Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

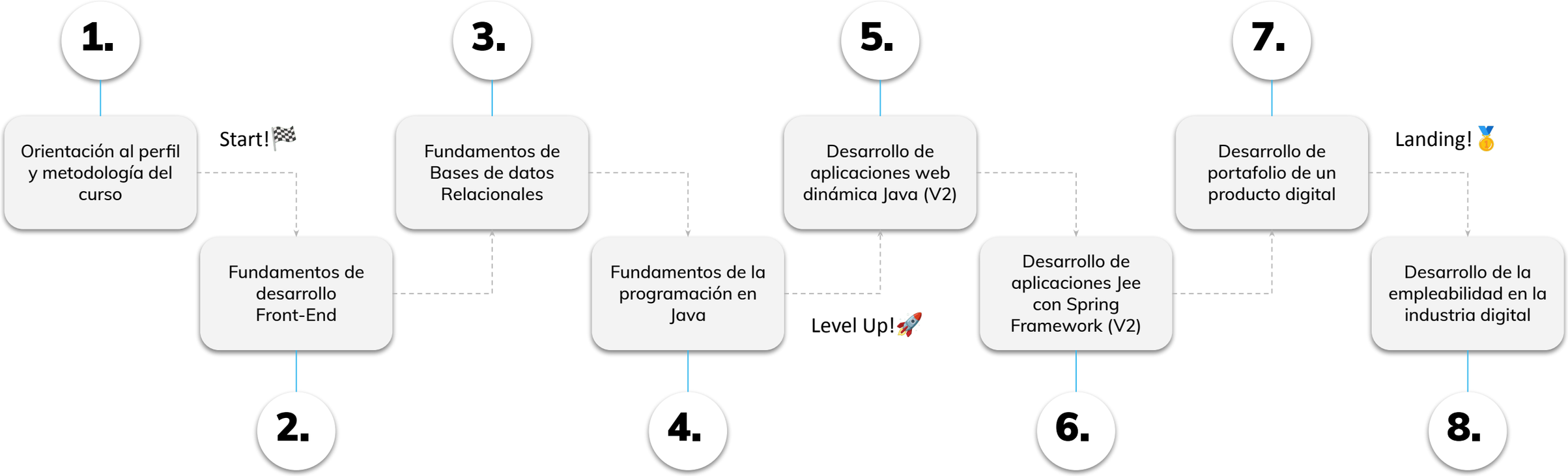
Capa de Acceso a Datos

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

English Always | alkemy

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



English Always | alkemy

REPASO CLASE ANTERIOR

En la clase anterior trabajamos 📚:

- ✓ *Patrón de diseño Singleton para conexiones*
- ✓ *Obtención de datos con Statement*

English Always | alkemy

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

5.4 Start! 🏁

Capa de acceso a datos

Obtención de datos (Objeto Resultset)
Modificación de datos.
La capa de acceso a datos (DAL)

Obtención y modificación de datos

Cambios y Resultados

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

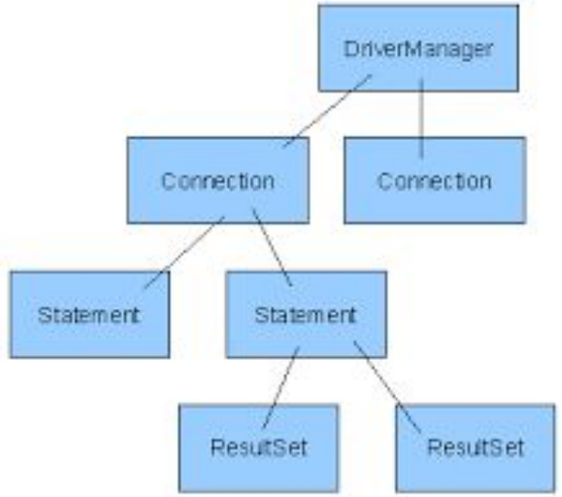
- *Obtener datos con Resultset*
- *Modificar datos con JDBC*
- *Conocer la capa de acceso a datos (DAL)*

Obtención de datos:
ResultSet

ResultSet

Para obtener datos almacenados en la BD se utiliza una consulta SQL (query). La consulta se puede ejecutar utilizando el objeto **Statement, con el método executeQuery()** al que se le pasa una cadena con la consulta SQL. **Los datos resultantes se devuelven como un objeto ResultSet**.

ResultSet result = stmt.executeQuery(sentenciaSQL);



ResultSet

Para obtener datos almacenados en la BD se utiliza una consulta SQL (query). La consulta se puede ejecutar utilizando el objeto **Statement, con el método executeQuery()** al que se le pasa una cadena con la consulta SQL. **Los datos resultantes se devuelven como un objeto ResultSet**.

ResultSet rs = stmt.executeQuery(sentenciaSQL);

```
// Crear sentencia SQL
String sql = "SELECT * FROM products";

// Preparar sentencia
Statement stmt = conn.createStatement();


// Ejecutar consulta
ResultSet rs = stmt.executeQuery(sql);
```

ResultSet

La consulta SQL devolverá una tabla que tendrá una serie de campos y un conjunto de registros, cada uno de los cuales consistirá en una tupla de valores correspondientes a los campos de la tabla.

El objeto **ResultSet proporciona el acceso a los datos de estas filas mediante un conjunto de métodos get** que permiten el acceso a las diferentes columnas de las filas.

El método ResultSet.next se usa para moverse a la siguiente fila del ResultSet, convirtiendo a ésta en la fila actual.

ResultSet

La consulta SQL devolverá una tabla que tendrá una serie de campos y un conjunto de registros, cada uno de los cuales consistirá en una tupla de valores correspondientes a los campos de la tabla.

El objeto **ResultSet proporciona el acceso a los datos de estas filas mediante un conjunto de métodos get** que permiten el acceso a las diferentes columnas de las filas.

El método ResultSet.next se usa para moverse a la siguiente fila del ResultSet, convirtiendo a ésta en la fila actual.

```
(rs.getString("nombre"))
(rs.getDouble("precio"))
```

ResultSet

El formato general de un ResultSet es una tabla con cabeceras de columna y los valores correspondientes devueltos por la “query”.

Por ejemplo, si la “query” es SELECT a, b, c FROM Table1, el resultado tendrá una forma semejante a:

a	b	c
12345	Chile	10,5
31245	Brasil	22,7
47899	Perú	56,7

ResultSet

El siguiente fragmento de código es un ejemplo de la ejecución de una sentencia SQL que devolverá una colección de filas, con la columna 1 como un int, la columna 2 como una String y la columna 3 como un valor real:

a	b	c
12345	Chile	10,5
31245	Brasil	22,7
47899	Perú	56,7

```
Statement stmt = connection.createStatement();
ResultSet r = stmt.executeQuery("SELECT a, b, c FROM Table1");
while (r.next()) {
    int i = r.getInt("a");
    String s = r.getString("b");
    double d = r.getDouble("c");
    // imprimimos los valores de la fila actual
    System.out.println("Fila = " + i + " " + s + " " + d);
}
```

ResultSet

Filas ResultSet

Un ResultSet mantiene un cursor que apunta a la fila actual de datos. El cursor se mueve **una fila hacia abajo cada vez que se llama al método .next**.

Las filas de ResultSet se recuperan en secuencia desde la fila más alta a la más baja.

```
while(rs.next()) {
  Product p = new Product();
  p.setId(rs.getInt("id"));
  p.setNombre(rs.getString("nombre"));
  p.setPrecio(rs.getDouble("precio"));
```

ResultSet

Columnas ResultSet

Los métodos **getX** suministran los medios **para recuperar los valores de las columnas** de la fila actual. Dentro de cada fila, los valores de las columnas pueden recuperarse en cualquier orden, pero para asegurar la máxima portabilidad, deberían extraerse las columnas de izquierda a derecha y leer los valores de las columnas una única vez. **Puede usarse, o bien el nombre de la columna o el número de columna, para referirse a esta**. Por ejemplo: si la columna segunda de un objeto ResultSet rs se denomina “nombre” y almacena valores de cadena, cualquiera de los dos ejemplos siguientes nos devolverá el valor almacenado en la columna.

```
String s = rs.getString("nombre");
String s = rs.getString(2);
```

```
while(rs.next()) {
  Product p = new Product();
  p.setId(rs.getInt("id"));
  p.setNombre(rs.getString("nombre"));
  p.setPrecio(rs.getDouble("precio"));
```

ResultSet

Las columnas **se enumeran de izquierda a derecha** comenzando con la columna 1.
Además, **los nombres** usados como input en los métodos getX s**on insensibles a las mayúsculas**.
Algunos de los datos que podemos traer con el método get son:

getInt()	Sirve para obtener un numero entero de la base de datos
getLong()	Sirve para obtener un número long de la base de datos
getDouble()	Sirve para obtener un número real de la base de datos
getBoolean()	Sirve para obtener un booleano de la base de datos
getString()	Sirve para obtener una cadena de la base de datos
getDate()	Sirve para obtener una fecha de la base de datos

LIVE CODING

Ejemplo en vivo

¡Utilizando ResultSet!

Vamos a trabajar en la clase que creamos con la conexión Singleton. Esta clase tiene métodos para conectarse, desconectarse y realizar una consulta (sentencia).

Ahora vamos a agregar el objeto ResultSet para que nos devuelva los resultados de la sentencia.

Tiempo: 20 minutos

Modificación de datos

Modificación de datos

El objeto **Statement también posee el método executeUpdate**: recibe como parámetro la cadena de caracteres que contiene la sentencia SQL a ejecutar.

Este método **únicamente permite realizar sentencias de actualización** de la BD: creación de tablas (**CREATE**), inserción (**INSERT**) y borrado de datos (**DELETE**). El método a utilizar es el siguiente:

```
stmt.executeUpdate(sentenciaSql);
```

Modificación de datos

Vamos a ver a continuación un ejemplo de estas operaciones.

Crearemos una tabla ALUMNOS en nuestra base de datos y añadiremos datos a la misma. La sentencia para la creación de la tabla será la siguiente:

```
// Creamos una tabla llamada Alumnos
String st_create = "CREATE TABLE ALUMNOS ("
                 + " expediente INTEGER,"
                 + " nombre VARCHAR(32),"
                 + "PRIMARY KEY (expediente)"
                 + ")";

stmt.executeUpdate(st_create);
```

Modificación de datos

Una vez creada la tabla podremos insertar datos en ella:

```
// Insertamos datos
String st_inserta = "INSERT INTO ALUMNOS(expediente, nombre)"
                  + "VALUES(1285, 'Manu')";

stmt.executeUpdate(st_inserta);
```

Modificación de datos

Cuando tengamos datos dentro de la tabla, podremos modificarlos utilizando para ello una sentencia **UPDATE**:

```
// Modificamos el registro
String st_actualiza = "UPDATE FROM ALUMNOS"
                    + "SET nombre = 'Nadu'"
                    + "WHERE expediente = 1285";

stmt.executeUpdate(st_actualiza);
```

Modificación de datos

Si queremos eliminar un registro de la tabla utilizaremos una sentencia DELETE:

```
// Borramos el registro
String st_borra = "DELETE FROM ALUMNOS"
                + "WHERE expediente = 1285";

stmt.executeUpdate(st_borra);
```

El método executeUpdate **retorna un entero** que nos dice el número de registros a los que ha afectado la operación, en caso de sentencias INSERT, UPDATE y DELETE. La creación de tablas retorna siempre 0.

LIVE CODING

Ejemplo en vivo

¡Modificando Registros!

- Vamos a crear una nueva tabla en la DB llamada “Cuenta”.
  - ↪ Esta tabla tendrá las columnas: id, saldo y fecha.
- Luego, vamos a modificar el saldo.

Tiempo: 20 minutos

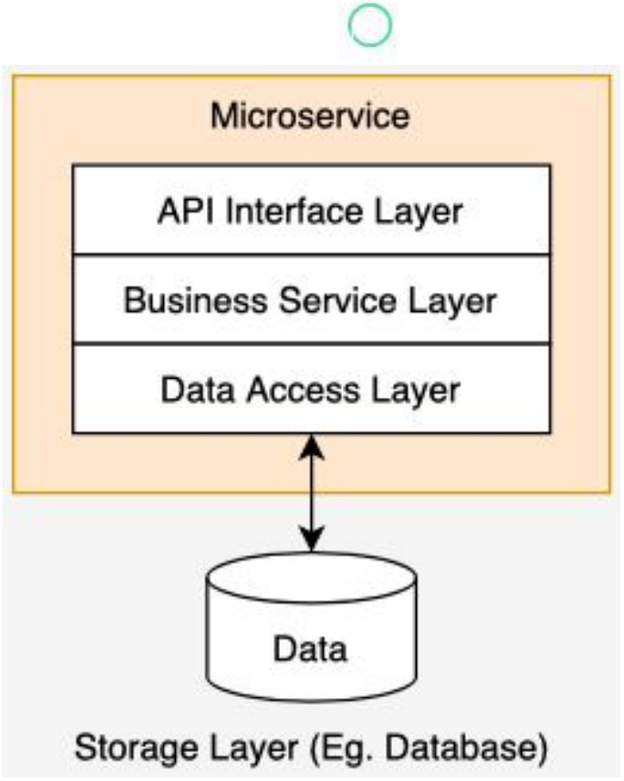
Capa de Acceso a Datos (DAL)

Capa de Acceso a Datos (DAL)

La capa de acceso a datos (**D**ata **A**ccess **L**ayer) es un componente de una arquitectura de software que es responsable de administrar el almacenamiento y la recuperación de datos de una aplicación.

Se encuentra entre la capa de lógica de negocios y el sistema de almacenamiento de datos.

El DAL está diseñado para ser reutilizable e independiente de la lógica comercial y la implementación del almacenamiento de datos. Se puede implementar utilizando diferentes tecnologías de acceso a datos y proporciona soporte para transacciones y otras funciones de acceso a datos.



Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro....

Trabajo Integrador del Módulo 💪🏽

Iremos completándolo progresivamente clase a clase.

English Always | alkemy

Ejercicio N° 1

Cambios y Resultados

Singletoneando

Manos a la obra: 🙌

Vamos a modificar el código para agregar los objetos ResultSet correspondientes.

Te invitamos a que generes algunos registros en tu BD Wallet para poder probar las consultas.

Consigna: ✍️

1- Agregar los objetos ResultSet para atrapar los resultados de las sentencias escritas.

2- Crear una tabla USUARIO con columnas “id, name, password, email, country”

3- Insertar cuatro registros de usuario.

4- Modificar el nombre de al menos dos usuarios.

Tiempo🕐: *30 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ *Comprender la ejecución de sentencias de modificación*
- ✓ *Aprender la implementación del objeto ResultSet.*
- ✓ *Conocer la utilización de la Capa de Acceso a Datos*

English Always | alkemy

#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 👇👇👇

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
   a. *Lectura Modulo 5, Lección 4: páginas 7 - 11*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 👋🏻

English Always | alkemy

Momento:

Time-out!

⏰5 min.