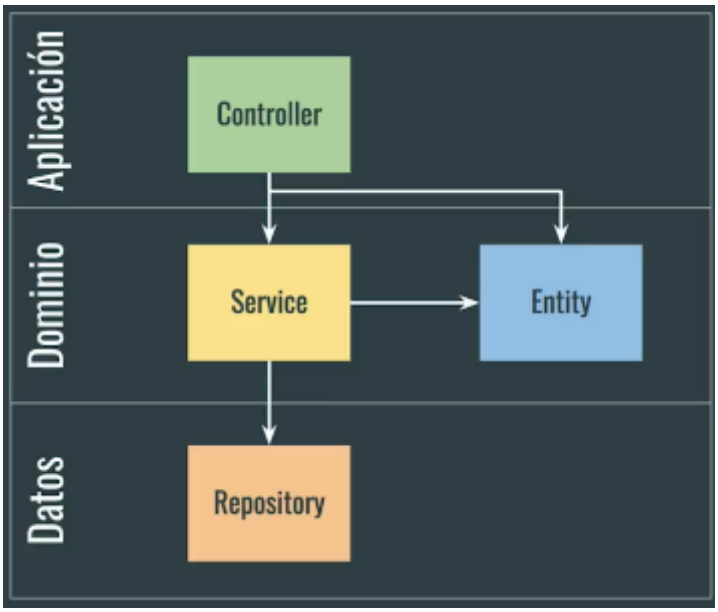
ÍNDICE

Arquitectura del software

Patrón de arquitectura

Esquema general del patrón



- Capa de aplicación:

Se corresponde con la interfaz del sistema, y sirve para la comunicación las peticiones del cliente, y Spring los identifican con “@RestController”.

Estos controllers, reciben las peticiones HTTP que quiere realizar el cliente, estas peticiones se pasarán desde la interfaz gráfica, que en este caso será con JavaFX.

En esta capa, se delega el trabajo a la capa de dominio, ya que es esta la que tiene la lógica del negocio.

- Capa de Dominio:

En esta capa se lleva a cabo toda la lógica del negocio. En esta capa hay dos elementos importantes:

Los Service.

Spring identifica estas clases con la anotación “@Service”, y es en estas donde vive toda la lógica de nuestra aplicación, como la venta de un vehículo, o la

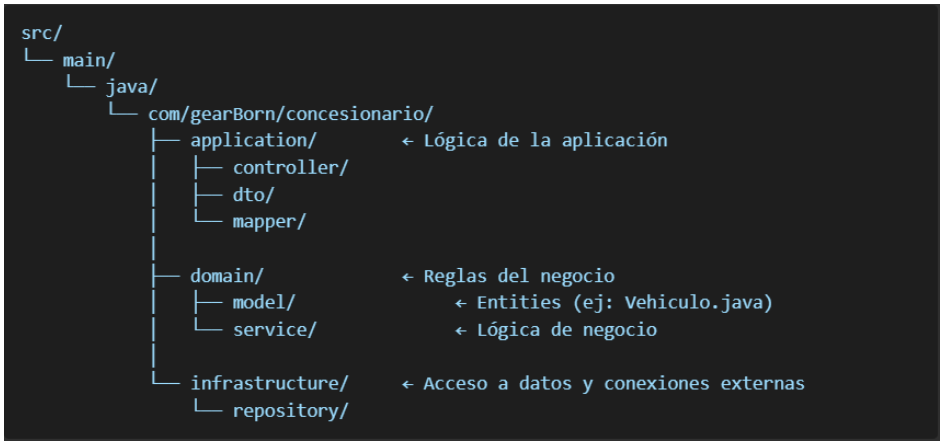
validación de usuarios, por ejemplo. Estas clases se comunican directamente con los repositories.

Las Entities.

Estas son las clases de nuestro proyecto, que se mapearán a la base de datos como las tablas donde se guardarán los datos de la aplicación.
Spring identifica estas clases gracias al uso de “@Entity” que es la anotación que le indica a hibernate que se tiene que mapear como una tabla de la base de datos a la que se conecta la aplicación.

- Capa de datos
  En esta capa se lleva a cabo la persistencia de los datos. Es la capa que trabaja directamente con nuestra base de datos.
  Aquí, nos encontramos a los repositories, que son las interfaces de nuestro código que se encargan de la comunicación real de la base de datos.
  En Spring, los repositories, extienden de las clases JpaRepository, CrudRepository o ListCrudRepository. Estas 3 clases tienen métodos predefinidos para realizar consultas básicas a la base de datos. Para indicarle a Spring cual es un repository o no, tenemos que usar “@Repository”

División de responsabilidades



Proyecto 1. BackEnd

El primer proyecto corresponde al backend del sistema, desarrollado con Spring Boot, cuya responsabilidad principal es gestionar la lógica del negocio, el acceso a la base de datos y la exposición de una API REST para que otros sistemas, como el frontend, puedan interactuar con la aplicación.

Este proyecto implementa una arquitectura por capas que separa claramente las responsabilidades entre controladores, servicios, entidades del dominio y repositorios. Además, se encarga de validar la entrada de datos, gestionar la autenticación y autorización de usuarios mediante roles, y realizar todas las operaciones relacionadas con la persistencia de la información.

El backend está diseñado para ser desacoplado, seguro y escalable, permitiendo su despliegue en la nube y su uso desde diferentes interfaces de cliente.

Proyecto 2. FontEnd

El segundo proyecto es la aplicación cliente desarrollada en JavaFX, encargada de proporcionar una interfaz gráfica intuitiva para los usuarios del sistema.

Su principal responsabilidad es permitir la interacción del usuario con las funcionalidades expuestas por la API REST del backend.

A través de formularios, tablas, botones y vistas dinámicas, esta aplicación captura la información ingresada por el usuario, la transforma en solicitudes HTTP y se comunica con el backend utilizando objetos DTO.

Además, interpreta las respuestas recibidas para mostrarlas en pantalla de forma clara y funcional.

Este proyecto se centra exclusivamente en la presentación y experiencia de usuario, manteniendo la lógica de negocio en el backend y asegurando una clara separación de responsabilidades entre ambas aplicaciones.

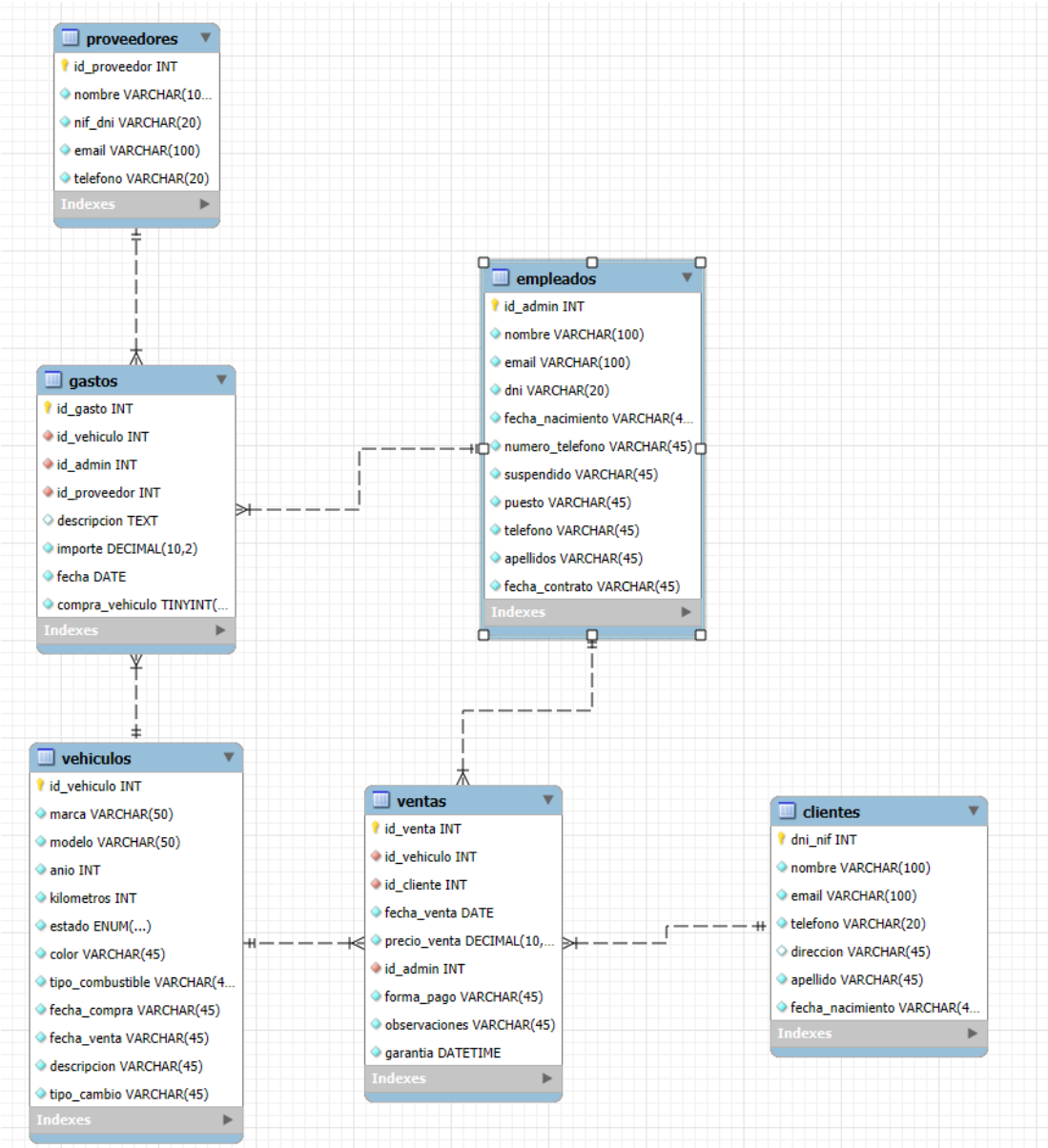
¿Como funciona el flujo?

Cuando se realiza una solicitud, como por ejemplo la creación de un nuevo vehículo, el flujo de datos comienza en el controlador. El Controller recibe un DTO enviado desde el frontend, el cual contiene la información estructurada que representa los datos necesarios para la operación. Este DTO no es utilizado directamente por la lógica de negocio, sino que pasa por un Mapper, cuya responsabilidad es transformar ese DTO en una entidad del dominio (una Entity).

Una vez convertida, la Entity representa el modelo real del negocio y es procesada por el Service, donde se aplica la lógica específica correspondiente (como validaciones, cálculos, o reglas de negocio). Finalmente, el Service delega al Repository la tarea de persistir esa entidad en la base de datos o de realizar la operación de acceso necesaria.

En el caso del flujo de respuesta, por ejemplo al listar vehículos, el proceso comienza desde el Repository, que accede a la base de datos para recuperar las entidades solicitadas. Luego, esas entidades pueden ser manipuladas o filtradas en el Service, según lo requiera la lógica de negocio. Posteriormente, las entidades se transforman nuevamente mediante el Mapper, esta vez desde Entity hacia DTO, para preparar los datos de salida. El DTO resultante es finalmente devuelto por el Controller al frontend, completando así el ciclo de comunicación de manera clara y estructurada.

Base de datos



Explicacion de la base de datos:

El modelo de base de datos diseñado representa el funcionamiento de una empresa dedicada a la compra y venta de vehículos, con funciones administrativas para gestionar clientes, empleados, proveedores, ventas y gastos asociados a cada operación.

La estructura se compone de las siguientes tablas principales:

1. Proveedores

Esta tabla almacena la información de los proveedores externos que brindan servicios o productos a la empresa, tales como talleres, repuesteros o entidades financieras. Se registran datos como el nombre, NIF/DNI, correo electrónico y teléfono de contacto.

2. Clientes

Contiene los datos personales de los clientes que realizan compras en la empresa. Se incluyen campos como el nombre, apellidos, NIF/DNI, dirección, correo electrónico, teléfono y fecha de nacimiento, lo que permite mantener un historial completo y personalizado para cada comprador.

3. Vehículos

Registra todos los vehículos que ingresan y se gestionan dentro del sistema. Incluye información como marca, modelo, año, kilómetros, color, tipo de combustible, estado (nuevo, usado, etc.), y detalles relacionados con la venta, como fecha, descripción y tipo de cambio.

4. Empleados

Esta tabla representa a los empleados o administradores que forman parte de la organización. Se almacena su identificación (`id_admin`), nombre, apellidos, DNI, correo electrónico, teléfonos, puesto dentro de la empresa, fecha de nacimiento, fecha de contrato y estado (activo o suspendido). Además, los empleados están involucrados tanto en la gestión de ventas como en el registro de gastos.

5. Gastos

La tabla de gastos permite llevar un control detallado de todos los costos asociados a cada vehículo. Esto incluye gastos por mantenimiento, reparaciones, compras, entre otros. Cada gasto está relacionado con un vehículo específico, un proveedor que lo ejecutó y un empleado que lo autorizó, permitiendo así la trazabilidad de cada operación.

6. Ventas

Registra las transacciones de venta realizadas a los clientes. Se vincula con los vehículos vendidos, los clientes compradores y los empleados responsables de la

operación. También se almacenan datos como la fecha de la venta, el precio final, la forma de pago, observaciones y la duración de la garantía otorgada.

Relaciones

El modelo establece múltiples relaciones entre tablas para garantizar integridad referencial y trazabilidad:

- Cada **venta** está asociada a un **vehículo**, un **cliente** y un **empleado**.
- Cada **gasto** está relacionado con un **vehículo**, un **proveedor** y el **empleado** que lo gestionó.
- Los **empleados** intervienen como actores clave tanto en los gastos como en las ventas.

Casos de uso

1. Registrar un nuevo cliente

Actor: Empleado
Descripción: El empleado carga los datos personales y de contacto del nuevo cliente que va a realizar una compra.

2. Registrar un nuevo vehículo

Actor: Empleado
Descripción: Se ingresan los datos del vehículo que ingresa al sistema para ser vendido, incluyendo características técnicas, color, estado, combustible, etc.

3. Registrar proveedor

Actor: Empleado o Administrador
Descripción: Se registra la información de un nuevo proveedor externo que ofrece productos o servicios relacionados con el mantenimiento o adquisición de vehículos.

4. Registrar gasto asociado a un vehículo

Actor: Empleado
Descripción: Se crea un registro de gasto indicando el proveedor, tipo de gasto, monto, descripción y vehículo relacionado. Se vincula al empleado que lo gestionó.

5. Registrar una venta

Actor: Empleado
Descripción: Se registra la venta de un vehículo a un cliente determinado. Se almacena la fecha, precio, forma de pago, observaciones, garantía y el empleado responsable de la transacción.

6. Consultar historial de ventas de un cliente

Actor: Empleado o Administrador
Descripción: Se accede al historial de vehículos adquiridos por un cliente, junto con fechas y detalles de cada venta.

7. Consultar historial de gastos por vehículo

Actor: Empleado o Administrador
Descripción: Permite visualizar todos los gastos registrados para un vehículo específico, incluyendo proveedor, tipo de gasto y monto.

8. Dar de baja o suspender a un empleado

Actor: Administrador
Descripción: El administrador marca un empleado como “suspendido” para que no pueda seguir realizando operaciones, sin eliminar su historial.

9. Consultar reportes de ventas por empleado

Actor: Administrador
Descripción: Permite generar informes que muestran cuántas ventas ha realizado cada empleado, sus montos totales y fechas.

10. Modificar datos de contacto de un cliente o proveedor

Actor: Empleado
Descripción: Se actualizan datos como email, teléfono o dirección de clientes o proveedores ya registrados.