Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

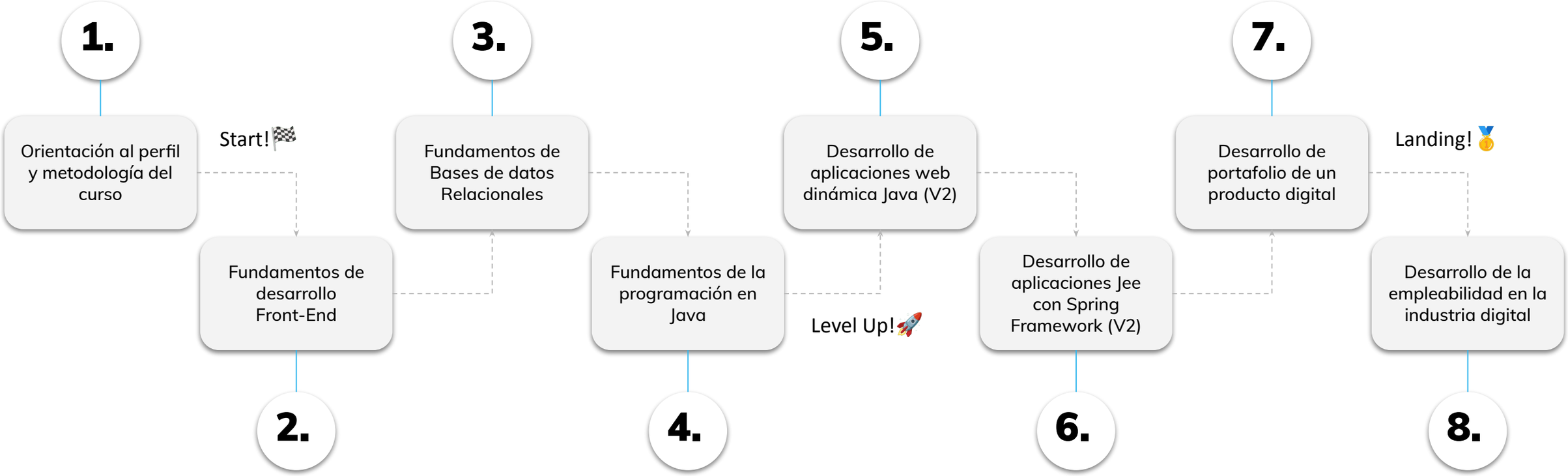
Capa de Acceso a Datos

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 📚:

- ✔ *Paso de parámetros desde un servlet a una vista JSP*
- ✔ *Validación de formulario en servlet*

English Always | alkemy

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

5.4 Start! 🏁

Capa de acceso a datos

Accesando datos en una aplicación web dinámica
Conexión a una Base de Datos mediante la biblioteca JDBC

Conexión a JDBC

Conectando

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

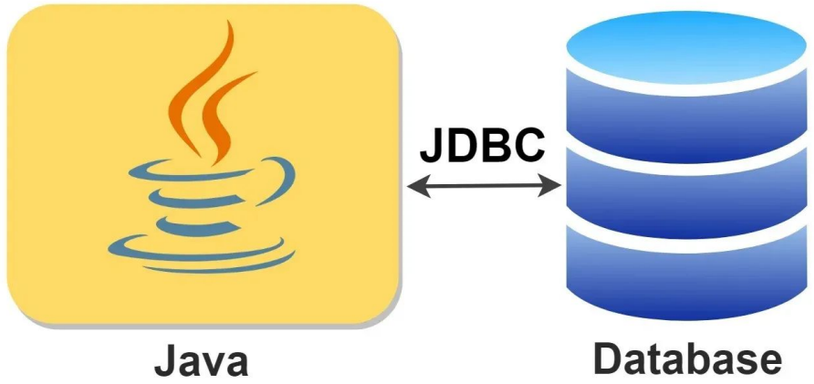
- *Acceder a datos en una aplicación web dinámica*
- *Conectar la aplicación a una base de datos mediante JDBC*

Accesando a datos en una aplicación web dinámica

Accesando a datos

Java Database Connectivity (**JDBC**) **permite que los programas Java accedan a sistemas de gestión de bases de datos**. La API JDBC consiste en un conjunto de interfaces y clases escritas en el lenguaje de programación Java.

Con estas interfaces y clases estándar, los programadores pueden escribir aplicaciones que se conecten con bases de datos, envíen consultas escritas en el lenguaje de consulta estructurada (SQL) y procesen los resultados.

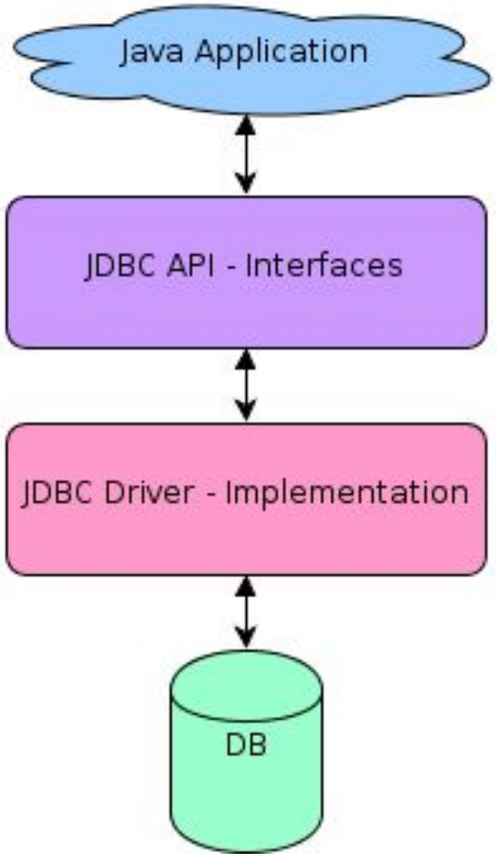


Accesando a datos

En general, hay dos **componentes principales de JDBC** a través de los cuales puede interactuar con una base de datos. Son los que se mencionan a continuación:

- **JDBC Driver Manager**: carga el driver específico de la base de datos en una aplicación para establecer una conexión con una base de datos. Se utiliza para realizar una llamada específica a la base de datos para procesar la solicitud del usuario.

- **API JDBC**: Es un conjunto de interfaces y clases, que proporciona varios métodos e interfaces para una fácil comunicación con la base de datos. Proporciona dos paquetes de la siguiente manera que contiene las plataformas java SE y java EE para exhibir capacidades WORA (write once run everything)



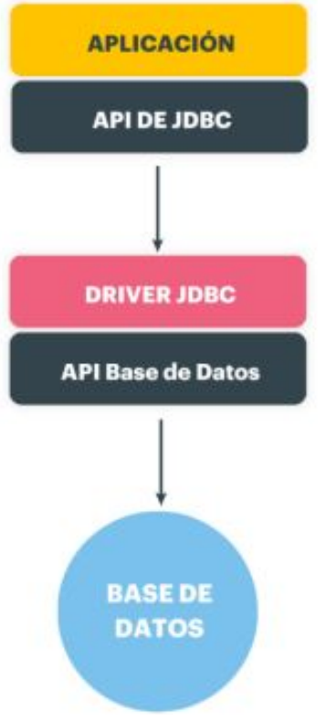
Accesando a datos

¿Cómo accede JDBC a la base de datos?

JDBC nos permitirá acceder a bases de datos desde Java. Para ello necesitaremos contar con un SGBD (sistema gestor de bases de datos) además de un driver específico para poder acceder a este SGBD.

La ventaja de JDBC es que nos permitirá acceder a cualquier tipo de base de datos, siempre que contemos con un driver apropiado para ella.

Como se observa en la imagen, cuando se construye una aplicación Java utilizando JDBC para el acceso a una base de datos, en la aplicación siempre se utiliza la API estándar de JDBC, y la implementación concreta de la base de datos será transparente para el usuario.



Accesando a datos

Componentes de la API JDBC

- **Driver**: Es el enlace que maneja toda la comunicación con la base de datos. Normalmente, una vez que se carga el controlador, el desarrollador no necesita llamarlo explícitamente.

- **Connection**: Es una interfaz con todos los métodos para contactar una base de datos. El objeto de conexión representa el contexto de comunicación, es decir, toda la comunicación con la base de datos es sólo a través del objeto de Connection.

- **Statement**: Encapsula una instrucción SQL que se pasa a la base de datos para ser analizada, compilada, planificada y ejecutada.

- **ResultSet**: Los ResultSet representan un conjunto de filas recuperadas debido a la ejecución de una consulta.

Conexión a una Base de Datos mediante JDBC

Conexión a una DB con JDBC

Para comunicarnos con una base de datos utilizando JDBC, se debe en primer lugar establecer una conexión con la base de datos a través del driver JDBC apropiado. El API JDBC especifica la conexión en la interfaz java.sql.Connection.

La clase DriverManager permite obtener objetos Connection con la base de datos.

Para conectarse es necesario proporcionar:

- **URL de conexión**, que incluye:
    - o Nombre del host donde está la base de datos.
    - o Nombre de la base de datos a usar.
- **Nombre del usuario** en la base de datos.
- **Contraseña del usuario** en la base de datos

Conexión a una DB con JDBC

1- Cargar el driver

El driver se debe cargar para poder utilizarlo. Esto lo realiza el método estático forName() de la clase Class. En este caso cargamos el driver de MySQL que es la base de datos con la que trabajaremos.

```
try {
    Class.forName("com.mysql.jdbc.Driver");
} catch (ClassNotFoundException ex) {
    log.error("No se encontro el Driver MySQL para JDBC.");
}
```

Conexión a una DB con JDBC

2- Obtener la conexión

La conexión a la BD está encapsulada en un objeto Connection, y para su creación se debe proporcionar la **url** de la **BD** y el **username** y **password** para acceder a ella. El formato de la url variará según el driver que se utilice.

```
// Obtener conexión a la BD
// url será el servidor de la DB
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/nombre_DB", "usuario", "password");
```

Conexión a una DB con JDBC

El objeto Connection representa el contexto de una conexión con la base de datos, es decir:

- Permite obtener objetos Statement para realizar consultas SQL.
- Permite obtener metadatos acerca de la base de datos (nombres de tablas, etc.)
- Permite gestionar transacciones

El siguiente código muestra un ejemplo de conexión y obtención de datos en JDBC a una base de datos MySQL.

```
try {
    Class.forName("com.mysql.jdbc.Driver");
    String url = "jdbc:mysql:// hostname/database-name";
    connection = DriverManager.getConnection(url, "user", "password");
} catch (SQLException ex) {
    connection = null;
    ex.printStackTrace();
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}
```

LIVE CODING

Ejemplo en vivo

Conectando: *Vamos a crear una clase donde crearemos un método que retorne la conexión a la base de datos.*

Debemos crear una DB en MySQL para poder probar la conexión

Tiempo: 20 minutos

English Always | alkemy

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

Conectando

Conectando

Manos a la obra: 🙌

Es hora de realizar nuestra primer conexión con la base de datos! Para esto es necesario que crees una nueva database MySQL llamada wallet.

Consigna: ✍️

1- Crear la BD Wallet.

2- Crear un nuevo paquete en el proyecto AlkeWallet donde guardaremos los archivos de conexión.

3- Crear una clase que manejará conexiones y que contenga el método createConnection que debe retornar la conexión exitosa a la BD.

Tiempo🕐: *30 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ *Conocer el acceso a datos desde la aplicación*
- ✓ *Aprender a configurar JDBC para conectarse a una base de 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 1 - 4*
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.