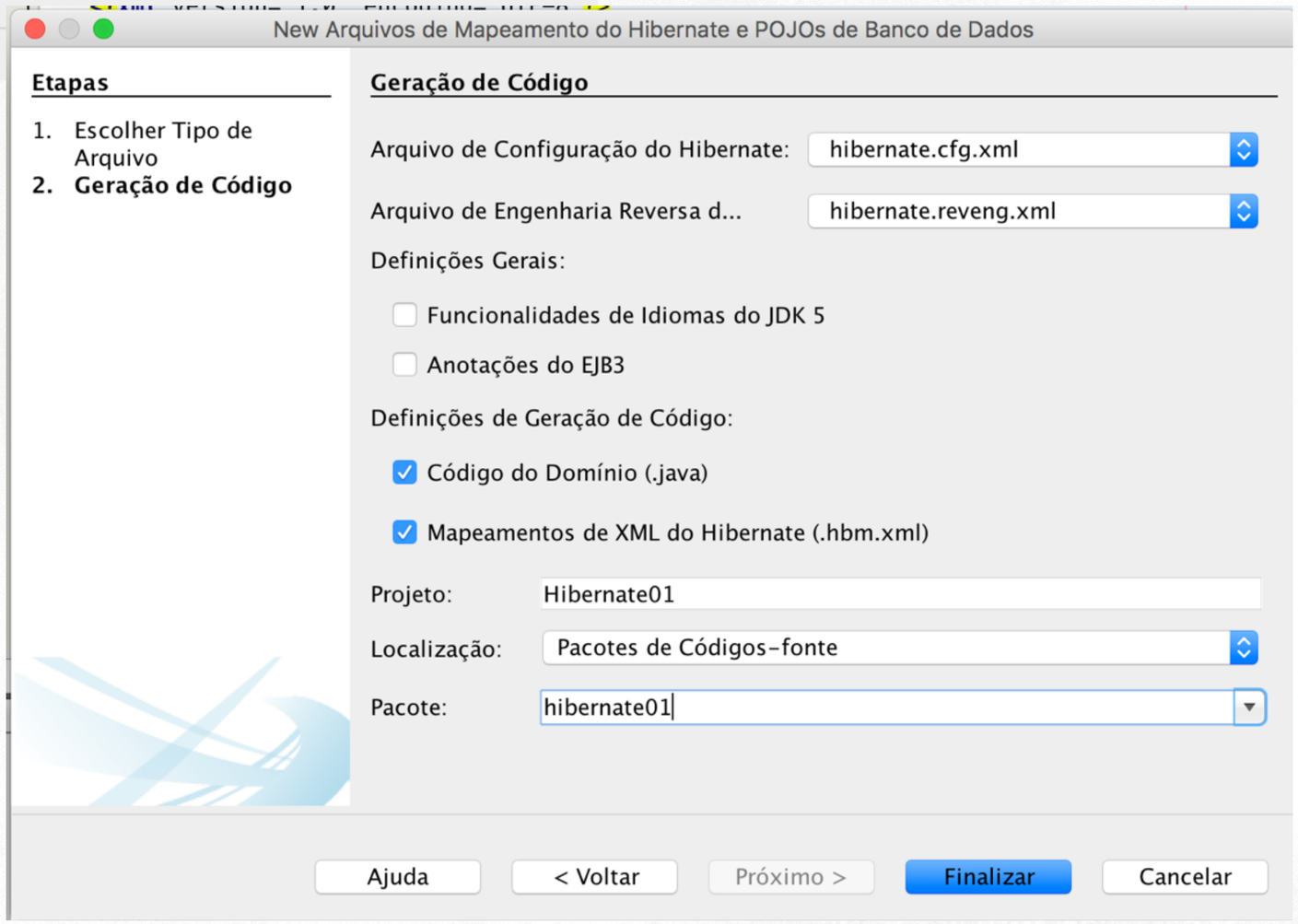
Criação das classes POJO através de engenharia reversa



HIBERNATE

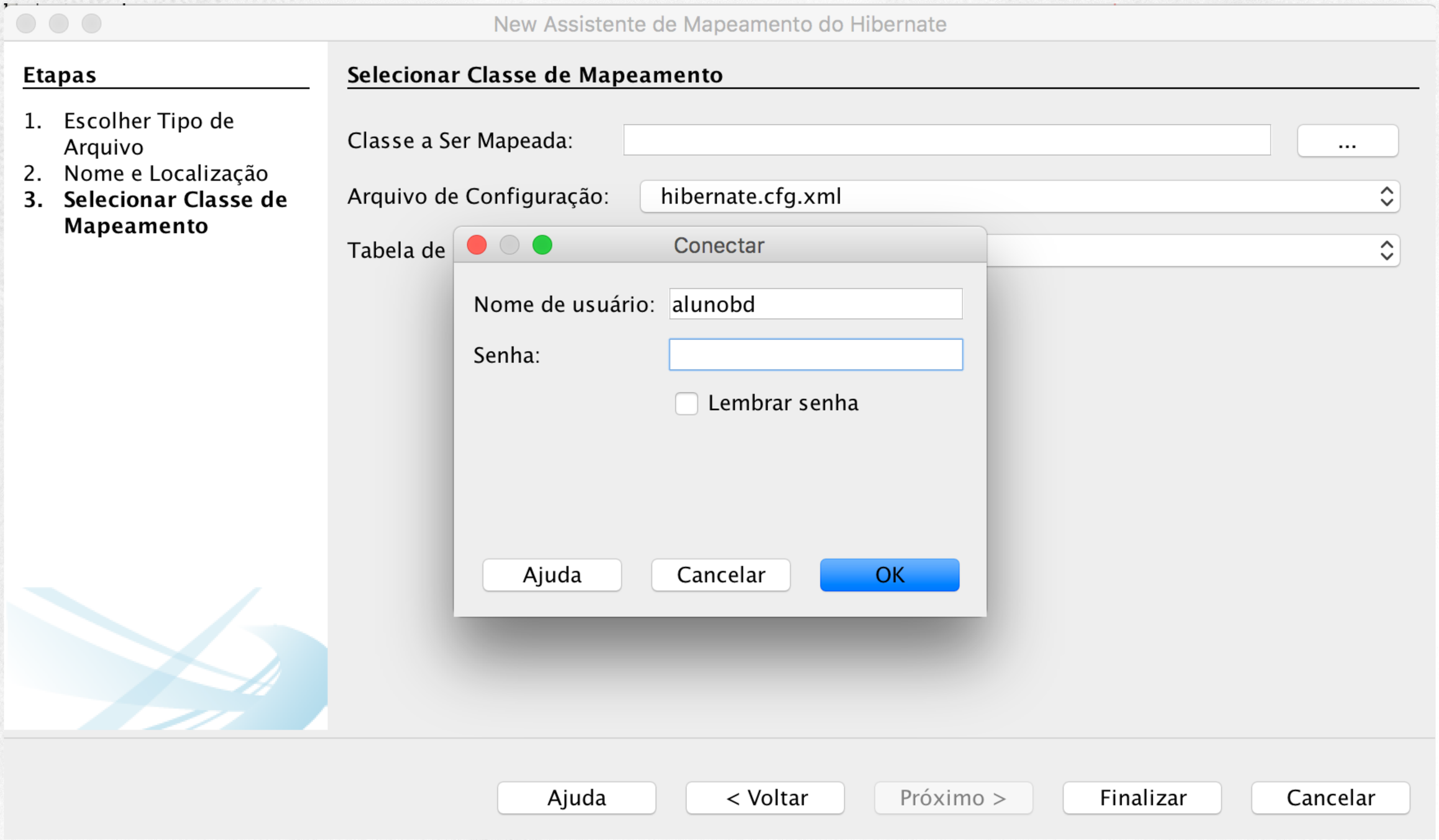
EXERCÍCIOS





HIBERNATE

EXERCÍCIOS



HIBERNATE

EXERCÍCIOS

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<!-- Generated 02/05/2016 15:39:58 by Hibernate Tools 4.3.1 -->
<hibernate-mapping>
    <class name="hibernate01.Pessoa" table="pessoa" catalog="unicarioca" optimistic-
lock="version">
        <id name="indice" type="java.lang.Integer">
            <column name="indice" />
            <generator class="identity" />
        </id>
        <property name="nome" type="string">
            <column name="nome" length="40" />
        </property>
        <property name="email" type="string">
            <column name="email" length="80" />
        </property>
        <property name="cpf" type="string">
            <column name="cpf" length="14" />
        </property>
        <property name="telefone" type="string">
            <column name="telefone" length="30" />
        </property>
    </class>
</hibernate-mapping>
```

HIBERNATE

EXERCÍCIOS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" “http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">


<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</
property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:8889/unicarioca?
zeroDateTimeBehavior=convertToNull</property>
    <property name="hibernate.connection.username">alunobd</property>
    <property name="hibernate.connection.password">unicarioca</property>
    <property
name="hibernate.current_session_context_class">org.hibernate.context.internal.ThreadLo
calSessionContext</property>
    <mapping resource="hibernate01/Pessoa.hbm.xml"/>
  </session-factory>



</hibernate-configuration>
```

HIBERNATE

EXERCÍCIOS

1. **ASSISTENTE DE CONFIGURAÇÃO DO HIBERNATE**
   Informações de conexões com SGBD

2. **ASSISTENTE DE ENGENHARIA REVERSA DO HIBERNATE**
   Informações de tabelas do BD para engenharia reversa

3. **ARQUIVOS DE MAPEAMENTO DO HIBERNATE E POJOS DE BANCO DE DADOS**
   Criação das classes POJO através de engenharia reversa

4. **ASSISTENTE DE MAPEAMENTO DO HIBERNATE**
   Informações de mapeamento

5. **HIBERNATEUTIL.JAVA**
   Arquivo para execução do hibernate

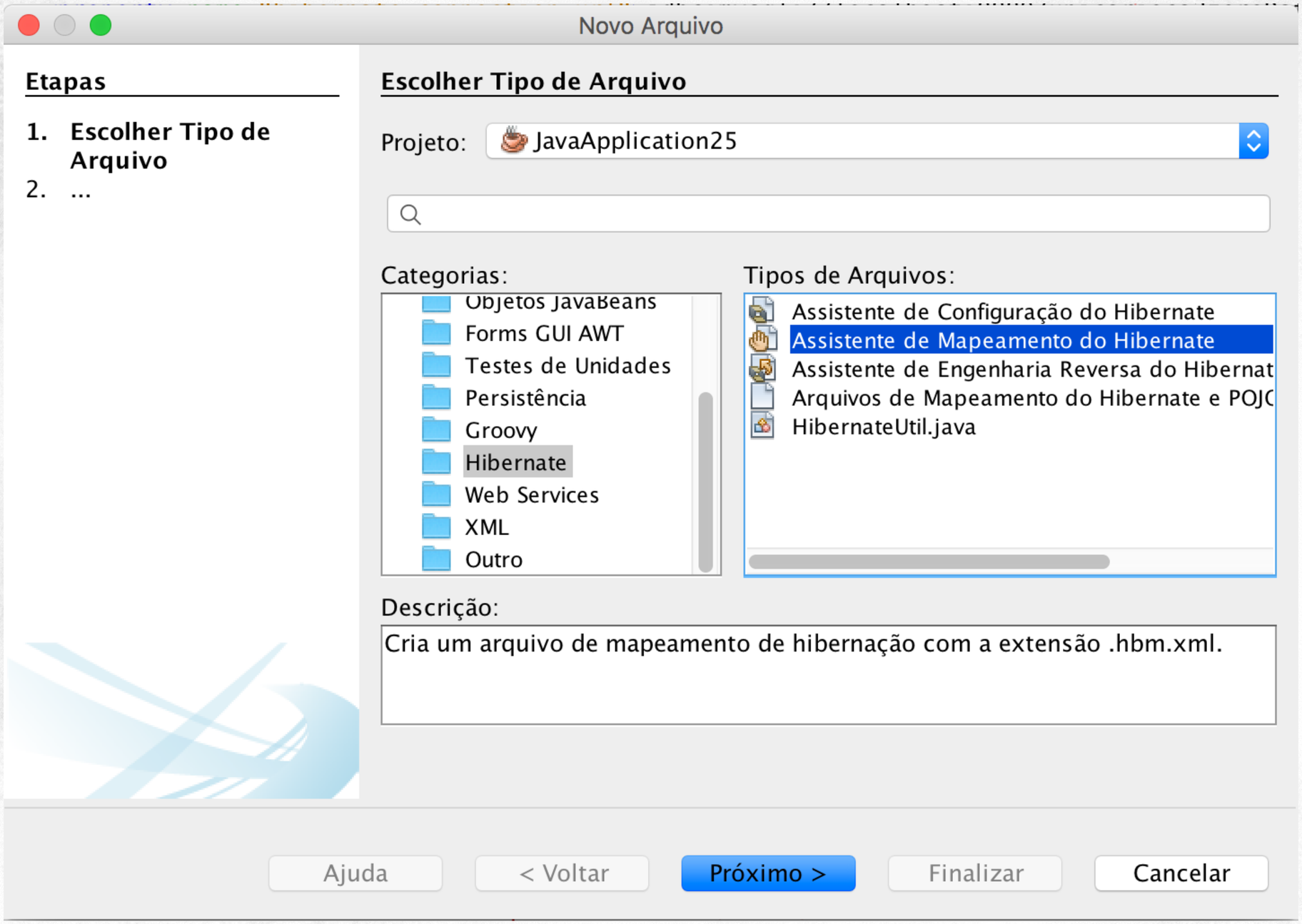
HIBERNATE

EXERCÍCIOS

4

ASSISTENTE DE MAPEAMENTO DO HIBERNATE

Informações de mapeamento



HIBERNATE

EXERCÍCIOS

1. **ASSISTENTE DE CONFIGURAÇÃO DO HIBERNATE**
   Informações de conexões com SGBD

2. **ASSISTENTE DE ENGENHARIA REVERSA DO HIBERNATE**
   Informações de tabelas do BD para engenharia reversa

3. **ARQUIVOS DE MAPEAMENTO DO HIBERNATE E POJOS DE BANCO DE DADOS**
   Criação das classes POJO através de engenharia reversa

4. **ASSISTENTE DE MAPEAMENTO DO HIBERNATE**
   Informações de mapeamento

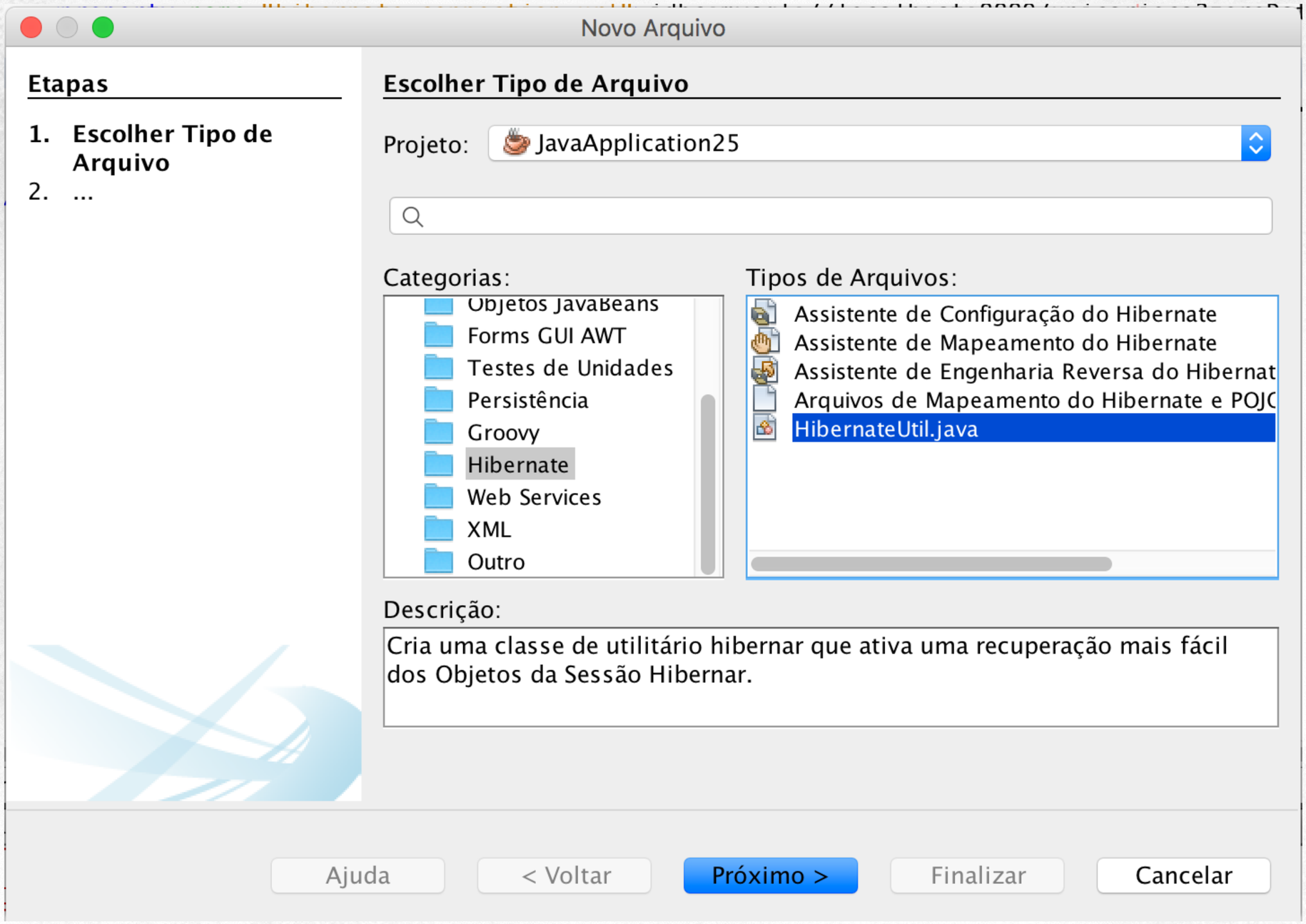
5. **HIBERNATEUTIL.JAVA**
   Arquivo para execução do hibernate

HIBERNATE

EXERCÍCIOS

5 HIBERNATEUTIL.JAVA

Arquivo para execução do hibernate



HIBERNATE

EXERCÍCIOS

```
public class NovoHibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            Configuration configuration = new Configuration();
            configuration.configure();
            ServiceRegistry serviceRegistry = new
StandardServiceRegistryBuilder().applySettings(configuration.getProperties()).build();

             sessionFactory = new Configuration().configure().buildSessionFactory(serviceRegistry);
        } catch (Throwable ex) {
            System.err.println(ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

HIBERNATE

EXERCÍCIOS

```
public class PessoaHelper {
    Session session = null;

    public PessoaHelper() {
        this.session = NovoHibernateUtil.getSessionFactory().getCurrentSession();
    }
```

[CONTINUA]

HIBERNATE

EXERCÍCIOS

[CONTINUANDO...]

```
    public List getList(int startID, int endID) {
    List<Pessoa> pessoaList = null;
    try {
        org.hibernate.Transaction tx = session.beginTransaction();
        Query q = session.createQuery ("from Pessoa as pessoa where pessoa.indice
between '"+startID+"' and '"+endID+"'");
        pessoaList = (List<Pessoa>) q.list();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return pessoaList;
}
}
```

HIBERNATE

EXERCÍCIOS

```
public static void main(String[] args) {
        PessoaHelper lista = new PessoaHelper();
        ArrayList pessoas = (ArrayList) lista.getList(0, 1000);
        Iterator proximo = pessoas.iterator();
        int i=0;
        while(proximo.hasNext()){
            Pessoa p = (Pessoa) pessoas.get(i);
            System.out.println(p.getNome());
            proximo.next();
        }
    }
```

HIBERNATE

AULA PASSADA

```
public List getList(int startID, int endID) {
List<Pessoa> pessoaList = null;
try {
    org.hibernate.Transaction tx =
session.beginTransaction();
        Query q = session.createQuery ("from Pessoa as pessoa
where pessoa.indice between '"+startID+"' and '"+endID+"'");
        pessoaList = (List<Pessoa>) q.list();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return pessoaList;
   }
```

HIBERNATE

EXERCÍCIO

Exercício: Implemente Inserir, Remover e Atualizar.

HIBERNATE

ATIVIDADE PRÁTICA SUPERVISIONADA

Servidor

Cliente

HIBERNATE

ATIVIDADE PRÁTICA SUPERVISIONADA

1- Projeto Java Web

Servidor

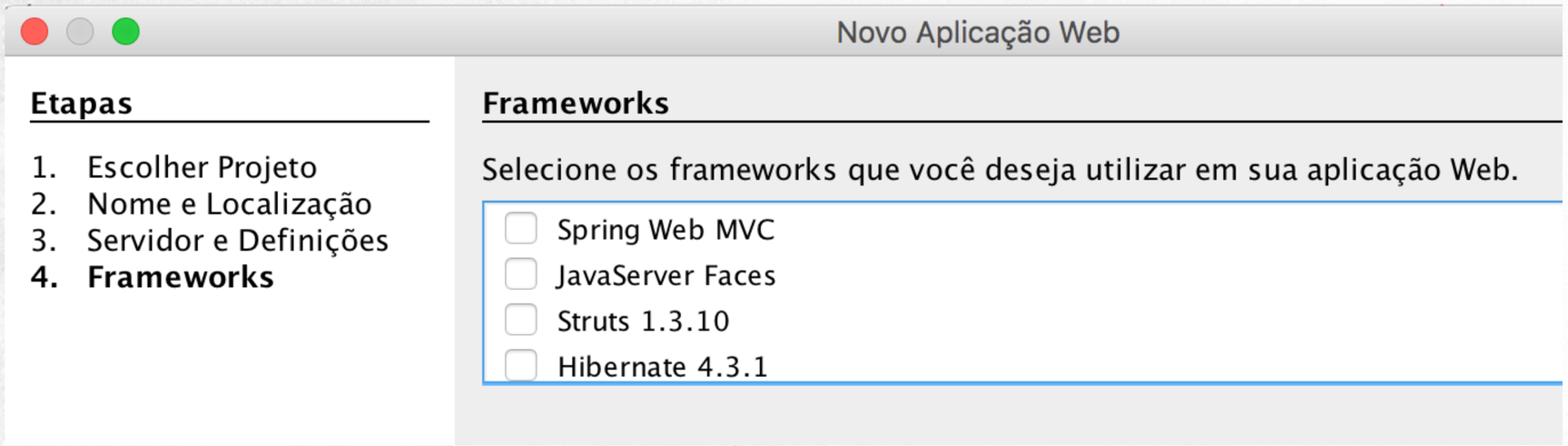
HIBERNATE

ATIVIDADE PRÁTICA SUPERVISIONADA

1- Projeto Java Web

a) Framework Hibernate

Servidor

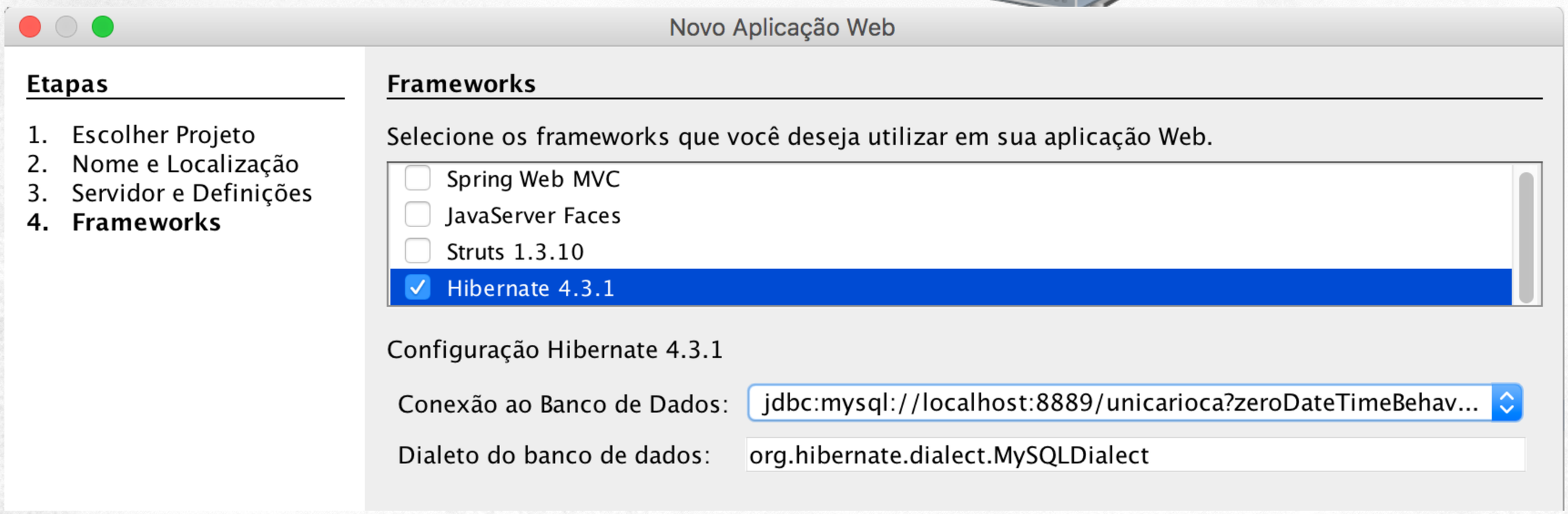


HIBERNATE

ATIVIDADE PRÁTICA SUPERVISIONADA

1- Projeto Java Web

a) Framework Hibernate

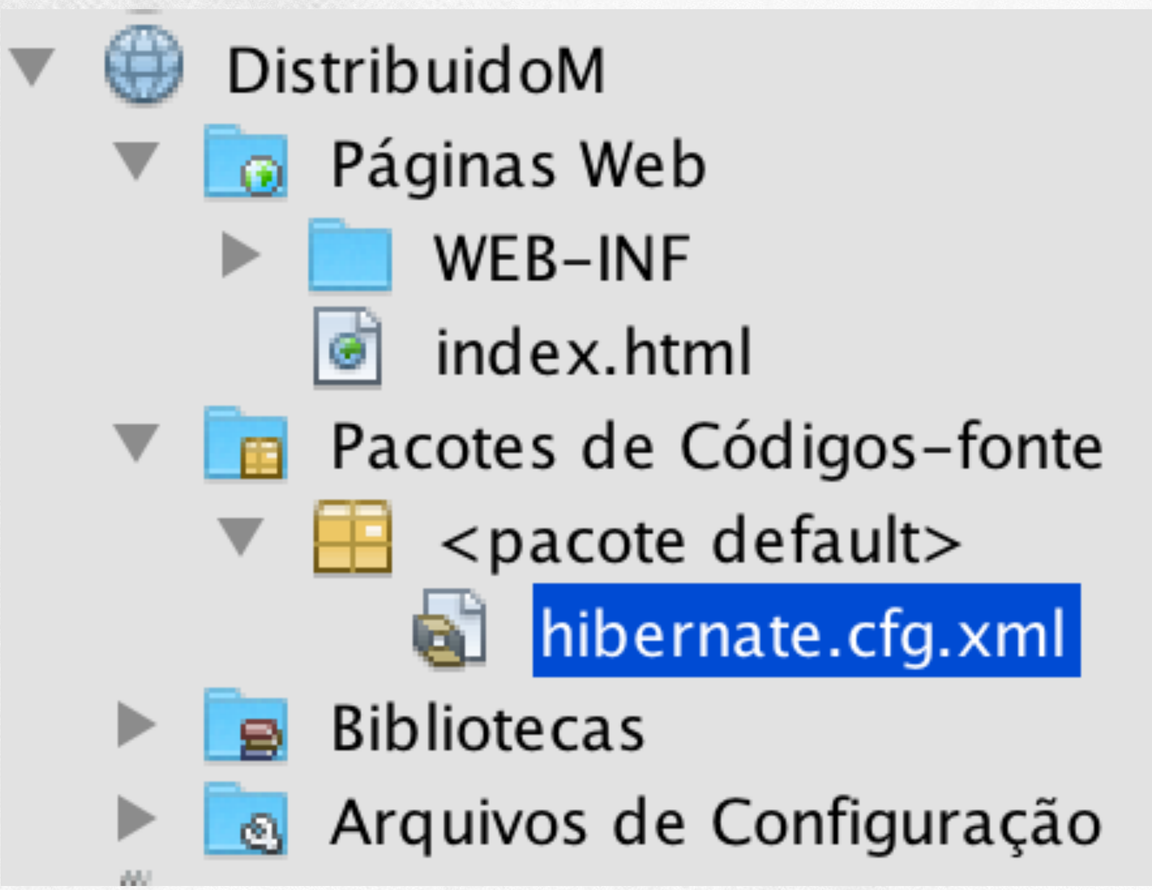


HIBERNATE

ATIVIDADE PRÁTICA SUPERVISIONADA

1- Projeto Java Web

a) Framework Hibernate



Servidor

HIBERNATE

ATIVIDADE PRÁTICA SUPERVISIONADA

1- Projeto Java Web

a) Framework Hibernate

b) Hibernate, DABD

Servidor

1 **ASSISTENTE DE CONFIGURAÇÃO DO HIBERNATE**
Informações de conexões com SGBD

2 **ASSISTENTE DE ENGENHARIA REVERSA DO HIBERNATE**
Informações de tabelas do BD para engenharia reversa

3 **ARQUIVOS DE MAPEAMENTO DO HIBERNATE E POJOS DE BANCO DE DADOS**
Criação das classes POJO através de engenharia reversa

4 **ASSISTENTE DE MAPEAMENTO DO HIBERNATE**
Informações de mapeamento

5 **HIBERNATEUTIL.JAVA**
Arquivo para execução do hibernate

HIBERNATE

ATIVIDADE PRÁTICA SUPERVISIONADA

1- Projeto Java Web

a) Framework Hibernate

b) Hibernate, DABD

Servidor

1 **ASSISTENTE DE CONFIGURAÇÃO DO HIBERNATE**
Informações de conexões com SGBD

2 **ASSISTENTE DE ENGENHARIA REVERSA DO HIBERNATE**
Informações de tabelas do BD para engenharia reversa

3 **ARQUIVOS DE MAPEAMENTO DO HIBERNATE E POJOS DE BANCO DE DADOS**
Criação das classes POJO através de engenharia reversa

4 **ASSISTENTE DE MAPEAMENTO DO HIBERNATE**
Informações de mapeamento

5 **HIBERNATEUTIL.JAVA**
Arquivo para execução do hibernate

HIBERNATE

ATIVIDADE PRÁTICA SUPERVISIONADA

1- Projeto Java Web

a) Framework Hibernate

b) Hibernate, DABD

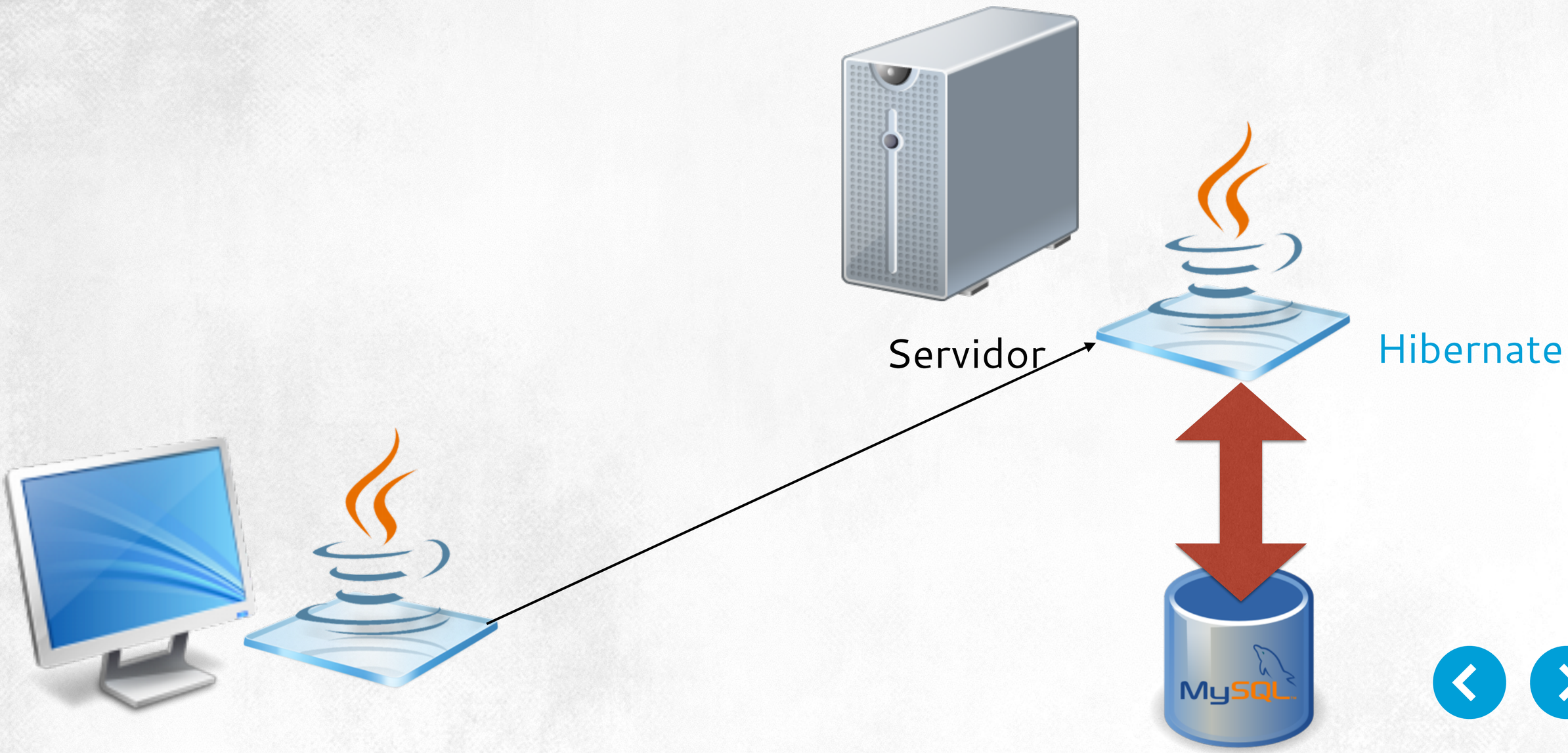
c) WebService, DAD

Servidor

```
@WebService(serviceName = "Pessoas")
public class Pessoas {

    public String listarPessoas() {
        String nomes = "";
        PessoaHelper lista = new PessoaHelper();
        ArrayList pessoas = (ArrayList) lista.getList(0, 1000);
        Iterator proximo = pessoas.iterator();
        int i = 0;
        while (proximo.hasNext()) {
            Pessoa p = (Pessoa) pessoas.get(i);
            nomes += "\n" + p.getNome();
            proximo.next();
        }
        return nomes;
    }

}
```

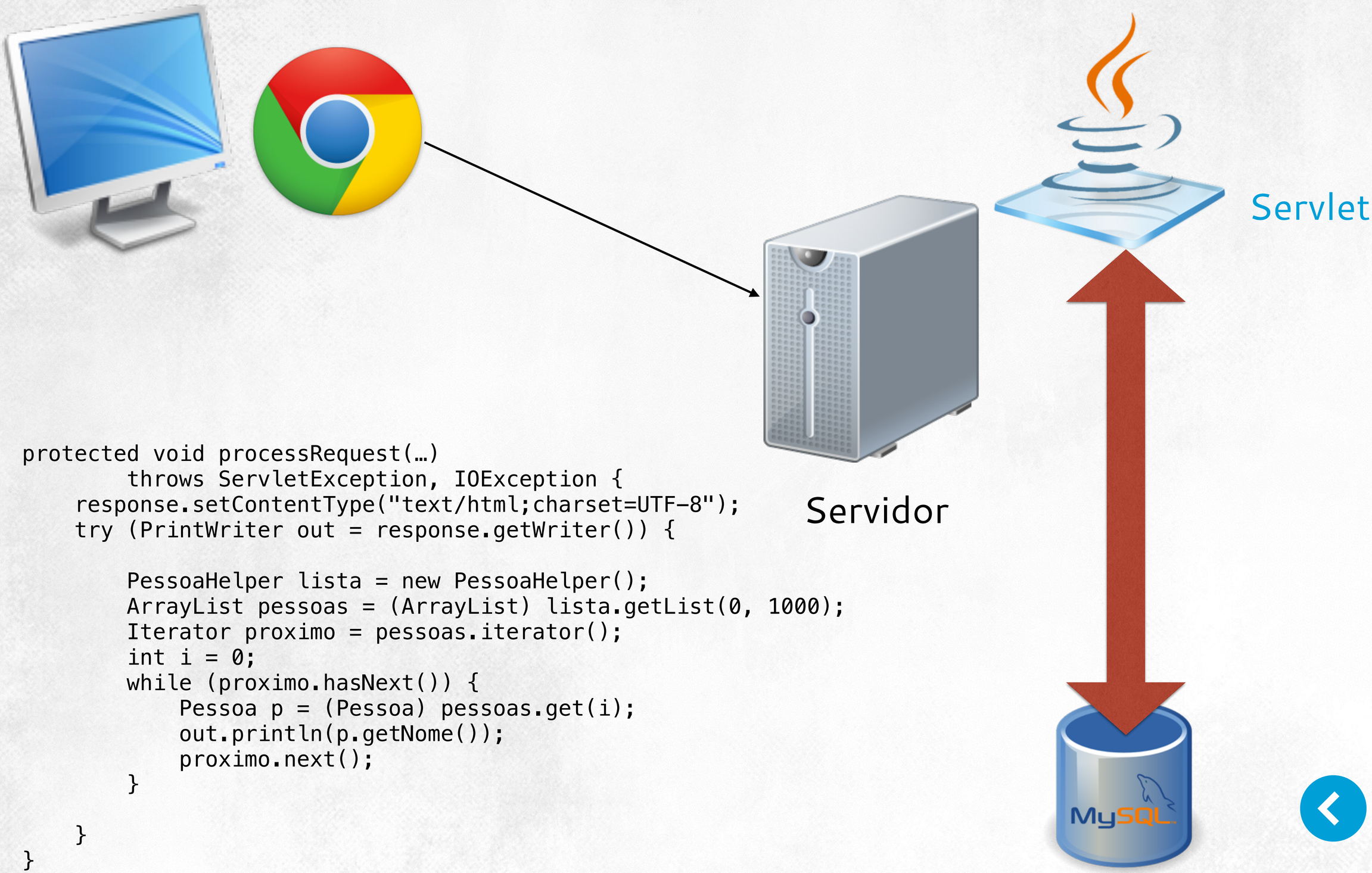
HIBERNATE

ATIVIDADE PRÁTICA SUPERVISIONADA



HIBERNATE

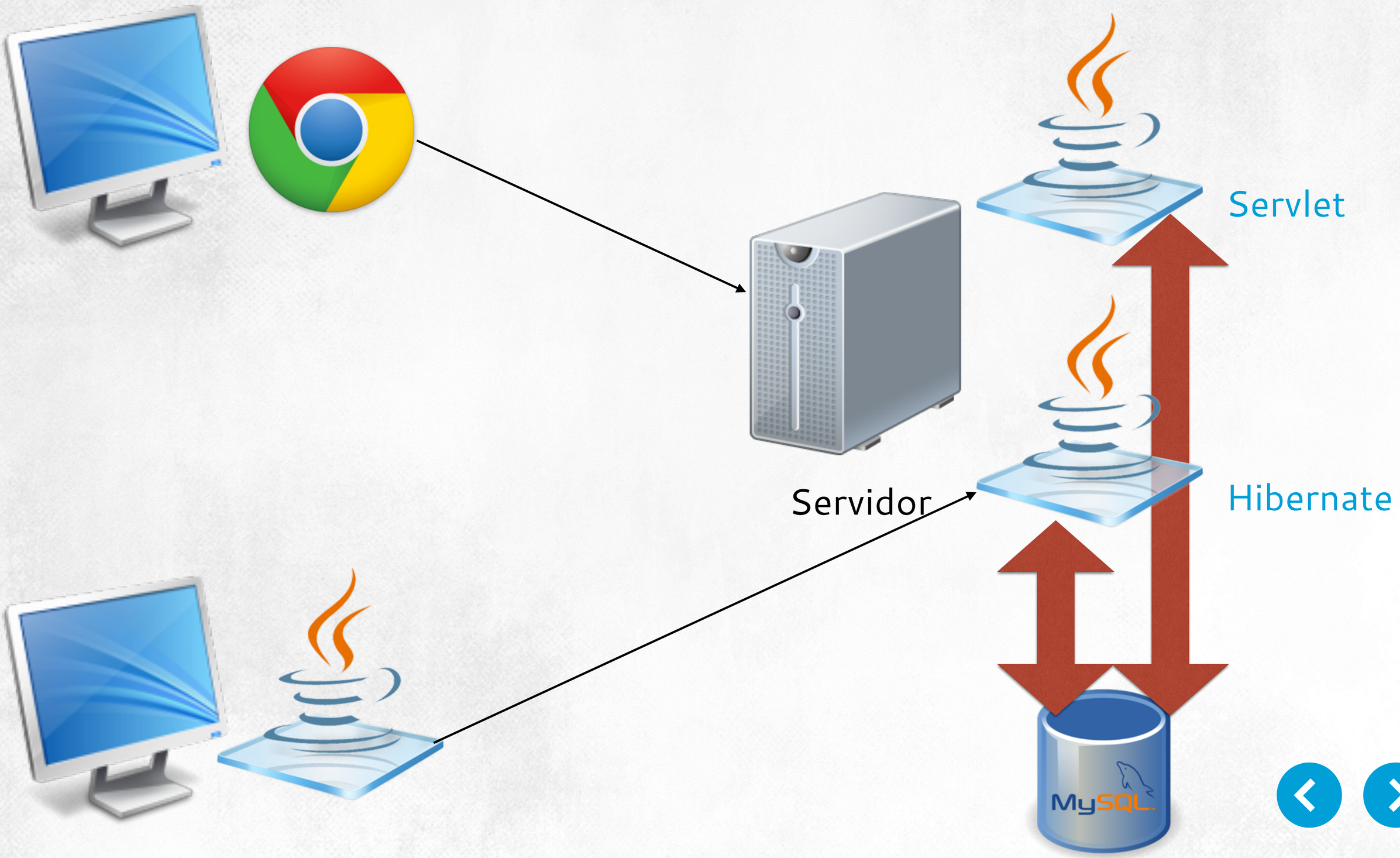
ATIVIDADE PRÁTICA SUPERVISIONADA




```
protected void processRequest(…)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

        PessoaHelper lista = new PessoaHelper();
        ArrayList pessoas = (ArrayList) lista.getList(0, 1000);
        Iterator proximo = pessoas.iterator();
        int i = 0;
        while (proximo.hasNext()) {
            Pessoa p = (Pessoa) pessoas.get(i);
            out.println(p.getNome());
            proximo.next();
        }

    }
}
```

HIBERNATE

ATIVIDADE PRÁTICA SUPERVISIONADA



OBRIGADO

BOA-SORTE!