Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

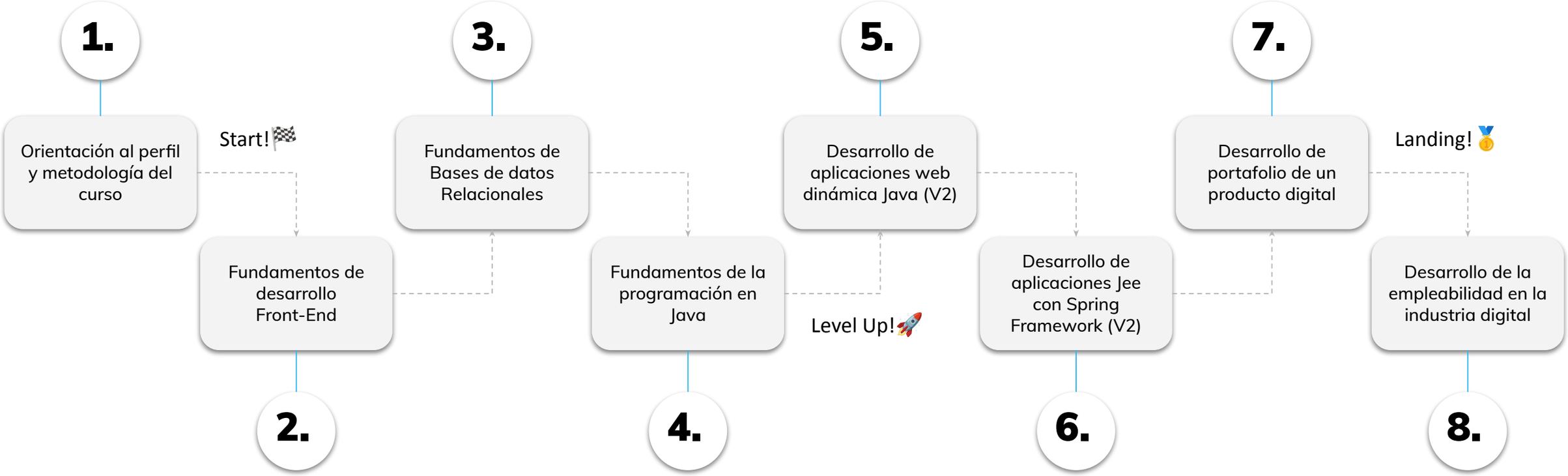
Capa de Acceso a Datos

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 📚:

- ✓ *Objeto ResultSet*
- ✓ *Modificación de datos*
- ✓ *DAL*

English Always | alkemy

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

5.4 Start! 🏁

Capa de acceso a datos

DAO

CRUD

El patrón DAO
Data Transfer Object
Implementando un DAO con métodos CRUD

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

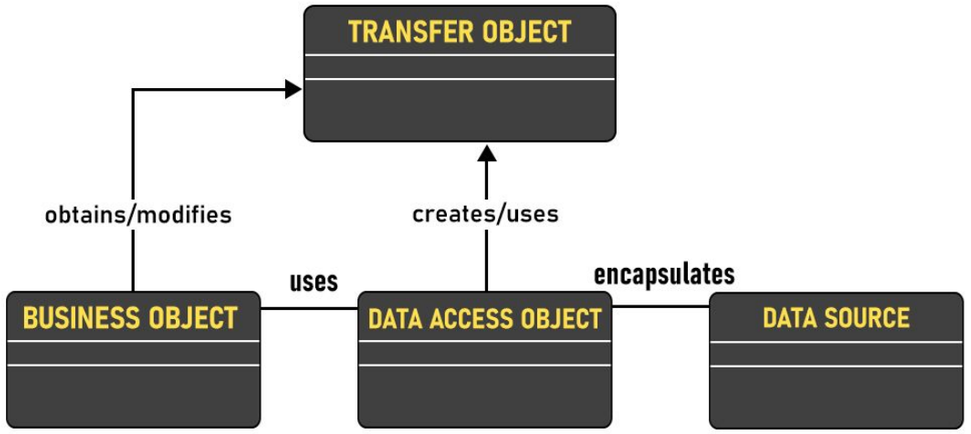
¿Qué aprenderemos?

- *Implementar una clase DAO para interactuar con la base de datos*
- *Conocer el patrón DTO*

El patrón de diseño DAO

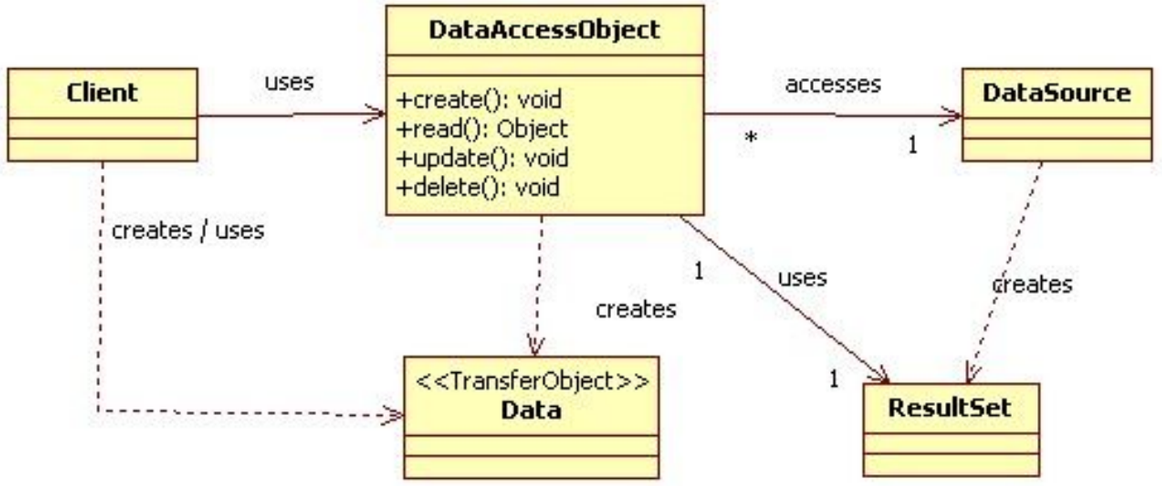
DAO

El patrón **Data Access Object** (DAO) pretende principalmente independizar la aplicación de la forma de acceder a la base de datos, o cualquier otro tipo de repositorio de datos. Para ello **se centraliza el código relativo al acceso al repositorio de datos en las clases llamadas DAO**. Fuera de las clases DAO no debe haber ningún tipo de código que acceda al repositorio de datos.



DAO

El patrón DAO propone separar por completo la lógica de negocio de la lógica para acceder a los datos, de esta forma, el DAO proporcionará los métodos necesarios para **insertar**, **actualizar**, **borrar** y **consultar (CRUD)** la información.



DAO

Esto lo vamos a lograr a través de cuatro clases:

- **Entidad**: va a ser la clase que va a representar a la tabla que queremos trabajar de la base de datos. Va tener como atributos las columnas de la tabla de la base de datos.

- **DAO**: esta clase va a ser la encargada de comunicarse con la base de datos, de conectarse con la base de datos, enviar las consultas y recuperar la información de la base de datos.

- **Servicio o Business Service**: va a tener toda la lógica de negocio del proyecto, usualmente se genera una para cada entidad. Es la que se encarga de obtener datos desde la base de datos utilizando el DAO y enviarla al cliente, o a su vez recibir la clase desde el cliente y enviar los datos al servidor, por lo general tiene todos los métodos CRUD (create, read, update y delete).

DAO

EntidadDaoImpl: esta clase va a extender de la clase DAO y se va encargar de generar las sentencias para enviar a la clase DAO, como un insert, select, etc. Y si estuviéramos haciendo un select, sería también, la encargada de recibir la información, que recupera la clase DAO de la base de datos sobre una entidad, para después enviarla al servicio, que será la encargada de imprimir dicha información.

Este es un objeto plano que implementa el patrón **Data Transfer Object** (**DTO**), el cual sirve para transmitir la información entre el DAO y el Business Service.

El patrón DTO

DTO

Un DTO u objeto de transferencia de datos es simplemente una clase plana sin ningún tipo de lógica de negocio en su interior, sólo es usado para guardar datos dentro de él.

Por norma general los DTO son clases con atributos con la palabra clave final, es decir son clases inmutables. Además, suelen implementar la interfaz Serializable ya que pueden ser utilizados para convertiros en bytes y enviar o obtener la información de alguna fuente.

```java
public class UsuarioDTO implements Serializable {
    private final String nombre;
    private final String apellidos;
    // constructor y otros métodos
}
```

DTO

Diferencias entre DTO y entidades

Los DTO y las entidades se utilizan comúnmente en aplicaciones Java, pero tienen diferentes propósitos.

Las **entidades** son **objetos que representan el estado de los datos en una base de datos** y normalmente se utilizan para implementar la **capa de persistencia** de una aplicación. Incluyen propiedades que se asignan a columnas en una tabla de base de datos, así como métodos para acceder y manipular los datos. Las entidades suelen estar diseñadas para ser persistentes, lo que significa que su estado puede almacenarse y recuperarse de una base de datos.

DTO

Diferencias entre DTO y entidades

Los **DTO**, por otro lado, **se utilizan para la transferencia de datos entre capas** de una aplicación y no se preocupan por la persistencia. Están diseñados para ser simples y livianos, sin ninguna lógica de negocios, e incluyen sólo los datos necesarios para las necesidades específicas de cada capa. Los DTO se utilizan normalmente para transferir datos desde una capa de servicio a una capa de presentación o entre microservicios.

Implementando un DAO con métodos CRUD

Implementando CRUD

Veamos un ejemplo de clases DAO. Esto lo vamos a lograr a través de las cuatro clases vistas anteriormente:

1- **Entidad**: va a ser la clase que va a representar a la tabla que queremos trabajar de la base de datos.

```
public class Usuario {

    private String nombre;
    private String correoElectronico;
    private Integer edad;
```

Implementando CRUD

2- **DAO**: representa una capa de acceso a datos que oculta la fuente y los detalles técnicos para recuperar los datos. Es la clase donde tenemos los objetos de conexión, sentencia y resultado.

```
public class DAO {
    //OBJETO CONNECTION ---> Encargado de INICIAR - TENER - MANTENER la conexion
    protected Connection conexion = null;
    //OBJETO RESULSET ---> Guardar todos los datos que llegan de la DB (las filas de la consulta)
    protected ResultSet resultado = null;
    //OBJETO STATEMENT ---> "Tiene" las consultas... es donde generamos las sentecias a ejecutar!
    protected Statement stmt = null;

    private final String user = "root";
    private final String password= "root";
    private final String database = "nombre_database";


    protected void conectarBase(){


    protected void consultarBase(String sql) {

    protected void desconectarBase(){

}
```

Implementando CRUD

3- **EntidadDAO**: esta clase va a extender de la clase DAO y se va encargar de generar las sentencias SQL para enviar a la clase DAO. Este es un objeto plano que implementa el patrón Data Transfer Object (DTO), el cual sirve para transmitir la información entre el DAO y el Business Service.

```
public class UsuarioDAO extends DAO {

    public List<Usuario> listarUsuario() throws Exception{
        try {
            //armado de la sentencia SQL
            String sql = "SELECT nombre,correoElectronico,edad "
                    + "FROM Usuario; ";

            //consultamos la DB haciendo uso del método del DAO
            consultarBase(sql);

            //Recorro el resultado de la consulta
            Usuario usuario = null;
            List<Usuario> usuarios = new ArrayList<>();

            while(resultado.next()){
                usuario = new Usuario();


                usuario.setNombre(resultado.getString(2));
                usuario.setCorreoElectronico(resultado.getString(3));
                usuario.setEdad(resultado.getInt(4));

                //lo agrego a la list!
                usuarios.add(usuario);
            }

            return usuarios;
```

Implementando CRUD

4- **Servicio o Business Service**: por lo general tiene todos los métodos CRUD (create, read, update y delete).

```java
public class UsuarioService {
    //llamado a UsuarioDAO
   private UsuarioDAO dao;
   // constructor para inicializarlo

    public UsuarioService() {
        this.dao = new UsuarioDAO();
    }

    public List<Usuario> listarUsuario() throws Exception{
        try {

        List<Usuario> usuarios = dao.listarUsuario();

        return usuarios;

        } catch (Exception e) {
            throw new Exception("Error en listar usuario Servicio");
        }
    }
```

LIVE CODING

Ejemplo en vivo

¡A CRUDear!

Vamos a crear los métodos necesarios para crear, insertar, eliminar y modificar los datos de nuestra aplicación.

Para ello vamos a mejorar lo que veníamos haciendo, agregando las clases DAO necesarias e implementando lo aprendido.

Tiempo: 25 minutos

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

CRUD

Singletoneando

Manos a la obra: 🙌

¡Finalmente vamos a unificar lo aprendido! Vas a trabajar sobre un nuevo proyecto donde organizarás las clases nuevamente para respetar el patrón DAO.

Consigna: ✍️

1- Crear las clases necesarias para interactuar con UserDAO, AccountDAO y CurrencyDAO .

2- Crear una nueva BD Wallet e insertar registros.

3- Modificar el nombre de al menos dos usuarios.

Tiempo🕐: *30 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ *Comprender la implementación del patrón DAO*
- ✓ *Aprender el concepto de los DTOs.*

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 11 - 16*
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.