DP1 2020-2021

Documento de Diseño del Sistema

Proyecto Talleres La Plata

< https://github.com/gii-is-DP1/dp1-2020-g2-11.git >

Miembros:

- Álvaro Alcón Granado
- Francisco José Anillo Carrasco
- Daniel Dorante Sánchez
- Alejandro Sevillano Barea
- Eugenio Benito Vicente Vázquez
- Jesús Viera Vázquez

Tutor: Carlos Guillermo Muller Cabezas

GRUPO G2-11

Versión 1.0

28-12-2020

Historial de versiones

Estos son ejemplo del contenido que debería tener el historial de cambios del documento a entregar a lo largo de los sprints del proyecto

Fecha	Versión	Descripción de los cambios	Sprint
28/12/2020	V1	● Creación del documento	3

Contents

Introducción

Este proyecto está basado en un taller, el objetivo requerido es conocer cuáles son las necesidades a satisfacer, que permita aportar más ventajas a la hora de realizar la gestión de su negocio. De esta manera, se centrará principalmente en concertar citas a través de Internet, cuestión que a día de hoy no se realiza, y facilitaría la tarea tanto para el propietario, como para el cliente del taller.

Además, guardar información acerca del problema de su vehículo, así como de la gestión administrativa con proveedores y clientes ofreciendo nuestra aplicación la posibilidad de organizar los pedidos que hace el taller a los diferentes proveedores.
La finalidad sería facilitar la gestión y el uso de los diferentes servicios que ofrece el taller para todos los usuarios de manera que se realice de forma óptima.

Diagrama(s) UML:

Diagrama de Dominio/Diseño

Diagrama de Cliente

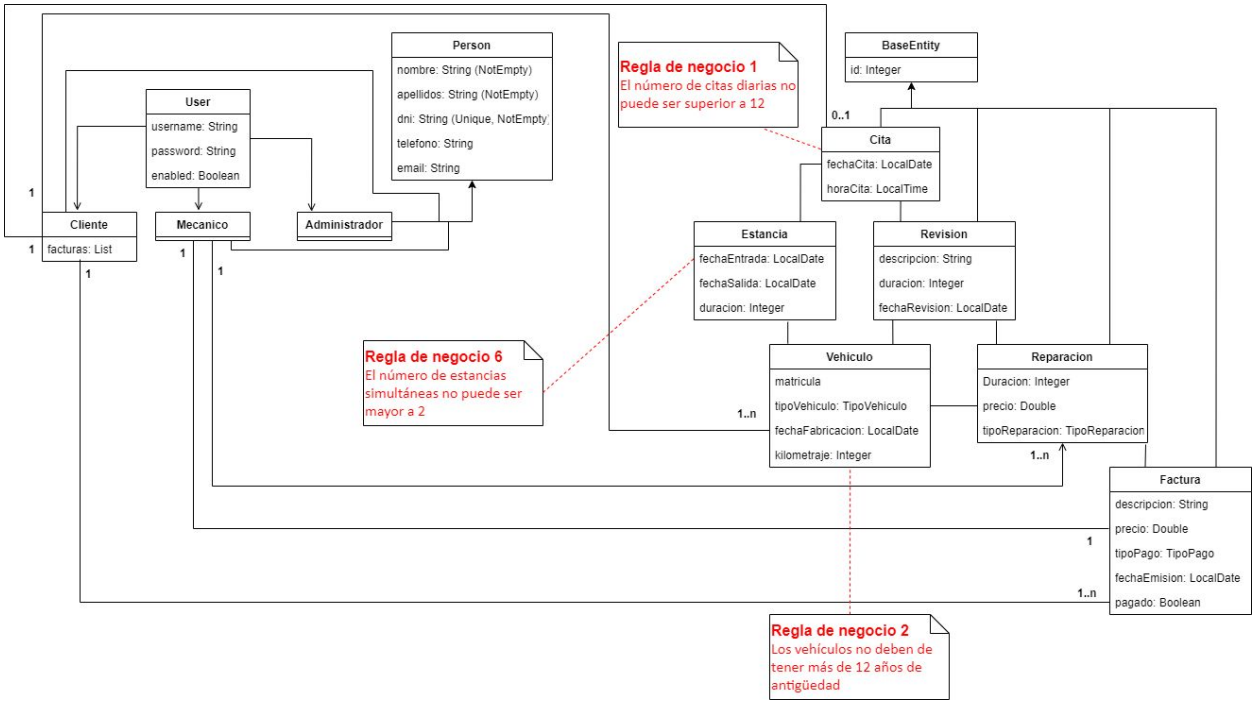


Diagrama de Pedido

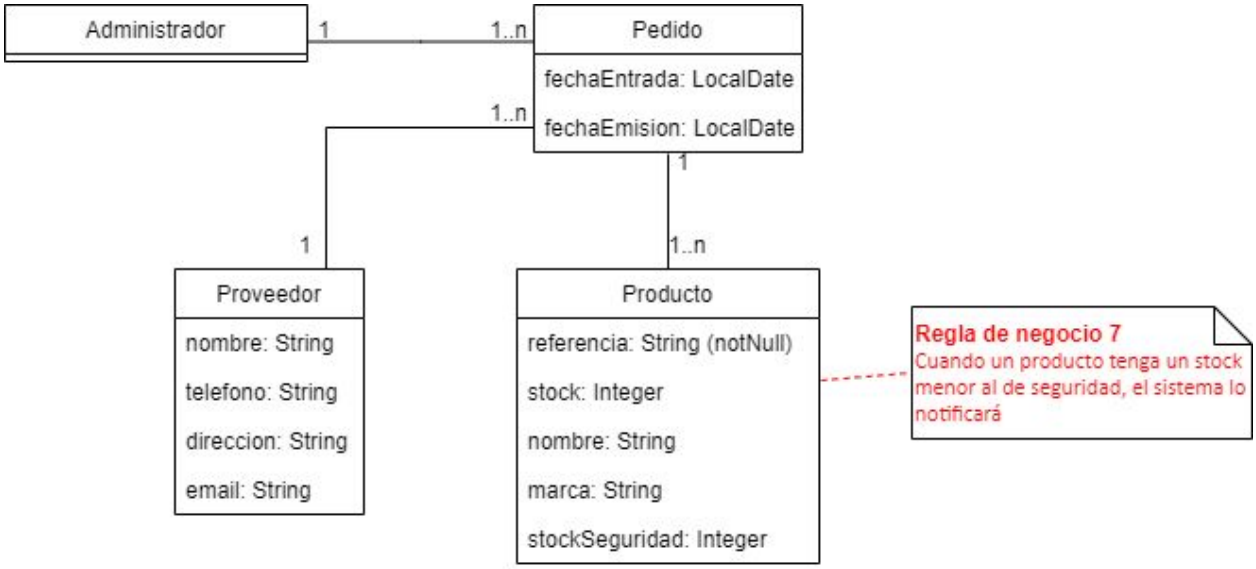
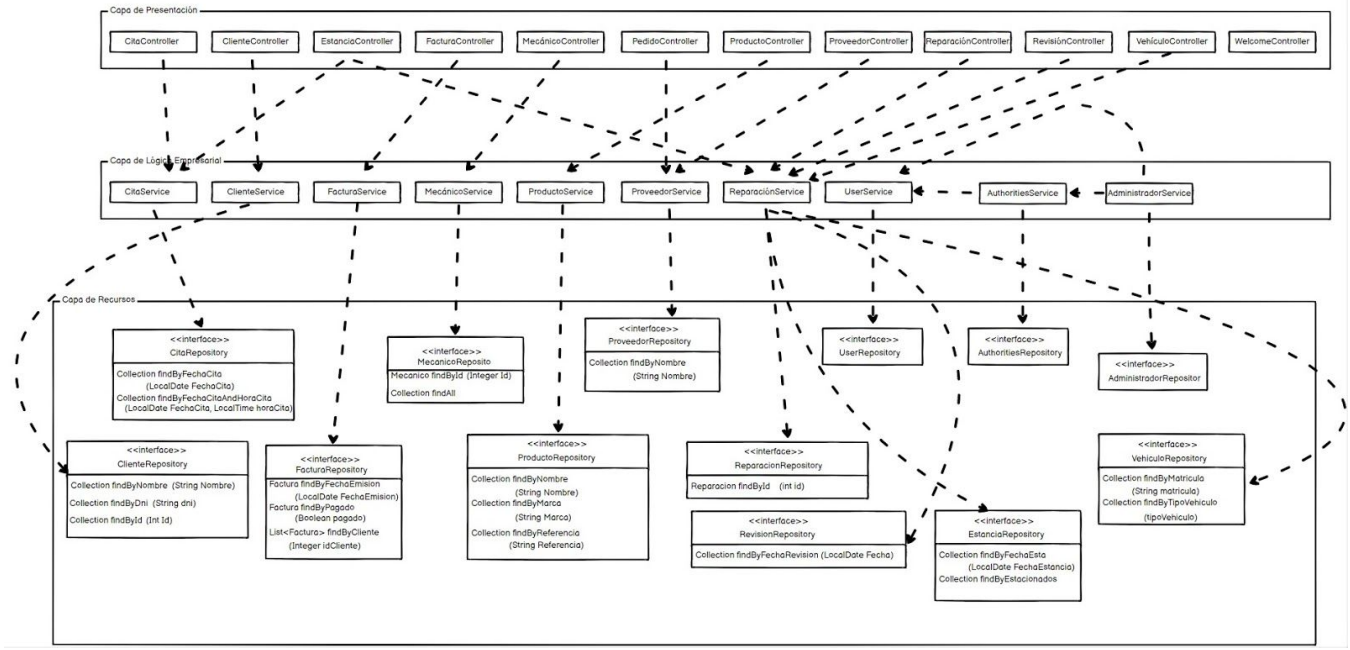


Diagrama de Capas (incluyendo Controladores, Servicios y Repositorios)



Patrones de diseño y arquitectónicos aplicados

Patrón: Modelo-Vista-Controlador

Tipo: Arquitectónico | de Diseño

Contexto de Aplicación

Nuestra aplicación tiene tres capas, la primera es la capa de recursos, los modelos, donde creamos las clases con los distintos atributos y métodos, necesarios para cambiar la información de la base de datos, estos son los paquetes: service, model y repository.

Por otro lado, nosotros tenemos la capa de vista, que son los archivos jsp, encargados de generar la interfaz donde los usuarios de la página podrán interactuar con esta. Se encuentra en la carpeta “WEB-INF”.

Por último están los controladores, que son los que realizan los métodos implementados en la capa de recursos. Estos se encuentran en el paquete web.

Clases o paquetes creados

Hay clases de modelos, en el paquete model, también hay clases service y repository y controladores, en el paquete web, tenemos paquetes para las pruebas de funcionamiento y un paquete para la base de datos y por último tenemos las vistas en la carpeta “WEB-INF”.

Ventajas alcanzadas al aplicar el patrón

Las ventajas de este patrón son: que al ser de capas, facilita el manejo de los errores, además tiene una estructura más fácil de controlar y tiene separados los datos de la representación visual.

Decisiones de diseño

Decisión 1: Agrupación de clases Repository

Descripción del problema:

A la hora de crear los Services nos hemos encontrado ante la casuística de crear una clase Service para cada clase Repository o crear una clase Service que contenga varias clases Repository.

Alternativas de solución evaluadas:

Alternativa 1.a: Crear una clase Service por cada clase Repository.

Ventajas:

- Más legible de cara a alguien que no ha seguido el desarrollo del proyecto.

Inconvenientes:

- Puede crear problemas de dependencia innecesarios.
- Puede ralentizar el arranque inicial de la aplicación.

Alternativa 1.b: Crear una clase Service por varias clases Repository.

Ventajas:

- Eliminamos problemas de dependencias ya que solo importamos una única clase.

Inconvenientes:

- Es más complicado encontrar errores ya que acumulamos métodos.

Justificación de la solución adoptada

Como hemos decidido evitar problemas de altas dependencias para esquivar errores hemos seleccionado la alternativa 1.b.