Capa de acceso a datos y objetos DAO

¡Comencemos!

¿Qué aprenderás?

- Construir Curso DAO
- Construir Forma Pago DAO
- Construir Inscripción DAO
- Construir y utilizar Facade

Introducción

En la capa de acceso a datos se llevan a cabo las tareas básicas para interactuar con las bases de datos del sistema. Antes de comenzar con el código hay que aclarar algunos nuevos conceptos que entran en juego en esta importante etapa del desarrollo de sistemas.

DAO

Sigla de Data Access Object, o en español objeto de acceso a datos. Es una clase especial que se encarga del acceso directo al motor de base de datos la cual es frecuentemente usada por las clases de las capas superiores (capa de lógica de negocio). Cada vez que la aplicación necesite acceder a los datos ya sea para guardar, obtener, modificar o eliminar sus registros debe pasar obligatoriamente por una clase DAO, la cual tiene acceso directo a la persistencia mediante jdbc en el caso de este proyecto. Por cada entidad de negocio debe existir una clase dao tal como se codifica a continuación en el proyecto.

DTO

Sigla de Data Transfer Object, trabajan codo a codo con los DAOs los cuales los utilizan para transportar los datos desde el motor de datos hasta la capa lógica de negocio. El objeto DTO es un pojo común y corriente que tiene la misma estructura que la tabla a la cual se está accediendo.

Es momento de generar la conexión a la base de datos mediante el API JDBC siguiendo las instrucciones dadas en el apartado anterior. A continuación se muestra el código del archivo CursoDAO, el cual es parte del package com.desafiolatam.daos.

Construcción Curso Dao

Clase CursoDao, se comunica con la base de datos

```
package com.desafiolatam.daos;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import com.desafiolatam.entidades.CursoDTO;

public class CursoDao {

  /**
   * Metodo encargado de obtener todos los cursos disponibles
   * @return Lista de cursos CursoDTO
```

```
        * @throws SQLException
        * @throws ClassNotFoundException
        * @autor developer
        */
       public List obtieneCursos() throws SQLException,
ClassNotFoundException {

            //creamos la lista de objetos que devolveran los resultados
            List<CursoDTO> listaDeCursos = new ArrayList<CursoDTO>();

            //creamos la consulta a la base de datos
            String consultaSql = " SELECT id_curso, descripcion, precio
"
                                       + " FROM curso ";

            //conexion a la base de datos y ejecucion de la sentencia
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conexion = null;
            String url = "jdbc:oracle:thin:@//localhost:1521/xe";
            conexion =
DriverManager.getConnection(url,"Empresa_DB","empresa");

            try{PreparedStatement stmt =
conexion.prepareStatement(consultaSql);{

                  ResultSet resultado = stmt.executeQuery();
                  while(resultado.next()) {
                        CursoDTO cursoDto = new CursoDTO();

cursoDto.setIdCurso(resultado.getInt("id_curso"));

cursoDto.setDescricion(resultado.getString("descripcion"));

cursoDto.setPrecio(resultado.getDouble("precio"));
                        listaDeCursos.add(cursoDto);
                  }

            }catch(Exception ex) {
                  ex.printStackTrace();
            }
            return listaDeCursos;
      }
    }
```

- Dentro del ciclo while se crea un objeto de tipo CursoDTO para luego asignar los parámetros del curso al mismo mediante sus métodos setIdCurso.

```
    while(resultado.next()) {
                        CursoDTO cursoDto = new CursoDTO();

cursoDto.setIdCurso(resultado.getInt("id_curso"));

cursoDto.setDescricion(resultado.getString("descripcion"));

cursoDto.setPrecio(resultado.getDouble("precio"));
                        listaDeCursos.add(cursoDto);
                  }
```

Luego de crear el objeto Curso, simplemente lo agrega a una lista la cual es retornada.

Construcción Forma de pago DAO

A continuación se creará la clase FormaDePagoDao, que mantiene la misma lógica que CursoDao, con la diferencia que en vez de retornar una lista de cursos, retorna una lista de formas de pago.

Clase FormaDePagoDao, se comunica con la base de datos

```
    package com.desafiolatam.daos;

    import java.sql.Connection;
    import java.sql.DriverManager;
    import java.sql.PreparedStatement;
    import java.sql.ResultSet;
    import java.sql.SQLException;
    import java.util.ArrayList;
    import java.util.List;

    import com.desafiolatam.entidades.CursoDTO;
    import com.desafiolatam.entidades.FormaDePagoDTO;

    public class FormaDePagoDAO {

      /**
       * Metodo encargado de obtener todas las formas de pago
       * @return Lista formas de pago FormaDePagoDTO
```

```
       * @throws SQLException
       * @throws ClassNotFoundException
       * @autor desafioLatam
       */
      public List obtieneFormasDePago() throws SQLException,
ClassNotFoundException {

            //creamos la lista de objetos que devolveran los resultados
            List<FormaDePagoDTO> listaDeCursos = new
ArrayList<FormaDePagoDTO>();

            //creamos la consulta a la base de datos
            String consultaSql = " SELECT id_forma_pago, descripcion,
recargo "
                                       + " FROM forma_pago ";

            //conexion a la base de datos y ejecucion de la sentencia
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conexion = null;
            String url = "jdbc:oracle:thin:@//localhost:1521/xe";
            conexion =
DriverManager.getConnection(url,"Empresa_DB","empresa");

            try(PreparedStatement stmt =
conexion.prepareStatement(consultaSql)){

                  ResultSet resultado = stmt.executeQuery();
                  while(resultado.next()) {
                        FormaDePagoDTO formaPago = new FormaDePagoDTO();

formaPago.setIdFormaDePago(resultado.getInt("id_forma_pago"));

formaPago.setDescripcion(resultado.getString("descripcion"));

formaPago.setRecargo(resultado.getString("recargo"));
                        listaDeCursos.add(formaPago);
                  }

            }catch(Exception ex) {
                  ex.printStackTrace();
            }
            return listaDeCursos;
      }
    }
```

Esta clase mantiene la misma lógica que CursoDao, con la diferencia de que la consulta a la base de datos es a la tabla de formas de pago. La instrucción para generar la lista de formas de pago es igual, terminando con el retorno de la lista.

Finalmente necesitamos de una clase que inserte en la base de datos la selección del usuario, para eso se creará la clase InscripcionDAO que tendrá un método de nombre insertar.

Construccion Inscripción DAO

Clase InscripcionDAO

```
    package com.desafiolatam.daos;

    import java.sql.Connection;
    import java.sql.DriverManager;
    import java.sql.PreparedStatement;
    import java.sql.ResultSet;
    import java.sql.SQLException;
    import java.util.ArrayList;
    import java.util.List;

    import com.desafiolatam.entidades.CursoDTO;
    import com.desafiolatam.entidades.FormaDePagoDTO;
    import com.desafiolatam.entidades.InscripcionDTO;

    public class InscripcionDAO {

      /**
       * Metodo encargado de guardar la inscripcion de un curso
       * @return Lista formas de pago FormaDePagoDTO
       * @throws SQLException
       * @throws ClassNotFoundException
       * @autor desafioLatam
       */
      public int insertarInscripcion(InscripcionDTO dto) throws
SQLException, ClassNotFoundException {

            int max = 0;

            //Query para obtener el una secuencia y asignar un id
            String consultaProximoId = " SELECT MAX(id_inscripcion)+1 "
                                     + " FROM inscripcion ";
```

```
            //Query que insertara el valor
            String insertarInscripcion = " INSERT INTO inscripcion("
                                         + " idInsc, nombre, celular,
idCurso, idFormaDePago "
                                         + " VALUES (?,?,?,?,?) ";

            //conexion a la base de datos y ejecucion de la sentencia
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conexion = null;
            String url = "jdbc:oracle:thin:@//localhost:1521/xe";
            conexion =
DriverManager.getConnection(url,"Empresa_DB","empresa");

            try(
                        PreparedStatement stmt1 =
conexion.prepareStatement(consultaProximoId);
                        PreparedStatement stmt2 =
conexion.prepareStatement(insertarInscripcion);

               ){

                  ResultSet resultado = stmt1.executeQuery();
                  if(resultado.next()) {
                        max = resultado.getInt("id_inscripcion");
                        stmt2.setInt(1, max);
                        stmt2.setString(2, dto.getNombre());
                        stmt2.setString(3, dto.getCelular());
                        stmt2.setInt(4, dto.getIdCurso());
                        stmt2.setInt(5, dto.getIdFormaDePago());

                        if(stmt2.executeUpdate() != 1) {
                              throw new RuntimeException("A ocurrido un
error inesperado");
                        }
                  }

            }catch(Exception ex) {
                  ex.printStackTrace();
                  throw new RuntimeException("A ocurrido un error
inesperado" + ex);
            }
            return max;
      }
    }
```

Con estas clases DAO y DTO tenemos cubiertas las capas interiores de la aplicación, ahora nos queda avanzar a las capas superiores para programar el comportamiento del sistema.

Si miramos nuevamente el diagrama de capas y de clases vemos marcado en verde lo que hasta el momento tenemos:

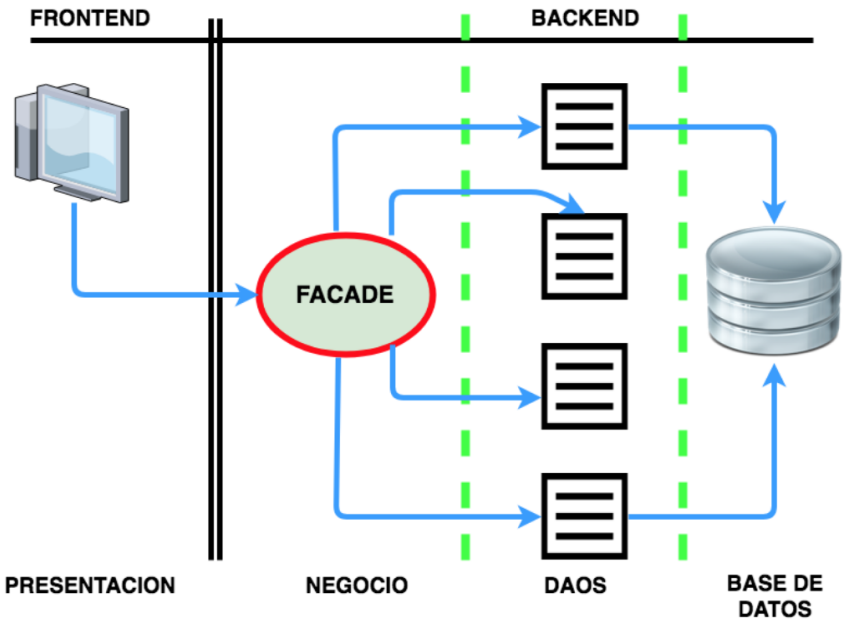

Imagen 1. Capa cubierta con clases DAOS y DTO.
Fuente: Desafío Latam

Facade, como se estudió anteriormente, el facade es un patrón de diseño que otorga el único punto de contacto entre el frontend y el backend. Trabaja codo a codo con la capa de presentación, facilitando el acceso a los servicios que darán la funcionalidad al sistema.

Por ejemplo, si una pantalla desea mostrar la lista de cursos disponibles en la base de datos, el facade debe tener un método a través del cual podamos acceder a los objetos de cursos DTO y su DAO correspondiente.

Si vemos las clases que se han programado hasta el momento, podemos saber que las acciones disponibles hasta el momento son: obtener cursos, obtener formas de pago y registrar una inscripción, por lo tanto se creará una clase de nombre Facade que tendrá declarados estos métodos.
Cabe mencionar que la clase Facade debe permanecer simple y no tener lógica, ya que esa responsabilidad en este caso cae en las clases DAO.

Clase Facade

```
    package com.desafiolatam.facade;

    import java.sql.SQLException;
    import java.util.List;

    import com.desafiolatam.daos.CursoDao;
    import com.desafiolatam.daos.FormaDePagoDAO;
    import com.desafiolatam.daos.InscripcionDAO;
    import com.desafiolatam.entidades.CursoDTO;
    import com.desafiolatam.entidades.FormaDePagoDTO;
    import com.desafiolatam.entidades.InscripcionDTO;

    public class Facade {

      public int registrarInscripcion(InscripcionDTO dto) throws
SQLException, ClassNotFoundException {
            InscripcionDAO dao = new InscripcionDAO();
            return dao.insertarInscripcion(dto);
      }

      public List<CursoDTO> obtenerCursos() throws SQLException,
ClassNotFoundException{
            CursoDao dao = new CursoDao();
            return dao.obtieneCursos();
      }

      public List<FormaDePagoDTO> obtenerFormasDePago() throws
SQLException, ClassNotFoundException{
            FormaDePagoDAO dao = new FormaDePagoDAO();
            return dao.obtieneFormasDePago();
      }
    }
```

Con la clase facade cubrimos toda la capa de backend, ahora toca trabajar en el front end en la próxima sesión.

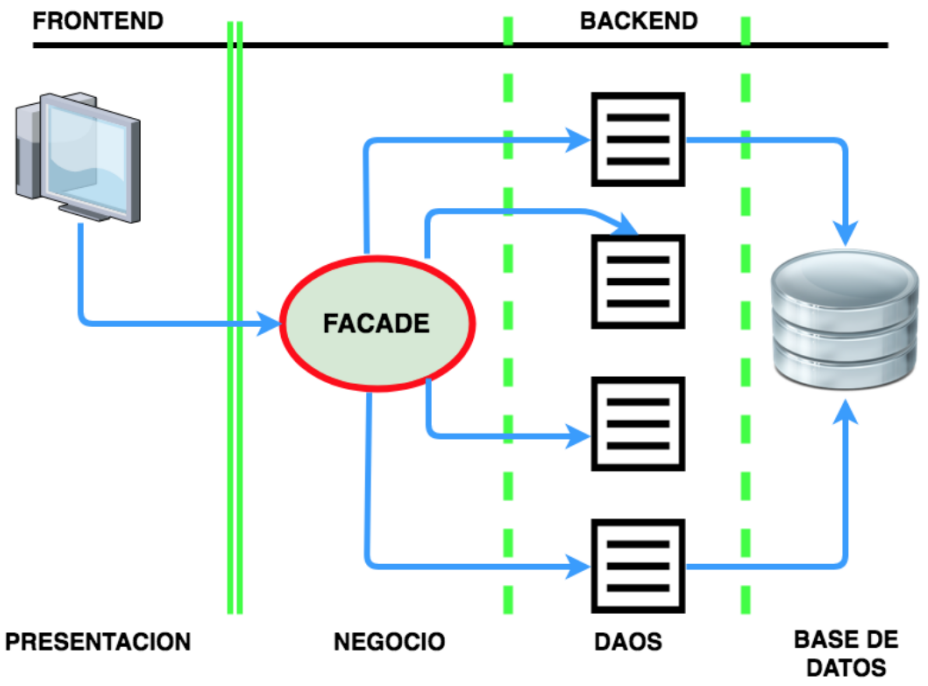

Imagen 2. Gracias al facade podemos comunicarnos con el backend.
Fuente: Desafío Latam

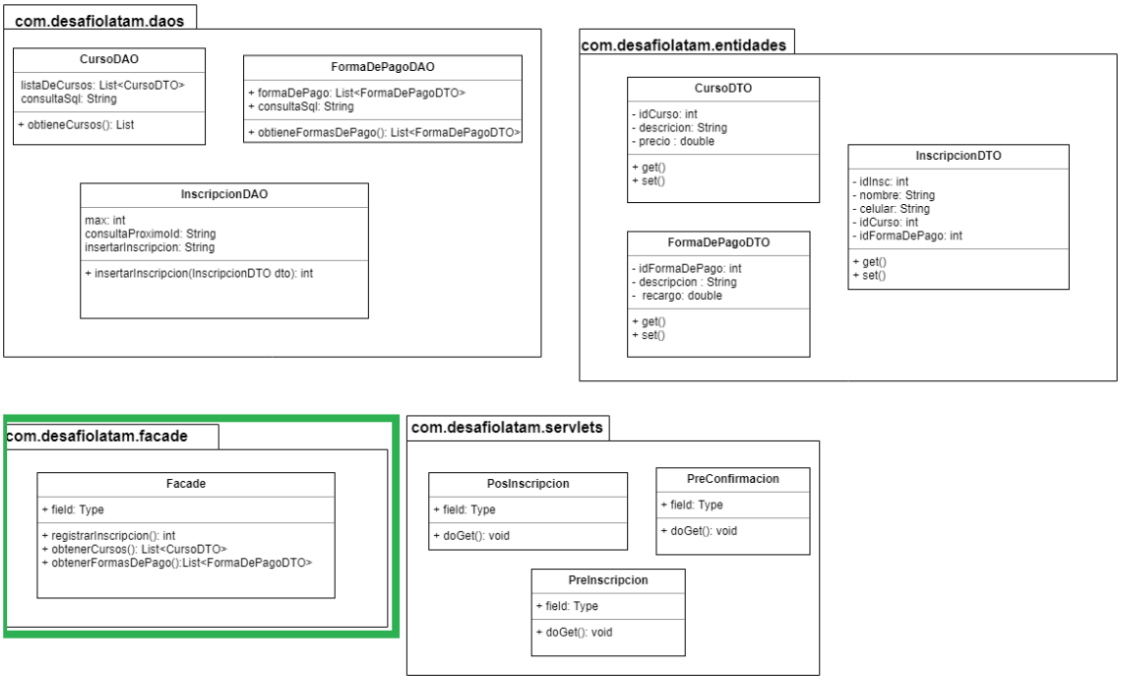

Imagen 3. Diagrama de clases - Facade.
Fuente: Desafío Latam