JSP – Java Sever Page

Curso Técnico em Informática

JDBC

Java Database Connectivity

Acesso a Banco de dados JAVA

Prof. William Geraldo Sallum

JSP - JDBC: Motivação

- Surgimento de sistemas cliente-servidor
  - diferentes fabricantes
  - diferentes linguagens
  - diferentes plataformas/ambientes
  - integração com sistemas legados
- Necessidade de comunicar-se
  - compartilhamento de dados
  - compartilhamento de serviços
  - centralização de regras de negócio
  - atendimento a requisitos não funcionais
    - desempenho
    - escalabilidade
    - concorrência
    - segurança ...

JSP - JDBC: Histórico

- SGBD: Nos primórdios
- conectividade através de "biblioteca" prioritária
  - uma "biblioteca" distinta para cada SGBD
    - escrita numa linguagem também específica
  - dependência forte da linguagem na qual o SGBD foi escrito
    - ou dependência forte da linguagem na qual a "biblioteca" foi escrita
- Problema imediato
  - e se for necessidade de substituir SGBD?
- Solução imediata
  - protocolo comum de acesso ao repositório de dados

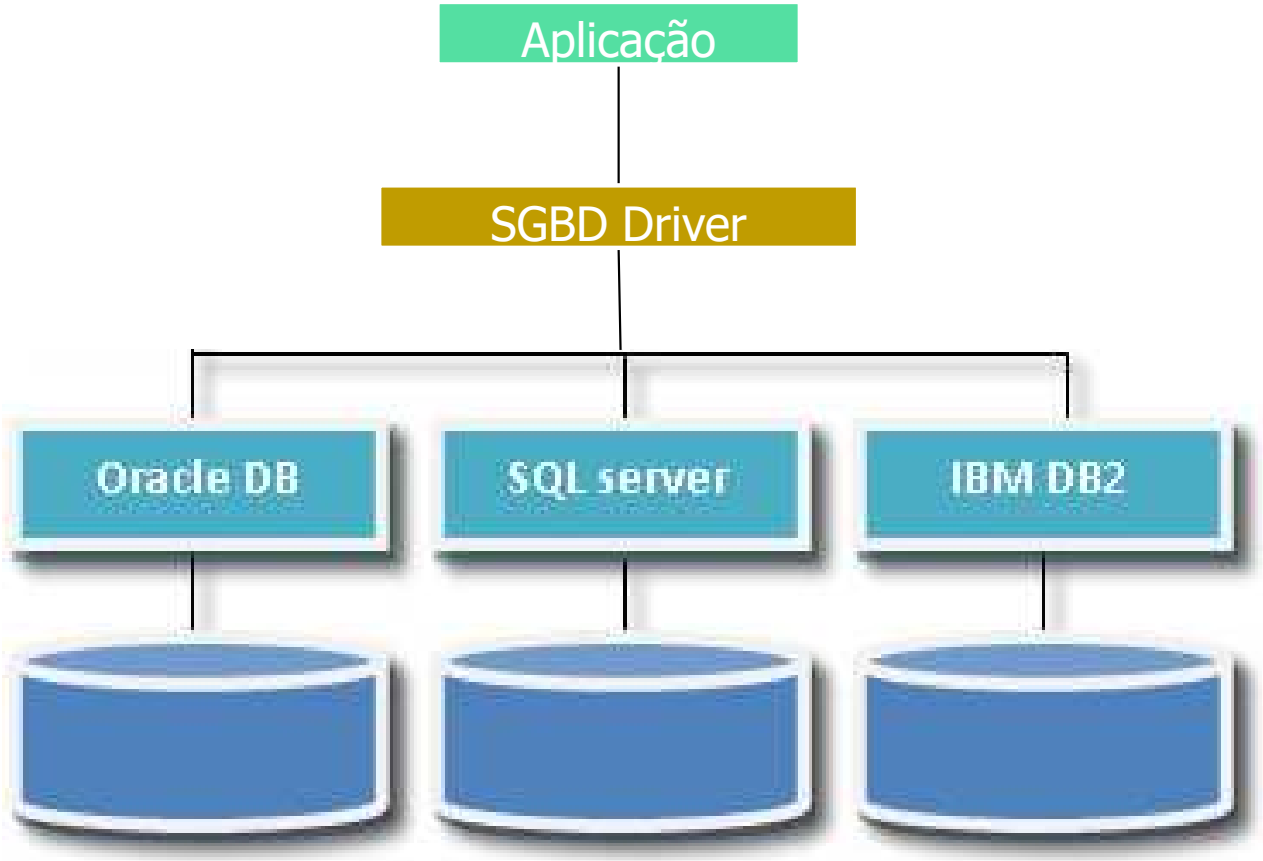
JSP - JDBC: Protocolo de Comunicação

- Vários padrões de comunicação e acesso a dados
  - RPC (*Remote Procedure Call*)
  - RDA (*Remote Database Access*) - ISO
  - EDA (*Enterprise Data Access*) – Indústria
  - CORBA (*Common Object Request Broker Architecture*)
  - DCOM (*Distributed Component Object Model*)
  - Java RMI (*Java Remote Method Invocation*)
  - .NET Remoting

JSP - JDBC: Uso de API Padronizada

- API = *Application Programming Interface*
- Especificação padronizada para acesso a SGBD
  - Independente
    - de fabricante
    - de linguagem de programação
    - de ambiente/Plataforma
    - de infraestrutura física
  - Baseado em um "*device driver*" (*middleware*) local
    - que concretiza/implementa a especificação padronizada

JSP - JDBC: Arquitetura



JSP - JDBC: Especificações mais difundidas

- ODBC
  - *Open Database Connectivity*
- JDBC
  - *Java Database Connectivity*
- ADO .Net
  - *Activex Data Objects*

JSP - JDBC

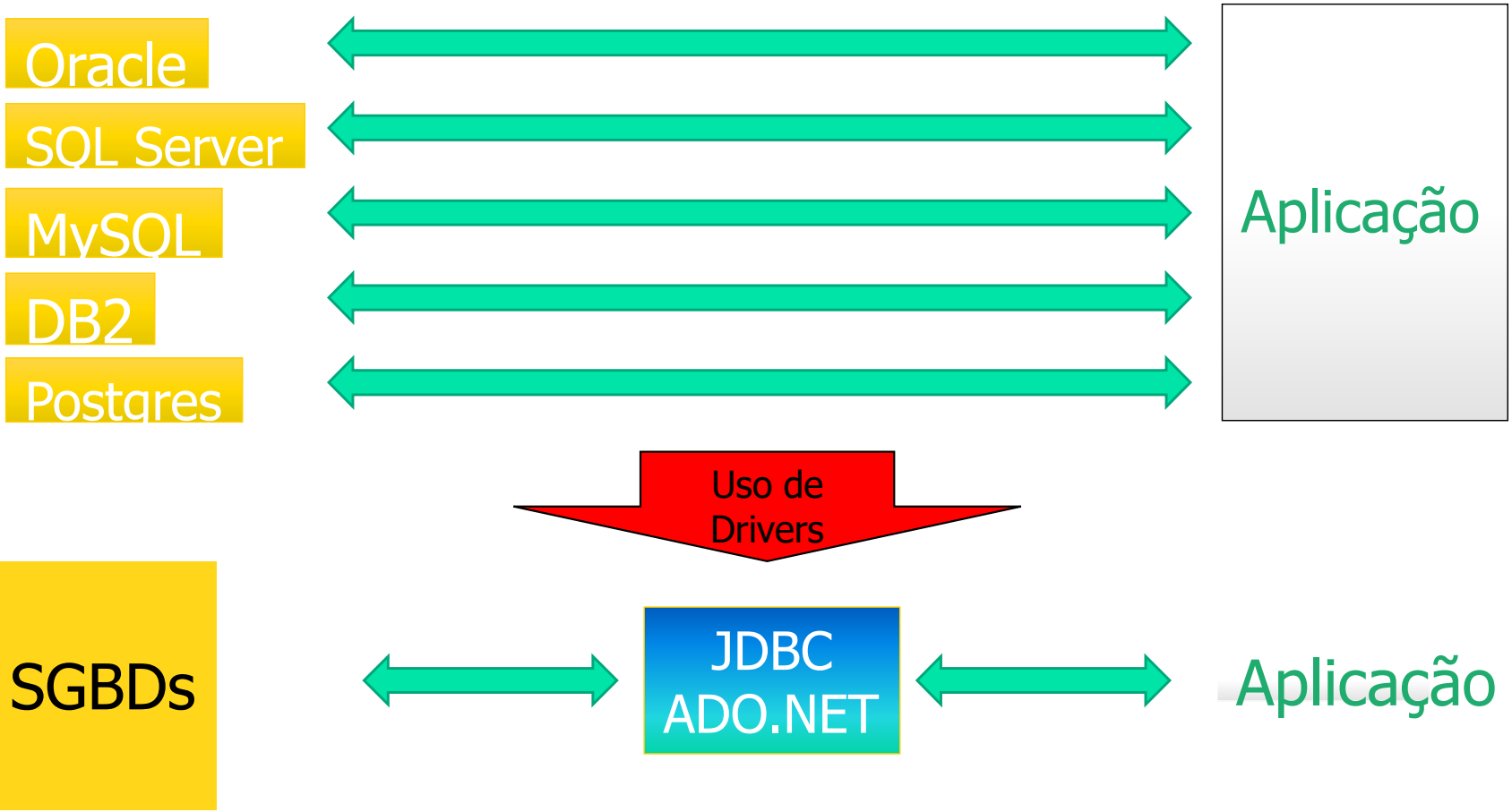
- O processo de armazenamento de dados é também chamado de **persistência**.
- A biblioteca de persistência em banco de dados relacionais do Java é chamada JDBC (*Java DataBase Connectivity*).
- Existem diversas ferramentas do tipo **ORM** (*Object Relational Mapping*) que facilitam bastante o uso do JDBC.

JSP - JDBC: APIs para conexão

- JDBC
  - Java Database Connectivity
  - Componente da plataforma Java
    - para conexão com bancos de dados transacionais
- ADO.NET
  - Componente da plataforma Microsoft .Net
    - para conexão com bancos de dados transacionais

JSP - JDBC: Objetivos

- Simplificar o acesso a diferentes fornecedores de SGDBs



JSP - JDBC: API para Conexão

- Fornece uma abstração para manipulação de base de dados relacionais
  - Baseado nos padrões ANSII SQL
    - Padrão implementado por quase todos os SGDBs
  - Permite que qualquer fornecedor implemente um Driver para seu(s) SGDB(s)

JSP - JDBC: API para Conexão

- Ou seja:
  - As aplicações passam a interagir com a API JDBC/ADO.NET para acesso ao banco de dados
    - Independentemente do SGDB utilizado
  - Os fornecedores de SGDBs: Oracle, Microsoft, MySQL, PostgreSQL, etc.
    - Implementam a especificação para conectar os serviços dos seus SGDBs via JDBC/ADO.NET

JSP - JDBC

- A conexão a um banco de dados é feita de maneira elegante com Java.
- Existe um conjunto de interfaces para conexão e acesso a BD em Java que fica dentro do pacote java.sql e nos referiremos a ele como **JDBC**.
- Neste pacote existe a interface Connection, que define métodos para executar uma query (como um insert e um select).

JSP - JDBC: Principais componentes

- **`DriverManager`**
  - Responsável por gerenciar todos os Drivers disponíveis para a aplicação
- **`Driver`**
  - Implementação da especificação JDBC
    - que seja aderente a um determinado SGBD
- **`Connection`**
  - Representa uma conexão física com o banco de dados
- **`Statement`**
  - Um conjunto de comandos SQL (*select, insert, update,* etc.)
- **`ResultSet`**
  - O resultado de uma consulta SQL via *statement*

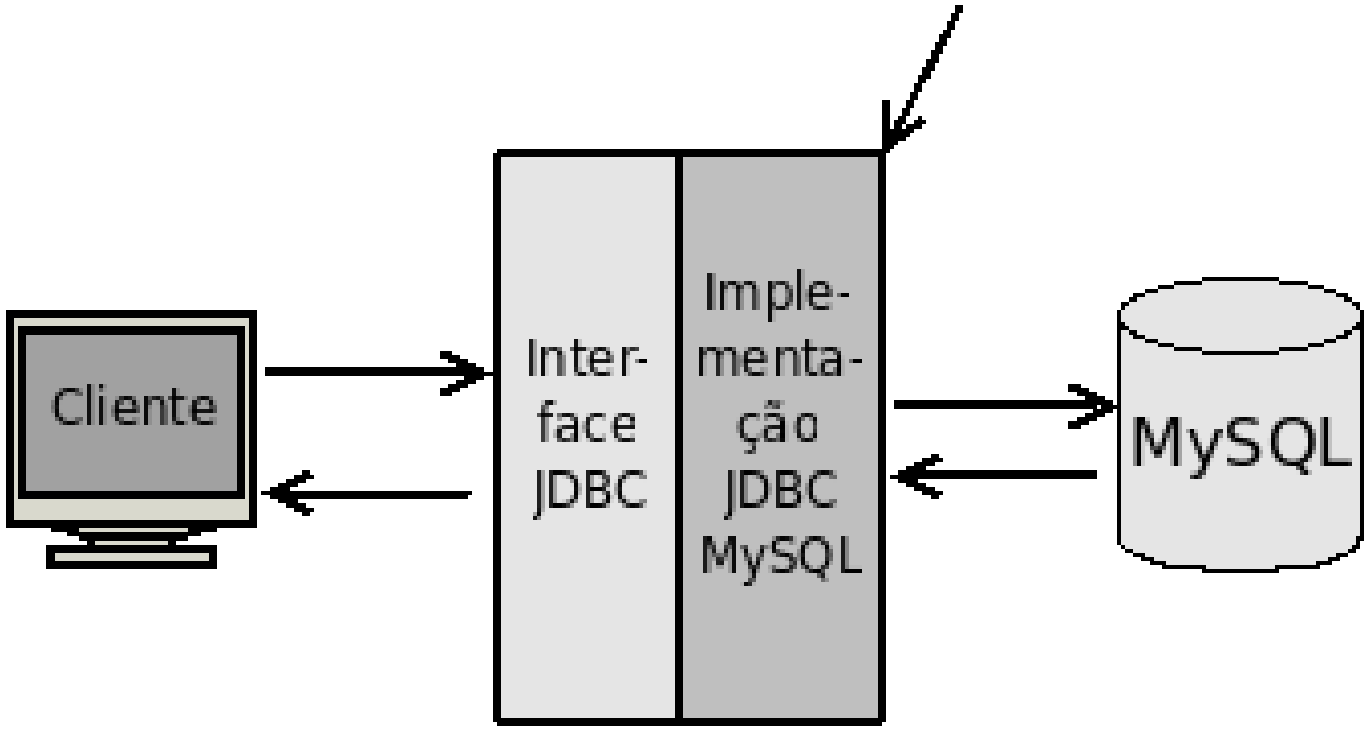
JSP - JDBC

Então, mãos a massa:

- Focaremos nos conceitos e no uso do JDBC.
- O **MySQL** é o banco de dados que usaremos durante o curso.

JSP - JDBC

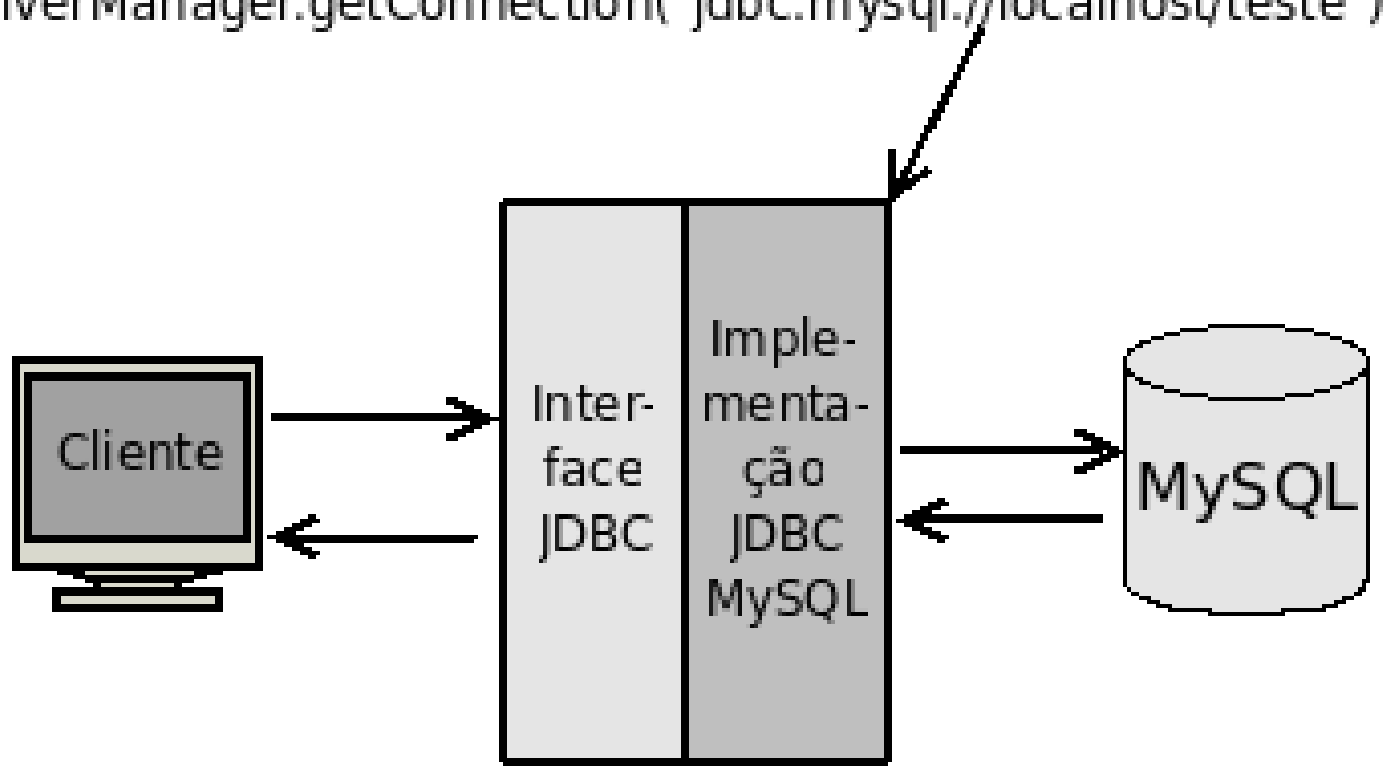
Qual implementação fazer ?



JSP - JDBC

- Para trabalhar com o MySQL, precisamos de classes concretas que implementem essas interfaces do pacote java.sql.

```
DriverManager.getConnection("jdbc:mysql://localhost/teste");
```



JSP - JDBC: URL de conexão

- String que fornece os dados necessários para criar uma conexão com o banco de dados:

  <jdbc:<subprotocolo>:<fonte>>

- Cada driver tem seu próprio subprotocolo
- Cada subprotocolo tem sua própria sintaxe e parâmetros
- Exemplos:
  - <jdbc:odbc:DataSource>
    - e.g. jdbc:odbc:Northwind
  - jdbc:mysql://host[:port]/database
    - e.g. jdbc:mysql://foo.nowhere.com:4333/accounting ou jdbc:mysql://localhost/fj21, ...

JSP - JDBC

Relação de algumas bases de dados

Banco	Driver	String de Conexão
Microsoft ODBC	sun.jdbc.odbc.JdbcOdbcDriver	jdbc:odbc:<<nome da base>>
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://<<ipDoBanco>>/<<baseDeDados>>
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@<<ipDoBanco>>:1521:<<nomeDaBase>>

JSP - JDBC

- Para abrir uma conexão com um banco de dados, precisamos utilizar sempre um driver e no caso a classe *DriverManager* para essa conexão.
- O método *getConnection* da classe *DriverManager* indica qual BD conectar:

String de conexão JDBC

```
Connection conexao = DriverManager.getConnection("jdbc:mysql://localhost/fj21");
```

JSP - JDBC

- Seguindo o exemplo anterior, seria possível rodar o exemplo abaixo e receber uma conexão para um banco MySQL, caso ele esteja rodando na mesma máquina:

```
public class JDBCConexao {
    public Connection getConnection() {
    try{
        return DriverManager.getConnection(
                "jdbc:mysql://localhost/meuDB", "root", "123");
       } catch (SQLException e){
                thow new RunTimeException(e);
        }
    }
}
```

JSP - JDBC

- E para obter uma conexão com o BD, usaremos:

```
Connection com = new JDBCConexao.getConnection();
```

- *Nota: É possível ter que configurar:*
  - a timezone utilizando: ?useTimezone=true&serverTimezone=UTC
  - o protocolo de segurança: ?useSSL=false

JSP - JDBC e o JSTL

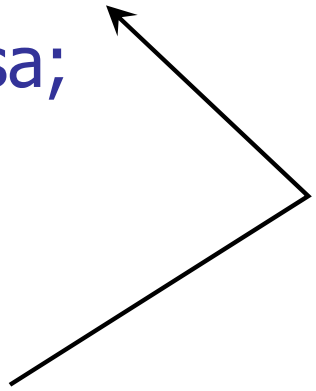
- Uma forma simplificada de conexão com BD através de JSTL (sql:setDataSource):

```
<sql:setDataSource  var = “conexao"
                    driver =  "com.mysql.jdbc.Driver"
                    url =  "jdbc:mysql://localhost/biblioteca"
                     user = "root"
                     password = ""/>
```

JSP - JDBC

- Consultando no BD (sql:query)

```
<sql:query dataSource = "${conexao}"   var = "result">
        SELECT * from Empresa;
</sql:query>
```



ponteiro da conexão criado em
<sql:setDataSource >

JSP - JDBC

- Para mostrar os resultados da consulta:

```
<table border = "1" width = "100%">
  <tr>
      <th>ID</th><th>NOME</th><th>SOBRENOME</th><th>IDADE</th>
  </tr>
      <c:forEach var = "row" items = "${result.rows}">
          <tr>
            <td>
              <c:out value = "${row.id}"/>  </td>
            <td>
              <c:out value = "${row.first}"/>  </td>
            <td>
              <c:out value = "${row.last}"/>   </td>
             <td>
              <c:out value = "${row.age}"/>   </td>
          </tr>
      </c:forEach>
</table>
```

JSP - JDBC

- Inserção no BD:

```
<sql:update dataSource = "${conexao}" var = "result">
    INSERT INTO Empresa VALUES (104, 30, 'Harumi', 'Sallum');
</sql:update>
```

JSP - JDBC

- Atualização no BD:

```
<c:set var = "empId" value = "104"/>
<c:set var = "sobreN" value = "Sallum"/>
<sql:update dataSource = "${conexao}" var = "count">
    UPDATE Empresa SET ID = ? WHERE last = ?
        <sql:param value = "${empId}" />
        <sql:param value = "${sobreN}" />
</sql:update>
```

- Utilizar sql:param junto com a cláusula **SET** e WHERE, identifica o registro cujo campo *last* é igual a Sallum e altera o campo Id tal como especificado em c:set.

JSP - JDBC

Deleta registros no BD (sql:update):

```
<c:set var = "empId" value = "104"/>
<sql:update dataSource = "${conexao}" var = "count">
    DELETE FROM Empresa WHERE Id = ?
        <sql:param value = "${empId}" />
</sql:update>
```

- Quando utilizado sql:param junto com a referência aberta de **?**, este indica que tal parâmetro especificado deve entrar no lugar dessa referência aberta.

JSP - Referências

- Slides renderizados de: http://docente.ifrn.edu.br/fellipealeixo/disciplinas/tads-2012/desenvolvimento-de-sistemas-web/material/antigo/jsp01.pdf. Último acesso em: 01/05/2017.

- Oracle. JDBC Basics. Disponível em: https://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html. Último acesso em: 06/05/2017.