HOTELAPP

DICCIONARIO DE DATOS, DISEÑO E IMPLEMENTACIÓN

PROYECTO No.2 - BASES DE DATOS

Responsables:
Daniel Esteban Castellanos Echeverría
castellanose_daniele@javeriana.edu.co
.
Miguel Angel González Rodríguez
miguel_gonzalezr@javeriana.edu.co
.
Erick Santiago Garavito Villamil
erick.garavitov@javeriana.edu.co

Versión 1.0.0
27 DE MAYO DE 2021

Contenido

1. Diccionario de datos

A continuación, se tiene el diccionario de datos de las tablas utilizadas en la aplicación Hotelapp.

P2_login:

Nombre de tabla	**Descripción**		
P2_login	Tabla que almacena la información de las personas que poseen una cuenta en la aplicación		
columna	**dominio**	**características**	**Descripción**
Usuario	VARCHAR(20)	PK,NN	Nombre de usuario para login.
Password	VARCHAR(20)	NN	Contraseña de la cuenta.
Nombre	VARCHAR(45)	NN	Nombre de la persona.
Apellido	VARCHAR(45)	NN	Apellido de la persona.
Telefono	INT(15)	NN	Número de teléfono de la persona.
Nombre Índice	**Columna**	**Descripción**	
PRIMARY	Usuario	Índice de la clave primaria	
Nombre Clave	**Columna**	**Tabla referencia**	**Columna referencia**	**Reglas**
pk_login	Usuario	N/A	N/A	Clave primaria de la tabla persona

P2_Servicio:

Nombre de tabla	Descripción
P2_Servicio	Tabla que almacena la información de los servicios que una empresa presta a las comunidades.

columna	dominio	características	Descripción
codigo	NUMBER(6)	PK, NN	Número identificador del servicio prestado.
Descripcion	VARCHAR2(300)	NN	Descripción del servicio prestado.
Precio	VARCHAR2(50)	NN	Costo del servicio

Nombre Índice	Columna	Descripción
PRIMARY	codigo	Índice ascendente de la clave primaria

Nombre Clave	Columna	Tabla referencia	Columna referencia	Reglas
pk_servicio	codigo	N/A	N/A	Clave primaria de la tabla servicio

P2_Departamento:

Nombre de tabla	Descripción
P2_Departamento	Tabla que almacena la información de los departamentos disponibles.

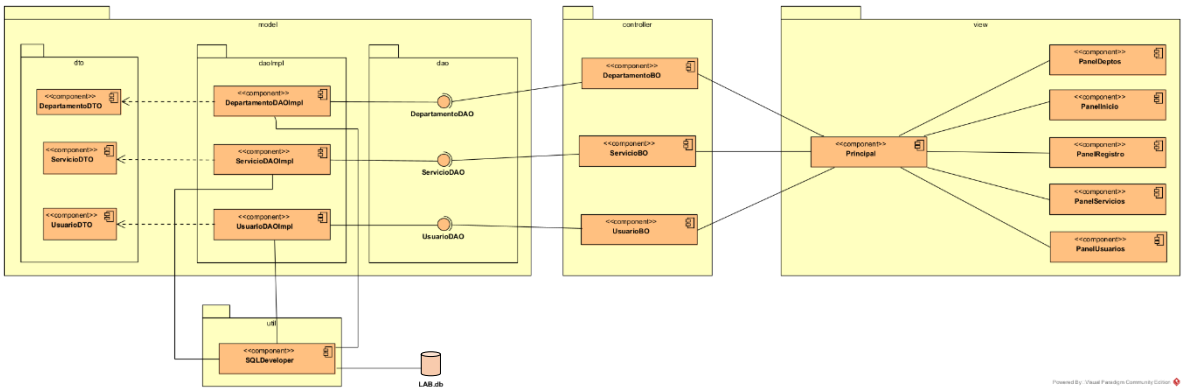
columna	dominio	características	Descripción
codigo	NUMBER(6)	PK, NN	Número identificador del departamento.
nombre	VARCHAR2(20)	NN	Nombre del departamento.

Nombre Índice	Columna	Descripción
PRIMARY	codigo	Índice ascendente de la clave primaria

Nombre Clave	Columna	Tabla referencia	Columna referencia	Reglas
pk_numero	numero	N/A	N/A	Clave primaria de la tabla servicio

2. Diagrama de componentes

En el diagrama de componentes, se evidencia el diseño de la aplicación implementado los patrones Modelo-Vista-Controlador (MVC), DAO y DTO.



En cuanto al patrón MVC, se dividió la aplicación en 3 paquetes:

- Modelo:
  En este paquete se alojaron las clases y paquetes encargados del acceso a los datos como la estructura de los datos, el acceso a los datos y la lógica de negocio. Para ello, se implementó el patrón DAO y DTO, por lo que se dividió en otros 3 paquetes como se explica a continuación:
  - DTO:
    Dentro de este paquete siguiendo el patrón DTO se ubicaron las clases que declaran la estructura de los datos junto a sus métodos constructores, *getters* y *setters*.

  - DAO:
    Dentro de este paquete siguiendo el patrón DAO se ubicaron las interfaces que tienen declarados los métodos que permitirán acceder a los datos, los métodos de CRUD.

  - DAOImpl:
    Dentro de este paquete siguiendo el patrón DAO, se tienen las clases que implementan los métodos definidos de las interfaces del paquete DAO, lo cual permite precisar la manera en cómo se accederá a los datos alojados en la base de datos.

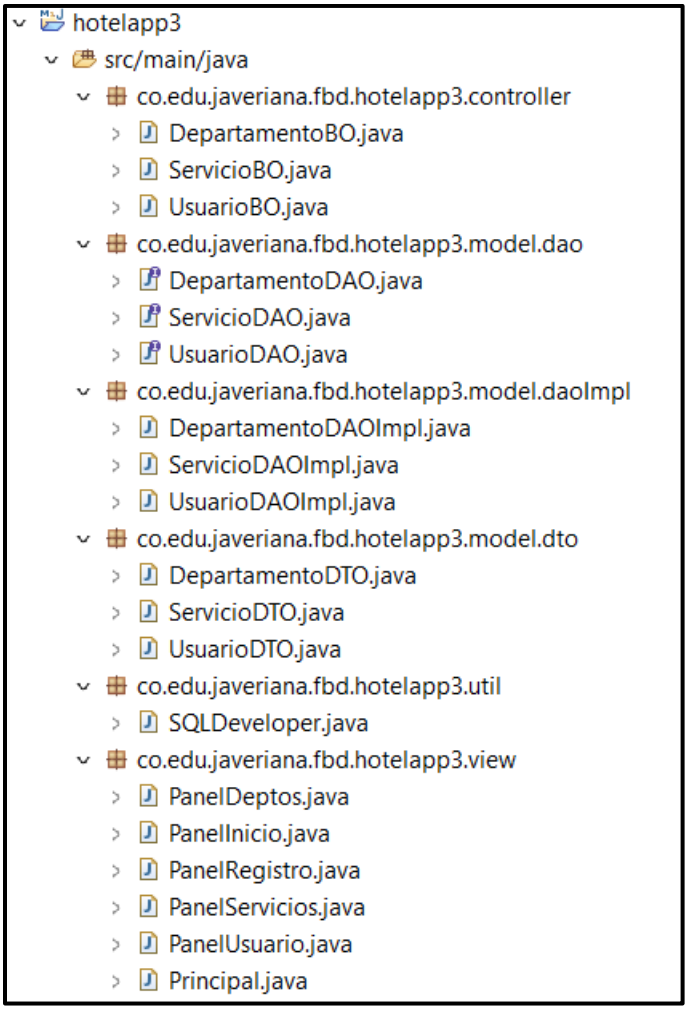
- Vista:
  En este paquete se tienen las clases que definen la manera en cómo se le presentará al usuario la información de la base de datos. Por medio de estas, el usuario podrá hacer uso de los métodos CRUD definidos en los paquetes dao y daoimpl.

- Controlador:
  En este paquete se tienen las clases que relacionan la vistas con los modelos. Se hace necesario hacer uso de este paquete, para permitir cambiar la capa de presentación de datos (vista) sin afectar la lógica de negocio, por lo que se es posible implementar el programa como una aplicación web, de escritorio o de móvil con tan solo cambiar la vista. Por medio de las clases BO, se enlaza la vista con el modelo, permitiendo acceder a los datos almacenados en la base de datos.

Además de estos 3 paquetes que representan cada una de las capas del patrón MVC, se tiene el paquete útil, en el cual se define la ruta de conexión hacia la base de datos, permitiendo que la aplicación cumpla su objetivo.

3. Implementación de la aplicación

La aplicación está diseñada en el lenguaje de programación JAVA y en la siguiente imagen se puede observar la implementación de los patrones MVC, DTO y DAO.



4. Conclusiones

- El patrón MVC es útil para el diseño de aplicaciones, debido a que, al separar la aplicación en 3 capas, permite organizar los diferentes elementos de esta. Además, permite que esta sea escalable, ya que, si se requiere ser implementada como aplicación web, de escritorio o móvil no se afectará la lógica de negocio, solo se agregará la nueva capa de presentación.

- El patrón DAO permite tener la declaración de los métodos CRUD en interfaces, las cuales se implementan en una clase aparte donde se define la lógica de negocio. La implementación de las interfaces con los métodos CRUD permite que sean implementados por otras clases que posiblemente accederán a los mismos datos, pero contenidos en otra base de datos, por lo que la lógica de negocio cambia.

- El patrón DTO, permite separar la estructura de los datos que se obtendrán de la base de datos de la lógica de negocio. Además, las clases que relacionan el modelo con la vista *Business Object* (BO), permiten que ante cualquier cambio en la vista la lógica de negocio sea la misma y que no sufra afectaciones.

- En general, es importante hacer uso de diferentes patrones de diseño que se ajusten a los requerimientos del sistema, para mejorar la escalabilidad, organizar los diferentes componentes y subsistemas y darle una organización lógica al programa a desarrollar.