

PetsAdministrator

Carlos Rábago Torcates

Sergio Martínez Rivera



ÍNDICE

• Introducción -----	1
• Módulos formativos aplicados -----	3
• Herramientas/Lenguajes -----	4
• Componentes del equipo -----	5
• Fases del proyecto -----	6 - 33
Fase de planificación -----	6
Fase de empatizar -----	7 - 9
Fase de definir -----	10 - 11
Fase de idear -----	12 - 13
Fase de prototipar -----	14 - 16
Fase de creación de la base de datos -----	17 - 19
Fase de creación del proyecto -----	20 - 28
Front-end -----	29
Back-end -----	31 - 33
• Conclusiones -----	34

INTRODUCCIÓN

En el marco de nuestra formación académica, nos encontramos en la etapa final de nuestros dos años de estudio, donde hemos adquirido conocimientos teóricos y prácticos en el campo de desarrollo de aplicaciones web. Como parte de este proceso de aprendizaje, es fundamental poner en práctica los conocimientos adquiridos en un proyecto real que permita consolidar nuestras habilidades y demostrar nuestra capacidad para enfrentar desafíos tecnológicos.

En línea con este objetivo, hemos decidido desarrollar una aplicación web que aborde una necesidad concreta: la gestión eficiente del inventario en una tienda de ropa para mascotas. La gestión del inventario es un aspecto crítico en cualquier negocio, ya que un control inadecuado puede resultar en pérdidas financieras, falta de disponibilidad de productos o una mala experiencia para los clientes.

Nuestra aplicación web permitirá gestionar su inventario de manera más eficiente y sencilla, optimizando los procesos y mejorando la toma de decisiones relacionadas con el stock. A través de funcionalidades como el registro, modificación y eliminación de productos en stock, así como la capacidad de realizar búsquedas filtrando por categorías o palabras clave, se facilitará el seguimiento y control de los productos disponibles.

Además, hemos considerado la importancia de garantizar la seguridad y confidencialidad de la información. Por esta razón, la aplicación contará con roles de usuario que tendrán diferentes permisos de acceso. Esto permitirá establecer niveles de autorización y asegurar que solo el personal autorizado tenga acceso a las áreas correspondientes de la aplicación. De esta manera, se protegerá la integridad de los datos y se minimizará el riesgo de accesos no autorizados.

Un aspecto destacado de nuestro proyecto es que la aplicación web será implementada dentro un sitio web que crearemos con el fin de demostrar el funcionamiento correcto de la aplicación. Esto proporcionará una integración perfecta entre la gestión del inventario y la plataforma de venta, ofreciendo una experiencia de usuario consistente y fluida tanto para los empleados encargados de la administración del inventario como para los clientes que acceden al catálogo de productos.

Con este proyecto, nuestro objetivo es poner en práctica los conocimientos adquiridos durante nuestros dos años de formación. Para lograrlo, hemos decidido desarrollar una aplicación web que permitirá gestionar el inventario de una tienda de ropa para mascotas de manera eficiente y sencilla. La aplicación permitirá registrar, modificar y eliminar los productos en stock, así como realizar búsquedas filtrando por categorías o palabras específicas. Además, la aplicación contará con roles de usuario que tendrán diferentes permisos para acceder a diferentes áreas de la aplicación, de esta manera, se garantizará la seguridad y confidencialidad de la información, evitando el acceso no autorizado a la misma. Todo esto estará implementado dentro de la propia web de la tienda.

Objetivos:

- Desarrollar una aplicación web: El objetivo principal es crear una aplicación web Implementación de roles de usuario: Se establecerán roles de usuario con diferentes permisos de acceso. Esto garantizará la seguridad y confidencialidad de la información almacenada en la aplicación, evitando el acceso no autorizado. Por ejemplo, los empleados podrán tener diferentes niveles de acceso según sus responsabilidades, lo que contribuirá a una gestión más segura y controlada del inventario.
- Integración en un sitio web: La aplicación web se implementará dentro del sitio web que crearemos para una tienda de ropa para mascotas. El objetivo es lograr una integración perfecta y una experiencia de usuario consistente. Esto permitirá a los clientes acceder al catálogo de productos y a la tienda en línea de manera intuitiva, para mejorar la interacción y fomentar las ventas.
- Funcional y de calidad que permita gestionar el inventario de una tienda de ropa para mascotas. Esto implicará el diseño, desarrollo e implementación de las funcionalidades necesarias para llevar a cabo dicha gestión.
- Gestión eficiente del inventario: Se busca mejorar la eficiencia en el control y seguimiento de los productos en stock. La aplicación permitirá registrar, modificar y eliminar productos de manera sencilla y precisa, facilitando la actualización del inventario y evitando inconsistencias o errores en la gestión de los productos.
- Búsqueda avanzada de productos: La aplicación ofrecerá la posibilidad de realizar búsquedas filtrando por categorías o palabras clave específicas. Esto permitirá a los usuarios localizar rápidamente los productos deseados, agilizando el proceso de búsqueda y optimizando el tiempo empleado en la gestión del inventario.

MÓDULOS FORMATIVOS APLICADOS

- Base de Datos: Uso de base de datos relacional.
- Programación: Utilizamos Java como lenguaje de programación principal al ser en el que más hemos profundizado durante curso.
- Diseño de Interfaces Web: Una de las primeras fases del proyecto se basó en el uso de la herramienta de Figma para poder realizar el diseño UX, los benchmarking etc.
- Lenguajes de marcas: Para la parte visual de la web usaremos los conocimientos adquiridos en HTML y CSS.
- Desarrollo de Aplicaciones Web en Entorno Cliente: Uso de JavaScript para la parte dinámica de la web.
- Desarrollo de Aplicaciones Web en Entorno Servidor: Uso de bases de datos y desarrollo con Java y Spring.
- Inglés Técnico: Los conocimientos adquiridos facilitaron durante el proyecto la lectura de documentos técnicos.

HERRAMIENTAS/LENGUAJES

IDE:

- ECLIPSE 

BASE DE DATOS:

- MYSQL
- MYSQL WORKBENCH
- SQL



DISEÑO:

- FIGMA 

LENGUAJES:

- PARA FRONT JAVASCRIPT 
- PARA BACK JAVA 

LENGUAJES DE MARCAS:

- HTML5 
- CSS 

OTROS:

- SPRING 
- SERVIDOR APACHE TOMCAT 
- GIT 
- GITHUB 

COMPONENTES DEL EQUIPO



Sergio Martínez Rivera



Carlos Rábago Torcates

Estudiantes del ciclo formativo de grado superior de Desarrollo de aplicaciones web en el curso académico 2021 - 2023.

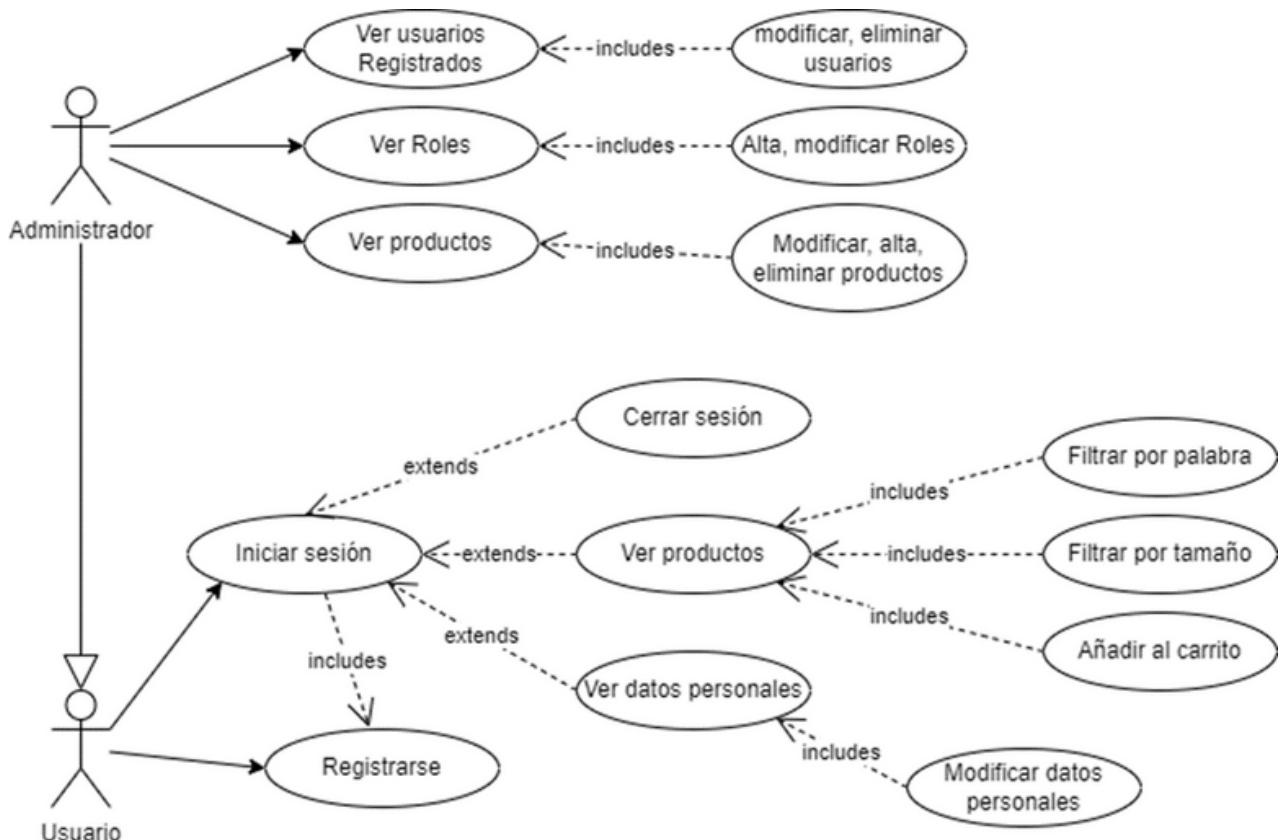
Iniciamos el proyecto con unas sesiones para sentar las bases del proyecto y discutir como íbamos a organizarnos a lo largo del mismo, la temática, tecnologías a utilizar etc.

Utilizamos GITHUB como repositorio para ir subiendo los cambios realizados por cada integrante.

FASES DEL PROYECTO

Fase de planificación

Una vez que determinamos el proyecto, nos embarcamos en la investigación de los recursos y tecnologías requeridos, así como en la comprensión de la amplitud involucrada en el desarrollo del mismo. En la primera versión de la aplicación pueden acceder dos tipos de usuarios: el administrador y el usuario (cliente). En el siguiente diagrama se presentan los distintos escenarios considerados para cada tipo de usuario.



Un usuario puede realizar las siguientes funciones:

- Iniciar sesión, para lo que necesita estar previamente registrado.
- Ver los productos que hay disponibles, filtrar por tamaños, filtrar por palabras, añadir al carrito.
- Ver sus datos personales y modificarlos.

Un administrador puede realizar todas las funciones que realiza un usuario y además puede:

- Ver los usuarios registrados, modificarlos y eliminarlos.
- Ver roles de usuario, alta y modificarlos.
- Modificar, alta y eliminar productos.

Fase de empatizar

Empezaremos con la fase de empatizar para conocer el producto y a nuestro usuario. Llevamos a cabo una investigación UX para comprender las necesidades de los usuarios y cómo satisfacerlas a través de nuestro producto. Esta fase del proyecto a su vez la dividimos en sub-fases.

- Investigación documental del producto y temática.
- Investigación de la competencia.
- Investigación de usuario y su contexto.

Investigación documental del producto y temática

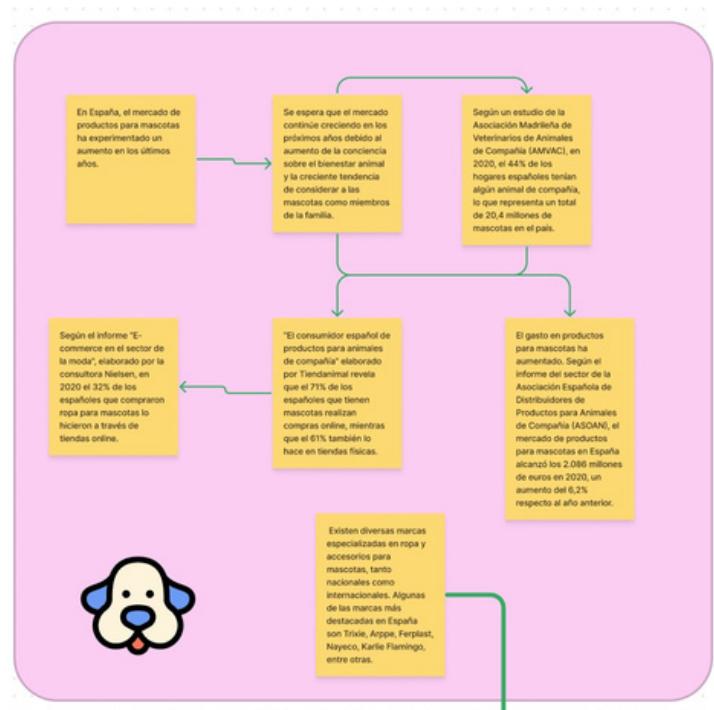
En este punto se investiga el momento en el que se encuentra el mercado de la venta online de productos para mascotas. Se centrará la atención principalmente en el mercado de venta de ropa de mascotas y no tanto en los productos en general como pueden ser correas, arneses o pienso ya que es en esencia la temática de nuestra web.

Por otro lado se hace una investigación general de la situación actual de los usuarios que tienen perros en nuestro país y como se comportan.

Toda esta información se realiza de a través de internet desde diversas encuestas y artículos encontrados sobre el tema en cuestión.

Por otro lado la información encontrada acerca del mundo de la gestión de inventarios es mucho más reducida y se basa principalmente en aquello que ofrecen las distintas plataformas de la competencia en sus respectivas webs. Por ello esto se investigará un poco mas en profundidad en la parte del benchmarking.

De este modo se llega a alas siguientes conclusiones en este apartado.



Investigación de la competencia

Se eligen como referencia las webs de venta de ropa y accesorios para mascotas más reconocidas según nuevamente una investigación a través de opiniones de usuario en foros, búsquedas en internet y múltiples artículos.

Las web sobre las que finalmente se centra esta investigación son por tanto finalmente, Ferplast y Arppe.

En este apartado se enfoca la investigación sobre todo en el tema visual general de la web así como la colocación de los elementos y la usabilidad de la misma.

Por tanto se ha logrado llegar a las siguientes conclusiones:

The screenshot shows the Ferplast website with several user interface elements highlighted by yellow sticky notes:

- Header:** Visually the page looks like it could be more modern, lacks a touch of originality.
- Product Categories:** Well categorized and easy access through the catalog.
- Product Listings:** Shows products from different species, including dogs, cats, birds, fish, and reptiles.
- Filtering:** Multiple filtering options available.
- Payment Options:** Multiple payment methods available.
- Product Detail:** Shows availability, financing options, and fast shipping.

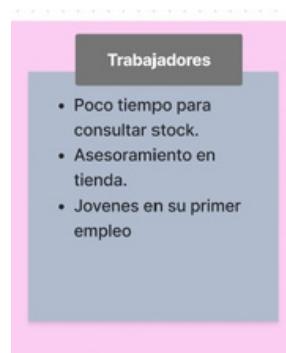
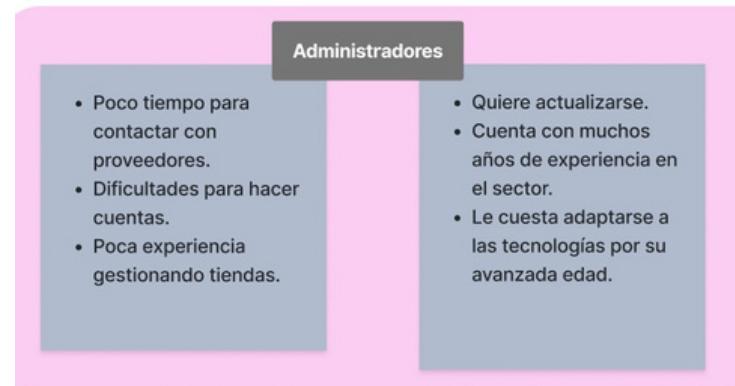
The screenshot shows the Arppe website with several user interface elements highlighted by yellow sticky notes:

- Header:** Good variety of products.
- Category Pages:** Simple and intuitive design.
- Product Details:** Description of the product is not very informative.
- Product Detail:** Product details do not provide much useful information for the user.
- Product View:** Not very useful for finding products that fit the user's needs.
- Product Selection:** Allows users to filter products by color, size, and brand.

Investigación de usuarios

Contando con la experiencia de las investigaciones previas y con la búsqueda de información al igual que en las anteriores, definimos de manera general 3 tipos de usuarios en los que se centra el proyecto, los **administradores**, **trabajadores** y los **clientes**.

Definidos los perfiles se obtienen las siguientes características comunes para cada uno de ellos:



Fase de definir

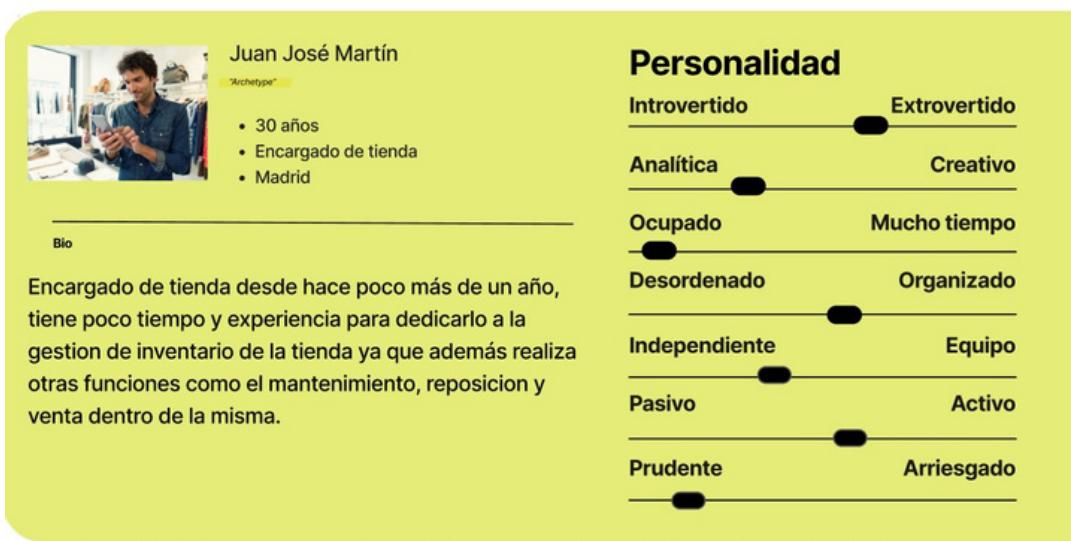
Una vez definida la fase de empatizar, le toca el turno a la de definir, donde se pondrá el punto de mira en las necesidades de los usuarios.

Una vez más este proceso se dividirá en dos sub-fases.

- User persona
- POV

User persona

Partiendo de los perfiles definidos en la investigación de usuario, se procede a la creación de un arquetipo. El arquetipo usado será un Administrador de una tienda de mascotas, que tiene poca experiencia en su puesto y necesita ayuda, por tanto el arquetipo queda de la siguiente manera:

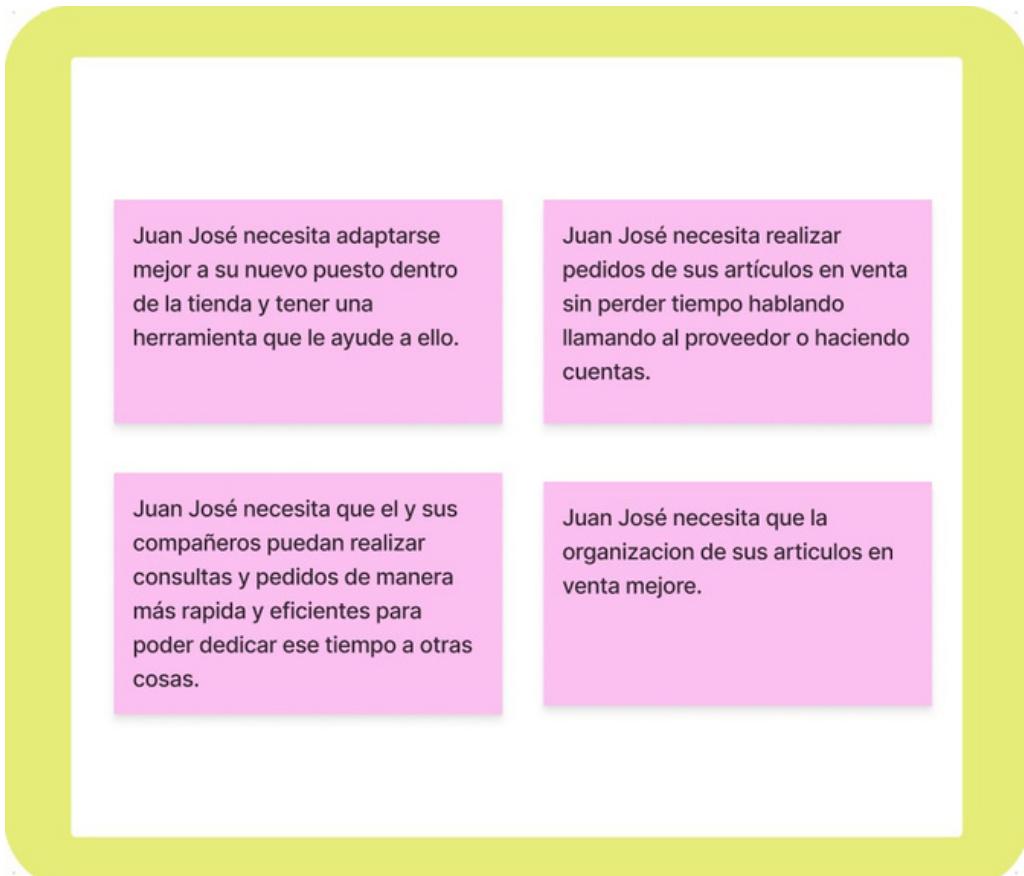


Sobre este arquetipo se plantean sus frustraciones, motivaciones y objetivos.



POV

Partiendo de las frustraciones, motivaciones y objetivos del arquetipo, se realiza el point of view, donde se intentará plantear que necesidades tiene el sujeto. Una vez definido el pov será más fácil buscar soluciones a través de ideas y siempre priorizando por importancia.



Fase de idear

Generaremos las ideas para dar solución a esas necesidades o problemas de nuestros usuarios obtenidos en la fase anterior. Para ello primero reformularemos los POV en How Might We? y más adelante, se realiza una sesión de barinstorming para buscar de entre todas las mejores soluciones de entre todas la propuestas.

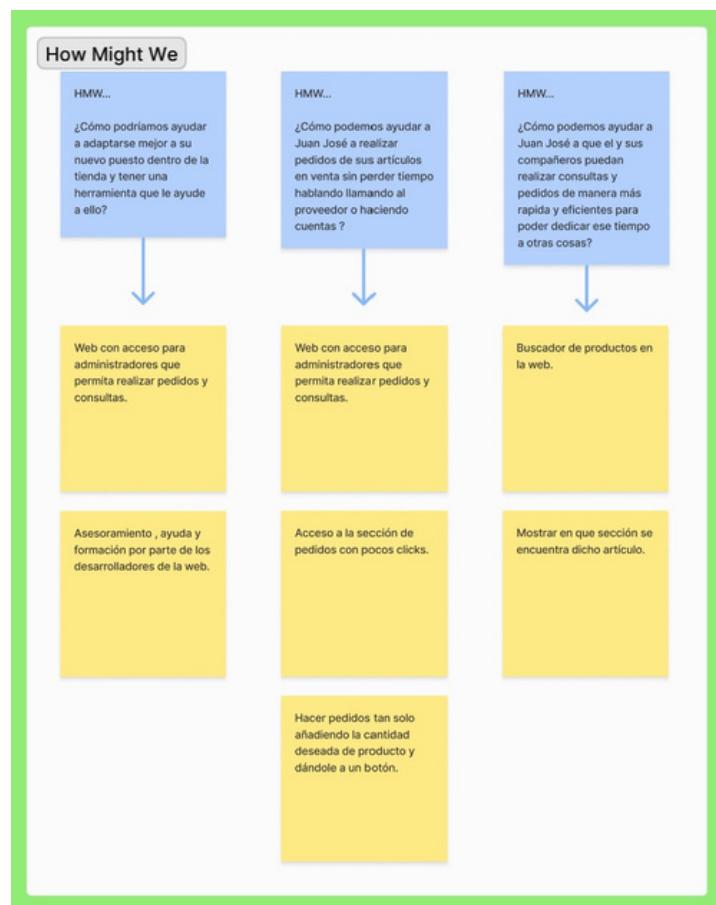
How Might We

Es una técnica utilizada en el diseño de pensamiento (design thinking) para formular preguntas que ayudan a abordar desafíos o problemas complejos. Se utiliza para plantear un enunciado que estimule la generación de ideas creativas y soluciones innovadoras.

El formato "how might we" se compone de tres partes:

1. "How" (cómo): Implica que se está buscando una solución o un enfoque para abordar una situación o un problema.
2. "Might" (podría): Sugiere que la solución o el enfoque propuesto aún no ha sido definido, y se anima a explorar múltiples posibilidades y opciones.
3. "We" (nosotros): Indica que el desafío se abordará de manera colaborativa, invitando a un equipo o grupo de personas a participar en la búsqueda de soluciones.

Este proceso ayuda a definir el problema de manera más clara y a generar ideas que conduzcan a soluciones prácticas y efectivas.

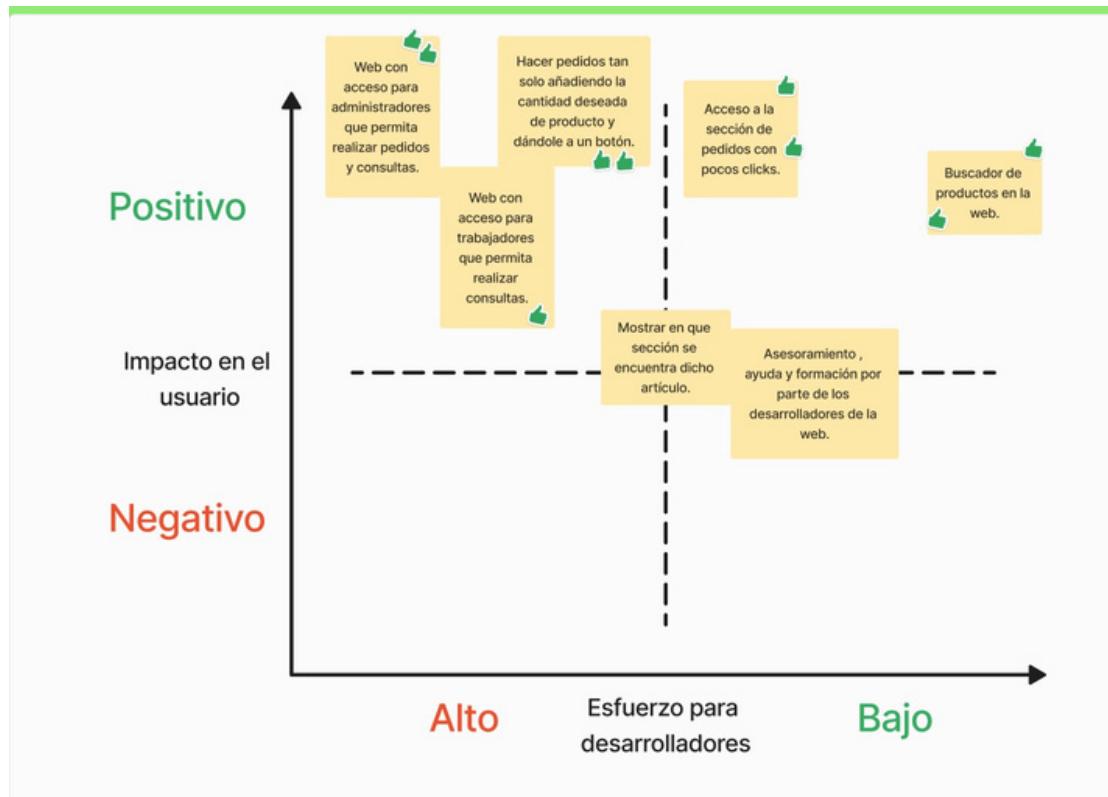


Brainstorming

Ambos integrantes del equipo generaron ideas en una sesión de brainstorming para obtener posibles soluciones frente al how might we.

Planteadas las soluciones se estimo la calidad de las mismas de cara a los usuarios y la dificultad y trabajo que podía llegar a suponer para el proceso del desarrollo de la web.

Por último se hizo un sistema de votación para saber a cuales dar prioridad.



Fase de prototipar

En esta fase se idea la parte estética y funcional de la web. Para ello se realiza un User Story Mapping donde se detallan las funcionalidades que puede realizar el usuario y los pasos que se siguen en esas acciones de manera secuencial.

Después se organizan por MVP (Producto mínimo viable) para marcarnos una ruta de que ir desarrollando y en qué orden hacerlo según la facilidad que suponga y la importancia.



Todas estas fases de investigación y prototipado, fueron realizadas con la herramienta FIGMA, vista en la asignatura de diseño de interfaces.

Puede consultarse el proyecto a través del siguiente enlace.

<https://www.figma.com/file/KeLzEu751Bn6FDS5GvqMFs/Pets-Administrator?type=whiteboard&t=uEjzeiUbDi9ZrLvH-1>

El proyecto se continua en FIGMA para realizar las siguientes fases:

- Wireframes
- Guía de estilos

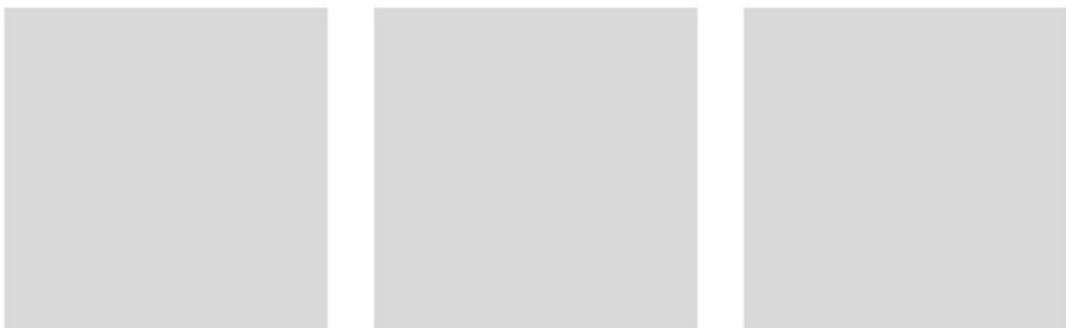
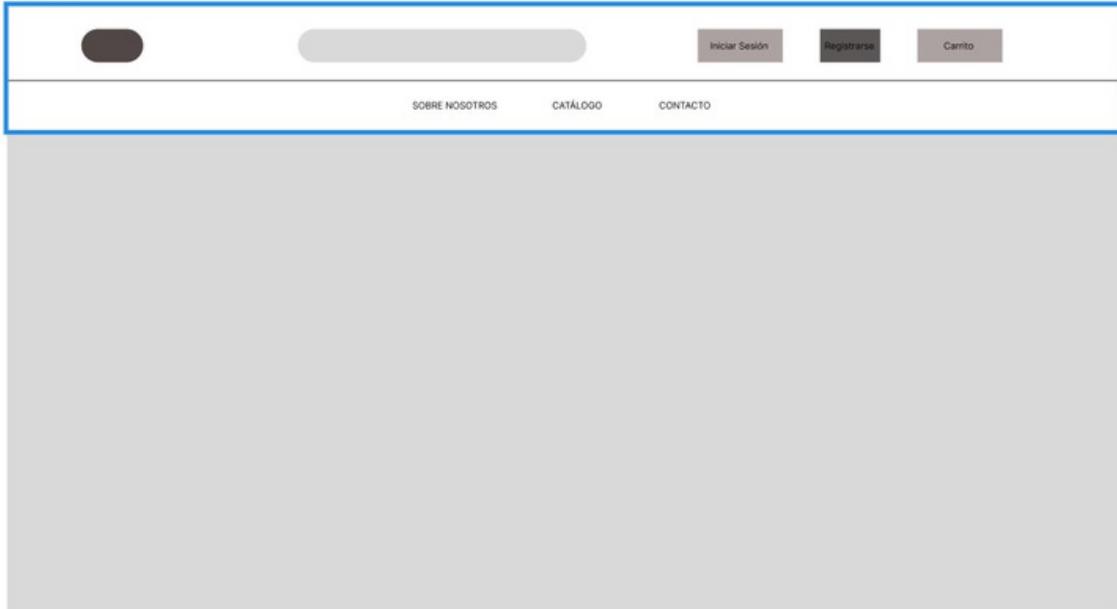
Wireframes

Se hace una representación visual básica y estructural de una interfaz de usuario. Tan solo muestra de manera conceptual el esquema y la disposición de los elementos que en un primer momento deseamos que tenga la web.

El wireframe entero puede consultarse en el siguiente enlace y se adjuntan algunas de las pantallas más relevantes en este documento.

<https://www.figma.com/file/SacAszwmN2sT5h5EyEBjr/Untitled?type=design&node-id=5-2&t=uEjzeiUbDi9ZrLvH-0>

HOME



PÁGINAS

[Sobre Nosotros](#)
[Productos](#)
[Contacto](#)

CATÁLOGO

[Pequeño](#)
[Mediano](#)
[Grande](#)

© 2023 PetsAdministrator, Inc. All rights reserved.

LISTA DE PRODUCTOS

The image shows a wireframe or placeholder for a product list page. At the top, there's a header with a dark grey rounded rectangle on the left, a light grey rounded rectangle in the center, and three small rectangular buttons on the right labeled "Iniciar Sesión", "Registrarse", and "Carrito". Below the header is a horizontal line with three menu items: "SOBRE NOSOTROS", "CATÁLOGO", and "CONTACTO". The main title "LISTA DE PRODUCTOS" is centered above a grid of eight product cards. Each card has a small thumbnail image (either solid grey or white), a light grey background, and a dark grey action button on the right. At the bottom is a dark grey footer bar containing a large circular icon on the left, and two columns of text on the right: "PÁGINAS" with links to "Sobre Nosotros", "Productos", and "Contacto", and "CATÁLOGO" with links to "Pequeño", "Mediano", and "Grande". A copyright notice "© 2023 PetsAdministrator, Inc. All rights reserved." is at the very bottom.

LISTA DE PRODUCTOS

SOBRE NOSOTROS CATÁLOGO CONTACTO

LISTA DE PRODUCTOS

PÁGINAS

Sobre Nosotros
Productos
Contacto

CATÁLOGO

Pequeño
Mediano
Grande

© 2023 PetsAdministrator, Inc. All rights reserved.

Fase de la creación de la base de datos

Acorde a lo establecido en el UML y el design thinking, era momento de empezar a darle forma al proyecto. Empezamos con la creación de la base de datos, para ello creamos un nuevo script de sql y comenzamos a realizar la creación de la misma y sus diferentes tablas y columnas.

La estructura del script fue la siguiente:

- Creación de la base de datos.

```
DROP DATABASE IF EXISTS petsadministrator;
CREATE DATABASE petsadministrator CHARACTER SET utf8mb4;
USE petsadministrator;
```

Drop database if exists se utiliza para borrar la base de datos en caso de que ya exista.

Create database es las sentencias que crean la base de datos.

Character set marca el rango de caracteres que utilizaremos. En este caso utf8mb4 es una versión extendida del UTF-8 que permite almacenar caracteres especiales.

Use sirve para seleccionar que queremos trabajar con esta base de datos.

- Creación de tablas y columnas

Principalmente tendremos la tabla de **roles** que servirá para definir que tipo de usuario está registrado y los accesos a los que dispondrá dentro de la web. Por otro lado la tabla de **usuarios** que almacenará todos los datos referentes al mismo.

```
CREATE TABLE roles (
    id_rol INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL
);

CREATE TABLE usuarios (
    id_usuario INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(20) NOT NULL,
    apellido1 VARCHAR(20) NOT NULL,
    apellido2 VARCHAR(20),
    fecha_nacimiento DATE NOT NULL,
    email VARCHAR(50) NOT NULL UNIQUE,
    username VARCHAR(200) NOT NULL UNIQUE,
    password VARCHAR(200) NOT NULL,
    enabled INT(1) NOT NULL DEFAULT '1',
    fecha_registro DATE NOT NULL,
    id_rol INT NOT NULL,
    foreign key(id_rol) references roles(id_rol)
);
```

La sentencia **Create table** es la sentencia usada para crear todas las tablas. Dentro de cada tabla se introducen todos las columnas de la tabla y sus tipos de datos. Por ejemplo la tabla de **roles** cuenta con un id_rol que servirá para identificar de forma inequívoca cada rol, será de tipo numérico (INT) y por cada vez que se cree un nuevo rol, a través de auto_increment automáticamente se le asignará un valor de manera consecutiva, es decir que si tenemos un rol existente con id_rol = 1 el siguiente rol que creemos, automáticamente se le asignara el id_rol= 2. Además el id_rol servirá como clave primaria de la tabla.

Todas las tablas del proyecto cuentan con una columna id que será la Primary Key en cada una de ellas.

Por último esta tabla cuenta con otra columna llamada nombre de tipo varchar que acepta 100 caracteres y que almacena un valor alfanumérico que se usará para asignarle un nombre al rol y además a la hora de crearse uno nuevo obligatoriamente debe contener un valor ya que está como NOT NULL, es decir que no puede ser su valor nulo.

Las tablas se relacionaran entre sí a través de las primary key y las foreing key, de este modo en el ejemplo anterior puede observarse que tanto la tabla de **roles** como la tabla de **usuarios** cuentan con una columna id que funciona como la primary key, sin embargo la tabla de usuarios cuenta además con una columna llamada foreingn key que tiene el valor de id_rol y hace referencia a la tabla roles. De este modo estamos relacionando ambas tablas y diciéndole a nuestra base de datos que existe una relación 1 - N , es decir de uno a muchos donde un usuario puede tener 1 único rol y un rol puede contener muchos usuarios.

- Insertar valores

```
INSERT INTO roles VALUES(1, 'Admin');
INSERT INTO roles VALUES(2, 'Cliente');

-- INSERTAR USUARIOS

INSERT INTO usuarios VALUES(1, 'María', 'Pérez', 'Pérez', '1993-04-07', 'mperez@gmail.com', 'mperez', '$2a$10$af5691PQ6lb2Yjvc3u2gAVoCS/q3h5J0jb1vhns8DfT7tM0', 1, '2023-01-07', 2);
INSERT INTO usuarios VALUES(2, 'Juan', 'Gómez', 'López', '1963-03-17', 'jgomez@gmail.com', 'jgomez', '$2a$10$af5691PQ6lb2Yjvc3u2gAVoCS/q3h5J0jb1vhns8DfT7tM0', 1, '2023-01-17', 2);
INSERT INTO usuarios VALUES(3, 'Daniel', 'Ruiz', 'Ortega', '1989-11-28', 'druiz@gmail.com', 'druiz', '$2a$10$af5691PQ6lb2Yjvc3u2gAVoCS/q3h5J0jb1vhns8DfT7tM0', 0, '2023-02-01', 2);
INSERT INTO usuarios VALUES(4, 'Laura', 'Barrio', 'Marín', '1988-09-15', 'lbarrio@gmail.com', 'lbarrio', '$2a$10$af5691PQ6lb2Yjvc3u2gAVoCS/q3h5J0jb1vhns8DfT7tM0', 1, '2023-01-22', 2);
INSERT INTO usuarios VALUES(5, 'Sergio', 'Martínez', 'Rivera', '1991-04-07', 'sergio4884m@gmail.com', 'smartinez', '$2a$10$af5691PQ6lb2Yjvc3u2gAVoCS/q3h5J0jb1vhns8DfT7tM0', 1, '2023-01-01', 1);
INSERT INTO usuarios VALUES(6, 'Carlos', 'Rabago', 'Torcates', '1990-09-15', 'carlosmiguel40@gmail.com', 'crabago', '$2a$10$af5691PQ6lb2Yjvc3u2gAVoCS/q3h5J0jb1vhns8DfT7tM0', 1, '2023-01-22', 2);
```

Para insertar valores se usará la sentencia **INSERT INTO** seguido del nombre de la tabla y **VALUES**. Seguido a ello y entre paréntesis van los valores separados por comas y en el formato y orden correspondiente, es decir, en el ejemplo de la tabla de roles el formato será (1, "Admin") donde 1 hace referencia al id y Admin a la columna nombre y además va entre "" al ser de tipo varchar.

Es importante introducir los valores de las columnas en el mismo orden que se crearon.

- Usuario y privilegios

```
CREATE USER petsadministrator IDENTIFIED BY 'petsadministrator';
grant all privileges on petsadministrator.* to petsadministrator;
FLUSH PRIVILEGES;
```

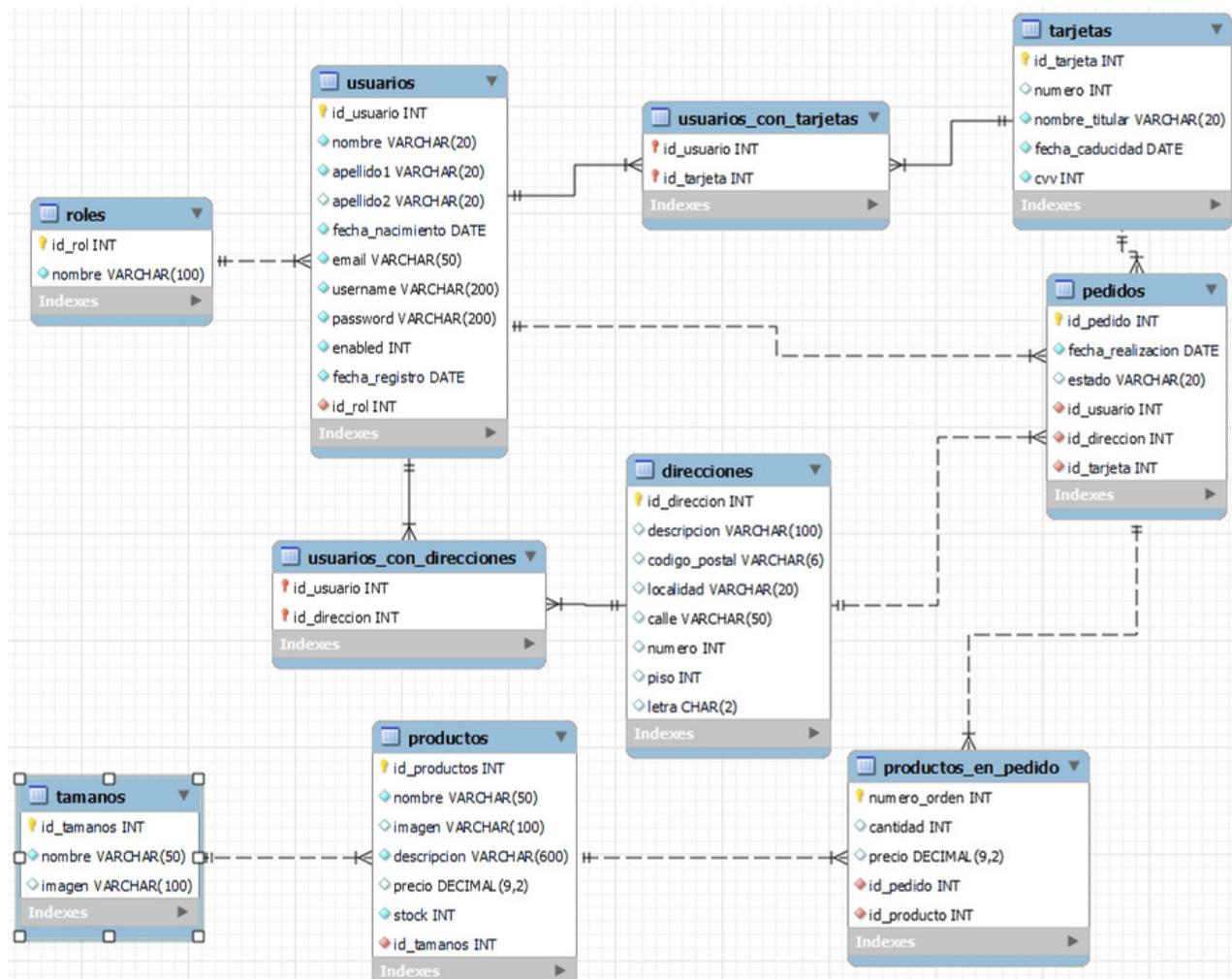
Para más adelante poder conectar con el proyecto la base de datos crearemos un usuario con privilegios.

La primera sentencia "**CREATE USER**" crea un nuevo usuario llamado "petsadministrator" con una contraseña "petsadministrator".

La segunda sentencia "**GRANT ALL PRIVILEGES**" otorga todos los permisos disponibles en la base de datos "petsadministrator" al usuario "petsadministrator". Esto significa que el nuevo usuario tendrá acceso total a todas las tablas y bases de datos en la base de datos "petsadministrator".

La tercera sentencia "**FLUSH PRIVILEGES**" actualiza los permisos de la base de datos para que el nuevo usuario tenga acceso inmediato a las tablas y bases de datos correspondientes.

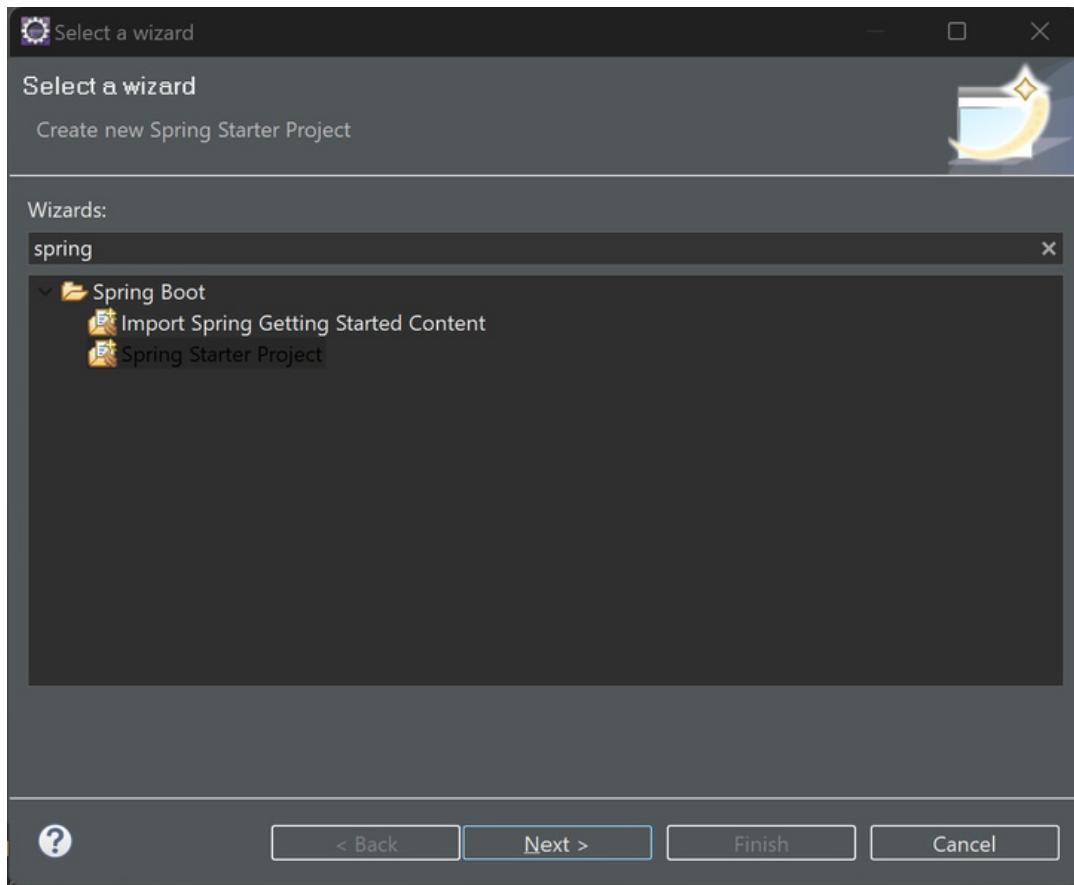
Una vez finalizada la base de datos completa tendrá este aspecto.



Fase de la creación del proyecto en eclipse

Para desarrollar el proyectos se utilizará como entorno de desarrollo eclipse, al ser el más utilizado a lo largo del curso y con el que los integrantes del equipo nos sentimos más familiarizados.

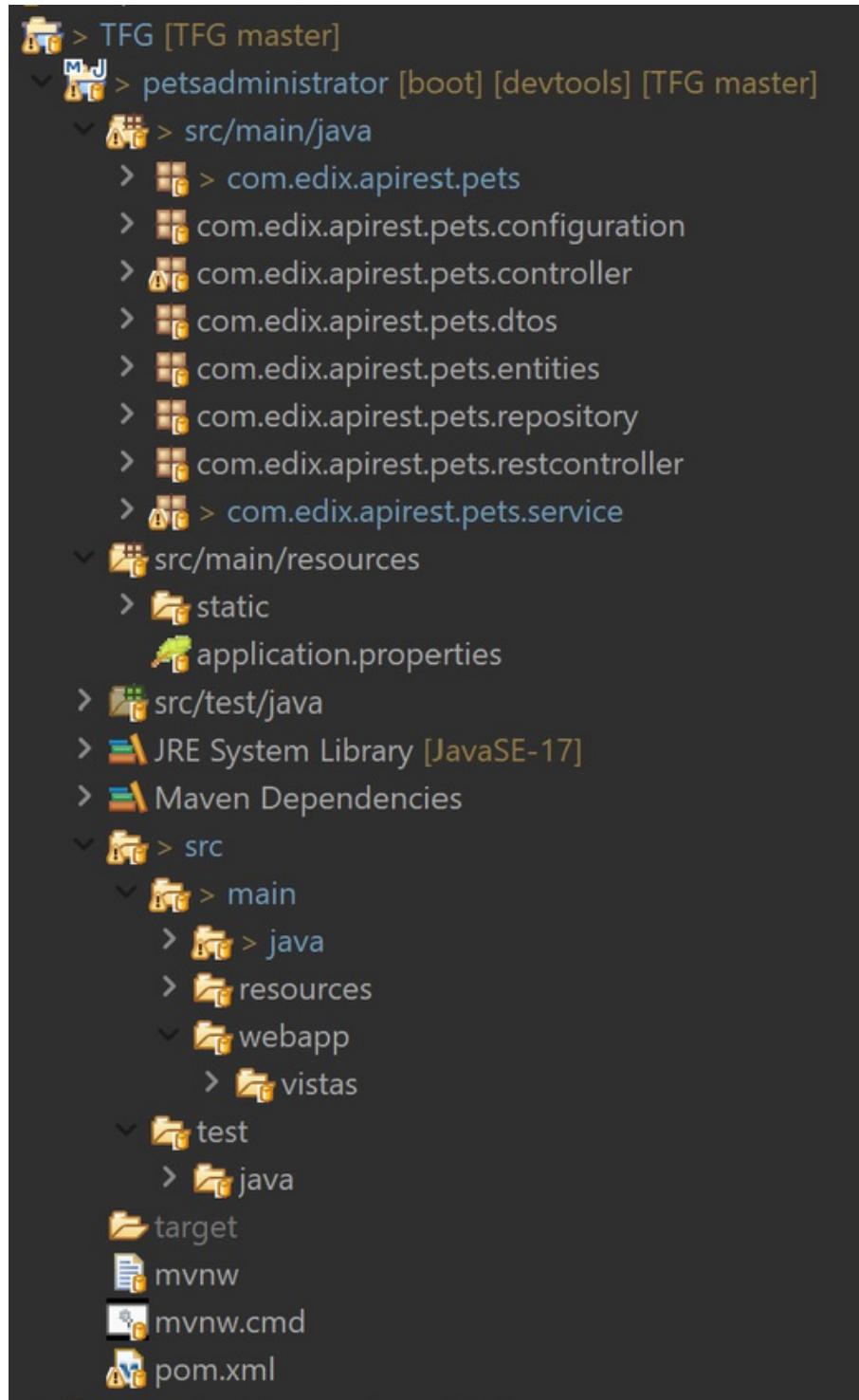
Para la creación del mismo se abre eclipse y selecciona el workspace sobre el que vamos a trabajar. Una vez dentro al ser un proyecto donde utilizaremos **spring** crearemos un nuevo proyecto spring starter project.



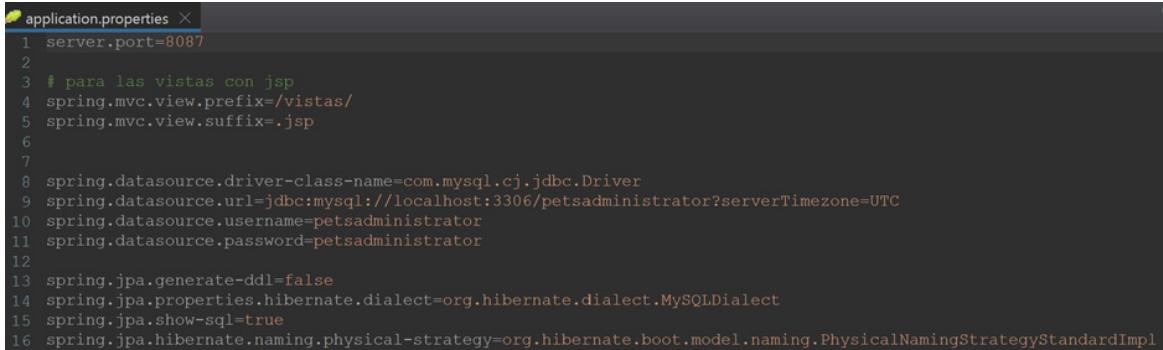
Se le añadirán los siguientes starters:

- MySQL driver: establece la conexión y la comunicación entre una aplicación Spring y una base de datos MySQL. El controlador permite que la aplicación se conecte al servidor de la base de datos MySQL, envíe consultas SQL y reciba los resultados.
- Spring Boot DevTools: Proporciona una serie de características y utilidades que agilizan el ciclo de desarrollo y ayudan a mantener un flujo de trabajo eficiente.
- Spring Web: Este módulo se basa en el patrón de diseño Modelo-Vista-Controlador (MVC) y ofrece una serie de componentes y utilidades para facilitar el desarrollo de aplicaciones web a través de los controladores para manejar las solicitudes web y generar respuestas, ayuda a la lectura de los JSP etc.
- Spring Security: Proporciona las herramientas y funcionalidades necesarias para dotar de seguridad a la web.

Una vez creado el proyecto tendrá el siguiente aspecto. Y a continuación se explica cada apartado.



Application properties



```
application.properties
1 server.port=8087
2
3 # para las vistas con jsp
4 spring.mvc.view.prefix=/vistas/
5 spring.mvc.view.suffix=.jsp
6
7
8 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
9 spring.datasource.url=jdbc:mysql://localhost:3306/petsadministrator?serverTimezone=UTC
10 spring.datasource.username=petsadministrator
11 spring.datasource.password=petsadministrator
12
13 spring.jpa.generate-ddl=false
14 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
15 spring.jpa.show-sql=true
16 spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

Aquí se configura el puerto en el que la aplicación se ejecutará cuando se inicie. En este caso el 8087.

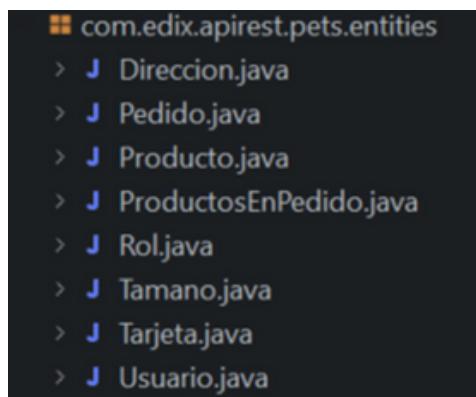
Por otro lado se le indica la ubicación y el formato de las vistas que en este caso serán los jsp dentro de la carpeta vistas.

Se debe especificar también los detalles de la conexión a la base de datos, como la URL, el nombre de usuario y la contraseña.

Entities

Se crea una carpeta dentro de la carpeta src/main/java donde se guardan las clases que corresponden a las tablas de la base de datos.

Las clases son objetos que representan datos que se pueden guardar en la base de datos. Una clase representa una tabla guardada en una base de datos, y cada objeto de una clase representa una fila en la tabla. Cada clase tiene atributos que se relacionan con las columnas de cada tabla, un constructor vacío, los métodos para obtener (getters) y modificar (setters) los atributos, el hashCode y el equals.

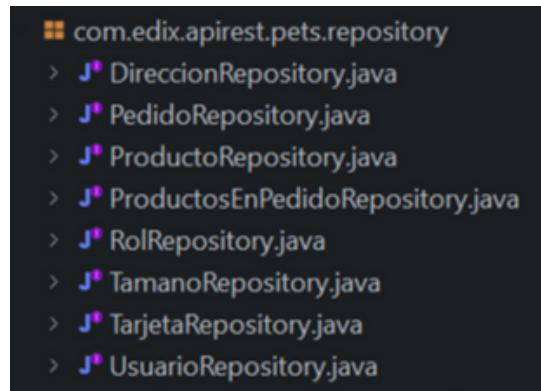


Repositories

Luego se crean los ‘repository’, estos están ubicados en el paquete com.edix.apirest.pets.repository, y desempeñan un papel fundamental al proporcionar interfaces que permiten la conexión y la interacción entre el sistema y la base de datos. Estas interfaces, que extienden las clases proporcionadas por JPA (Java Persistence API), actúan como contratos abstractos que definen los métodos y operaciones necesarios para gestionar la información almacenada en la base de datos.

Las interfaces en el 'repository' ofrecen una capa de abstracción que oculta los detalles de la implementación subyacente y permite a las clases de servicios y controladores trabajar con entidades y objetos de dominio en lugar de preocuparse por las consultas y manipulaciones específicas de la base de datos.

Cada interfaz en 'repository' define métodos que encapsulan operaciones comunes, como buscar, borrar, actualizar o crear registros en la base de datos. Estos métodos se basan en convenciones de nombres y anotaciones proporcionadas por JPA, que se encargan de generar automáticamente las consultas SQL correspondientes para interactuar con la base de datos.



Service

Seguidamente se crean los ‘service’, que están en el paquete com.edix.apirest.pets.service, los cuales se utilizan para albergar las interfaces que definen los métodos que encapsulan la lógica del negocio. Estas interfaces actúan como contratos que especifican las operaciones que se pueden realizar en el sistema. Dentro de estas interfaces, se definen métodos que realizan tareas como la manipulación de datos, la ejecución de cálculos o la interacción con otros componentes del sistema. Estas interfaces proporcionan una capa de abstracción entre la capa de presentación y la capa de acceso a datos, permitiendo una separación clara de responsabilidades y facilitando la modularidad y el mantenimiento del código. En este mismo paquete se encuentran los 'serviceImpl' que albergan las clases que implementan la lógica del negocio definida en las interfaces de los 'service'. Estas clases son responsables de aplicar y ejecutar los métodos declarados en las interfaces, llevando a cabo las operaciones específicas que se requieren para satisfacer los requisitos funcionales del sistema.

```
com.edix.apirest.pets.service
> J DireccionService.java
> J DireccionServiceImpl.java
> J PedidoService.java
> J PedidoServiceImpl.java
> J ProductoService.java
> J ProductoServiceImpl.java
> J RolService.java
> J RolServiceImpl.java
> J TamanoService.java
> J TamanoServiceImpl.java
> J TarjetaService.java
> J TarjetaServiceImpl.java
> J UsuarioService.java
> J UsuarioServiceImpl.java
```

Controllers

A continuación se crean los ‘controllers’, ubicado en el paquete com.edix.apirest.pets.controller, estos juegan un rol crucial al actuar como la capa de conexión entre el backend y las solicitudes realizadas desde fuera de la aplicación, ya sea a través de una interfaz de usuario o servicios externos.

En los controllers se encuentran las clases que se encargan de recibir y gestionar las solicitudes HTTP entrantes, así como de enviar las respuestas correspondientes. Estas clases actúan como controladores, ya que se encargan de orquestar la lógica necesaria para procesar las solicitudes y coordinar las acciones pertinentes en el backend.

Dentro de los métodos de los controladores, se realiza la validación de los datos ingresados, se invocan los servicios correspondientes para ejecutar la lógica de negocio y se generan las respuestas adecuadas, que pueden incluir datos recuperados de la base de datos, mensajes de error o información procesada.

```
com.edix.apirest.pets.controller
> J CestaController.java
> J DireccionesController.java
> J HomeController.java
> J PedidosController.java
> J ProductoRestController.java
> J ProductosController.java
> J TarjetasController.java
> J UsuarioController.java
```

Seguridad

Para la parte de seguridad agregamos el paquete Spring Security que es una biblioteca incluida en el conjunto de herramientas de Spring que engloba una variedad de subproyectos o módulos que ofrecen funcionalidades adicionales para aplicaciones que lo consideren necesario. En particular, Spring Security se enfoca en unificar todas las características relacionadas con el control de acceso de usuarios en proyectos de Spring, que es para lo que lo usamos en este proyecto

El control de acceso permite restringir las acciones que un conjunto específico de usuarios o roles puede llevar a cabo en una aplicación. En este sentido, Spring Security se encarga de supervisar las llamadas a la lógica empresarial o limitar el acceso a ciertas URL en peticiones HTTP.

Para agregar la capa de seguridad realizamos una configuración sobre la aplicación indicando a Spring Security cómo debe comportarse la capa de seguridad.

El primer paso consiste en agregar las dependencias del paquete "spring-boot-starter-security". Esta inclusión traerá automáticamente las demás bibliotecas JAR necesarias para que Spring pueda implementar los mecanismos de seguridad requeridos. Simplemente con añadir esta dependencia Spring Boot protegerá por defecto todo el acceso a la aplicación, evitando que usuarios no identificados puedan invocar cualquier controlador.

En el archivo pom.xml se pueden ver las dependencias que añade.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>
```

Adicionalmente también añadimos la dependencia de las ‘taglibs’ que nos proveen acceso a la información de seguridad del los usuarios y posibilitan el poder filtrar el contenido que se muestra al usuario en base a sus privilegios o tipo de usuario del que se trate.

En el archivo pom.xml se pueden ver las dependencias que añade.

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-taglibs</artifactId>
    <version>5.4.5</version>
</dependency>
```

La etiqueta ‘authorize’ nos sirve para filtrar el contenido que se mostrará en base a los privilegios del usuario, por ejemplo, el contenido que solo será mostrado a aquellos usuarios con privilegios de administrador, esta etiqueta la utilizaremos de la siguiente manera:

```
<sec:authorize access="hasAuthority('Admin')">
    <li><a href="/lista-roles" class="nav-link px-2 text-black">Roles</a></li>
    <li><a href="/lista-usuarios" class="nav-link px-2 text-black">Usuarios</a></li>
</sec:authorize>
```

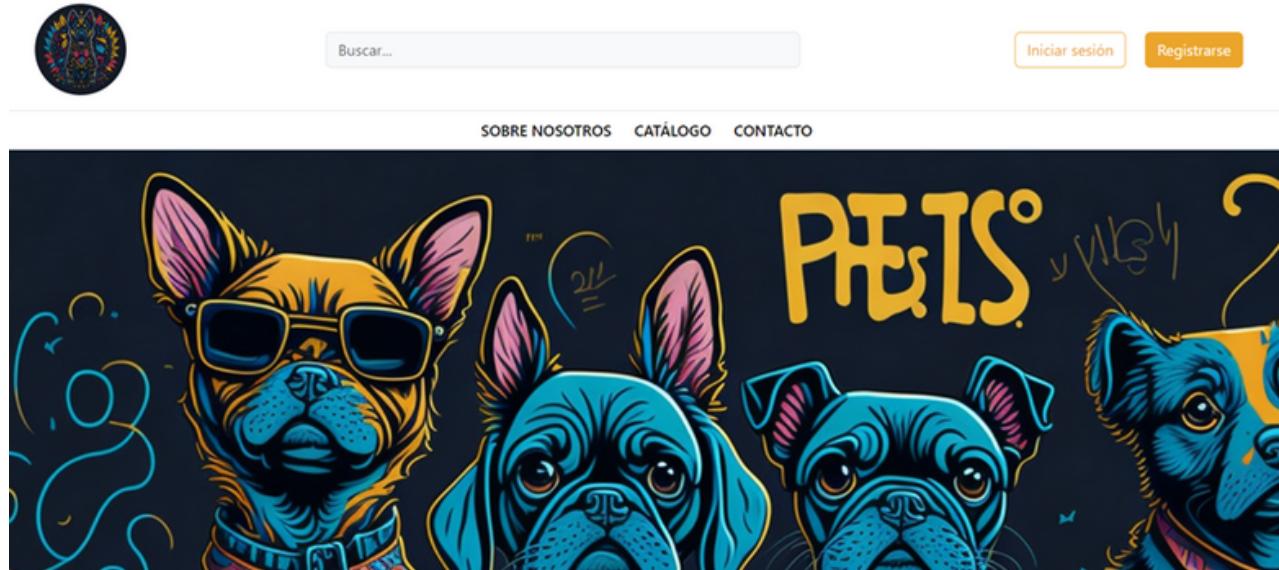
Aquí podemos ver como los elementos de ‘Roles’ y ‘Usuarios’ de la lista sólo los verán aquellos que tengan privilegios de ‘Admin’.

Otro ejemplo sería para los usuarios que no están registrados, como se puede ver a continuación:

```
<sec:authorize access="!isAuthenticated()">
    <a href="/index" class="btn btn-outline-primary me-3">Iniciar sesión</a>
    <a href="/registro" class="btn btn-primary me-3">Registrarse</a>
</sec:authorize>
```

Con esto los usuarios que entran sin estar registrados o haber iniciado sesión, se les muestran las opciones de ‘iniciar sesión’ y ‘Registrarse’

En nuestro caso tenemos dos roles, el usuario y el administrador, además también esta la persona que accede sin estar registrada o logueada, para esta última la aplicación solo mostrará en el menú la siguiente información.

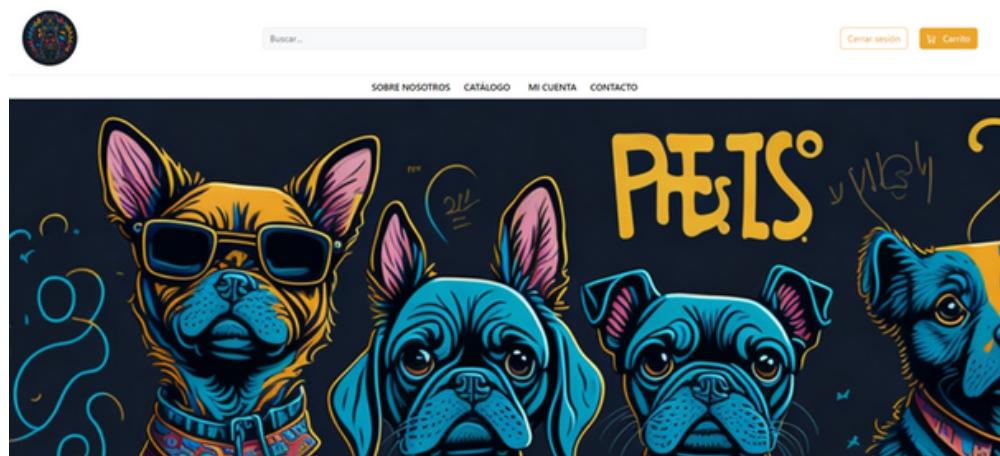


Opciones de usuarios según su rol

El usuario tiene la opción de iniciar sesión, registrarse y ver el carrito, en la opción de catálogo se puede ver el catálogo de productos y el detalle de cada uno.

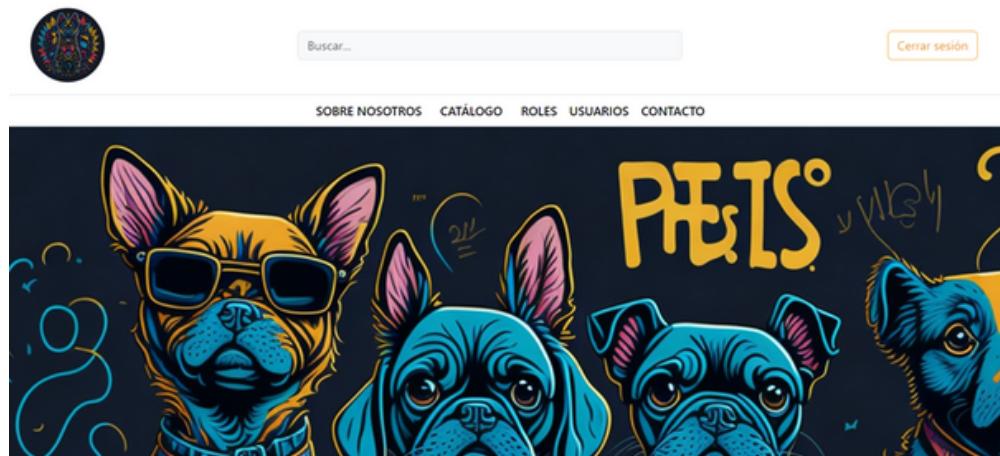
Lista de productos							
Filtros:		Todos	Pequeño	Mediano	Grande	↑ Precio ascendente	↓ Precio descendente
IMAGEN	NOMBRE	TAMAÑO		PRECIO	OPCIONES		
	Chubasquero Marvel	Pequeño		20.99 €	<button>Ver</button>		
	Abrigo Verde de plumas		Mediano	25.99 €	<button>Ver</button>		

Para el caso de los usuarios registrados muestra la siguiente información.



Como se puede observar añade al menú la opción de 'mi cuenta' dónde están todos los datos personales del usuario y es donde podrá modificarlos de así quererlo y la opción de cerrar sesión.

Luego esta el caso del administrador que nos muestra lo siguiente.



Como se puede apreciar, añade la opción de roles y de usuarios, ya que el administrador va a tener acceso a toda la información de la aplicación y a realizar cambios si es necesario.

En el caso del catálogo para el administrador, incluye las opciones de agregar un nuevo producto y editar o eliminar uno existente.



Filtros:						↑ Precio ascendente	↓ Precio descendente
IMAGEN	NOMBRE	TAMAÑO	PRECIO	OPCIONES			
	Chubasquero Marvel	Pequeño	20.99 €	<button>Ver</button>	<button>Editar</button>		
	Abrigo Verde de plumas	Mediano	25.99 €	<button>Ver</button>	<button>Editar</button>		
	Chaqueta Vaquera	Pequeño	22.99 €	<button>Ver</button>	<button>Editar</button>		

Como se puede apreciar, añade la opción de roles y de usuarios, ya que el administrador va a tener acceso a toda la información de la aplicación y a realizar cambios si es necesario.

En el caso del catálogo para el administrador, incluye las opciones de agregar un nuevo producto y editar o eliminar uno existente.

Front-end

Para el desarrollo del front-end usamos Bootstrap que es un framework de desarrollo web que combina HTML, CSS y JavaScript para facilitar la creación de interfaces, se basa en un sistema de rejillas (grid system) que organiza los elementos de la página en filas y columnas. Esto permite un diseño flexible y adaptable a diferentes dispositivos y tamaños de pantalla. Además, Bootstrap proporciona una amplia gama de componentes preestilizados, como botones, formularios, navegación, tarjetas, entre otros, que se pueden usar de manera sencilla, para usar estos componentes simplemente consiste en agregar las clases correspondientes que nos indica la documentación en las etiquetas HTML.

Para implementar Bootstrap en el proyecto, primeramente agregamos en el ‘head’ del HTML el <script> y los <link> que nos proporciona la documentación. Se pueden apreciar a continuación.

```
<html>
  <head>
    <meta charset="UTF-8">
    <title>Header</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="..."/>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.3/font/bootstrap-icons.css">
    <link rel="stylesheet" type="text/css" href="/css/styles.css" media="screen"/>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenUIKfdB1d...
  </head>
```

Luego es simplemente ir agregando las clases que se necesiten en cada caso, a continuación se puede ver un ejemplo de como se usan las clases.

```
<div class="py-3 container-fluid">
  <div class="d-flex flex-wrap align-items-center justify-content-around">
    <a href="/" class="d-flex align-items-center mb-2 mb-lg-0 text-black text-decoration-none">
      
    </a>
  </div>
```

También hemos usado CCS para dar algunos estilos adicionales, para ello hemos hecho el link de la hoja de estilos que esta ubicada en la siguiente ruta src/main/resources/static/css/style.css.

Para permitir mostrar información que cambia según lo que el usuario necesita se usan archivos JSP, de esta manera se hacen webs dinámicas.

Un archivo JSP es una página HTML con código Java dentro de ella. Permite combinar el contenido estático HTML con elementos dinámicos generados por el código Java. Los archivos JSP se ejecutan en el servidor web y el resultado generado se envía al cliente como una página HTML estática.

Back-end

A continuación y para no extender demasiado el documento se explica y se aportan imágenes de las acciones más relevantes que se pueden realizar.

Una vez creada la entidad Producto es necesario crear el repositorio que tiene las consultas asociadas a los métodos que vamos a declarar para obtener los resultados deseados. En nuestro caso vamos a necesitar hacer consultas que nos devuelvan una lista de productos la cual este filtrada por tamaño, una lista de productos que este ordenada por precio ascendente y descendente y una lista de productos que contenga la palabra clave que el usuario solicite.

Para realizar estas consultas hacemos uso de la anotación @Query que permite escribir instrucciones de SQL y exponerlas como métodos, a continuación se muestran los métodos creados en el repositorio de Producto.

```
package com.edix.apirest.pets.repository;

import java.math.BigDecimal;

public interface ProductoRepository extends JpaRepository<Producto, Integer>{

    @Query("select p from Producto p where p.tamano.nombre = ?1")
    public List<Producto> findByTamano(String tamano);

    @Query("select p from Producto p order by precio ASC")
    public List<Producto> orderByPriceAsc();

    @Query("select p from Producto p order by precio DESC")
    public List<Producto> orderByPriceDesc();

    @Query("select p from Producto p where p.nombre like ?1")
    public List<Producto> buscador(String nombre);
}
```

Una vez creado el repositorio creamos el service donde declaramos los métodos que vamos a necesitar para las funciones que vamos a realizar con los productos, entre los mas importantes estan eliminar, insertar y modificar un producto.

```
// Borrar un producto
int borrarProducto(int idProducto);

// Insertar un producto
int insertarProducto(Producto producto);

// Modificar un producto
int modificarProducto(Producto producto);
```

Seguidamente creamos la clase que implementa este service, esta lleva la lógica de cada método para aplicar las funcionalidades deseadas, en la siguiente imagen se muestra la implementación de los métodos definidos en el service.

```

// Borrar un producto
@Override
public int borrarProducto(int idProducto) {
    int filasBorradas = 0;
    try {
        prepo.deleteById(idProducto);
        filasBorradas=1;
    }catch(Exception e) {
        e.printStackTrace();
    }
    return filasBorradas;
}

// Insertar un producto
@Override
public int insertarProducto(Producto producto) {
    int filasInsertadas = 0;
    try {
        prepo.save(producto);
        filasInsertadas = 1;
    }catch(Exception e) {
        e.printStackTrace();
    }
    return filasInsertadas;
}

// Modificar un producto
@Override
public int modificarProducto(Producto producto) {

    int filasModificadas = 0;
    Producto prod = null;
    try {
        prod = prepo.getOne(producto.getIdProductos());
        prod = producto;
        prepo.save(prod);
        filasModificadas = 1;
    }catch(Exception e) {
        e.printStackTrace();
    }
    return filasModificadas;
}

```

Finalmente en el controller de los productos se crean los métodos que hacen uso de toda la lógica creada anteriormente y conecta esa información del back-end para ser mostrada en el front-end a través de los JSP.

Alta producto

Para dar de alta un producto vamos con la ayuda de la anotación @GetMapping vamos a mostrar un formulario, que en este caso estará definido en el JSP ‘alta-producto.jsp’, donde se podrán insertar los datos necesarios para registrar un nuevo producto.

```

// Página de formulario para crear un producto nuevo
@GetMapping("/alta-producto")
public String formAltaProd(Model model) {
    List<Tamano> lista = tserv.todosTamanos();
    model.addAttribute("listaFamilias", lista);

    return "alta-producto";
}

```

Posteriormente con la anotación @PostMapping y haciendo uso de la lógica creada en pasos anteriores, procesamos los datos obtenidos con el formulario lo añadimos a la base de datos. El usuario en el front verá un mensaje con la información de que su producto se ha añadido correctamente.

```

@PostMapping("/alta-producto")
public String procesarFormulario(RedirectAttributes ratt, Model model, Producto producto ) {
    System.out.println(producto);
    pserv.insertarProducto(producto);

    ratt.addFlashAttribute("mensaje", "<div class=\"alert alert-success\" role=\"alert\">\r\n"
        + " El producto se ha añadido con éxito\r\n"
        + "</div>");

    return "redirect:/lista-productos";
}

```

Eliminar

Para eliminar un producto hemos agregado un botón, que sólo verá el usuario con el rol de administrador, que aparecerá en cada producto de la lista de los mismos y que al pulsarlo el método que se muestra a continuación se encargará de obtener el id del producto en cuestión y aplicar la lógica creada el pasos anteriores para eliminar de la base de datos el producto.

```
@GetMapping("/borrar-producto/{id}")
public String eliminar(Model model, @PathVariable(name="id") int idProducto) {
    if (pserv.borrarProducto(idProducto) == 1)
        model.addAttribute("mensaje", "<div class=\"alert alert-success\" role=\"alert\">\r\n"
                           + " Producto eliminado con éxito\r\n"
                           + "</div>");
    else
        model.addAttribute("mensaje", "<div class=\"alert alert-warning\" role=\"alert\">\r\n"
                           + " El producto no se ha podido eliminar\r\n"
                           + "</div>");

    return "forward:/lista-productos";
}
```

Modificar

Para la modificación de un producto primeramente mostramos un formulario, que será un jsp llamado ‘editar-producto’, que contiene la toda la información del producto en ese momento, cuyos campos se pueden editar con la información que se necesite.

```
@GetMapping("/editar-producto/{id}")
public String enviarFormularioEditar(Model model, @PathVariable(name="id") int idProductos) {
    model.addAttribute("producto", pserv.buscarUno(idProductos));

    List<Tamano> lista = tserv.todosTamanos();
    model.addAttribute("listaFamilias", lista);

    return "editar-producto";
}
```

Dicho formulario tiene un botón que al pulsarlo se pondrá en funcionamiento la lógica que contiene el siguiente método que se muestra a continuación.

```
@PostMapping("/editar-producto/{id}")
public String procesarFormularioEditar(Model model, Producto producto, @PathVariable(name="id") int idProductos ) {

    if (pserv.buscarUno(idProductos) == null){
        model.addAttribute("mensaje", "<div class=\"alert alert-warning\" role=\"alert\">\r\n"
                           + " El producto no existe\r\n"
                           + "</div>");
    }else{
        producto.setIdProductos(idProductos);
        if (pserv.modificarProducto(producto) == 1) {
            model.addAttribute("mensaje", "<div class=\"alert alert-success\" role=\"alert\">\r\n"
                               + " Producto editado con éxito\r\n"
                               + "</div>");
        }else {
            model.addAttribute("mensaje", "<div class=\"alert alert-warning\" role=\"alert\">\r\n"
                               + " El producto no se ha podido editar\r\n"
                               + "</div>");

        }
    }
    return "redirect:/lista-productos";
}
```

Este método se encargará de sustituir la información que ingrese el administrador en el producto que desea modificar.

Usuarios y credenciales de acceso

Para operar y configurar la base de datos, a continuación se listan los nombres y contraseñas que se deberán utilizar.

Para la base de datos:

- Usuario = petsadministrator
- Contraseña = petsadministrator

Para la web:

- usuarios cliente = **crabago**, mprez, jgomez, druiz, lbarrio.
- usuario administrador = **smartinez**
- contraseña = 1234

Todos los usuarios tienen la misma contraseña.

Para figma:

<https://www.figma.com/file/SacAszwfmN2sT5h5EyEBjr/Untitled?type=design&node-id=0%3A1&t=la5xnkCkxLT3gm6E-1>

<https://www.figma.com/file/KeLzEu751Bn6FDS5GvqMFs/Pets-Administrator?type=whiteboard&t=2mkZ1pdBPNNeVvjw-1>

Conclusiones

Con el desarrollo de este proyecto, nuestro objetivo ha sido reflejar los conocimientos adquiridos a lo largo de nuestra formación, además de aplicar y consolidar los conocimientos teóricos y prácticos adquiridos durante la formación académica en el campo de desarrollo de aplicaciones web. Ha sido una oportunidad para poner en práctica habilidades y enfrentar desafíos tecnológicos reales.

En un principio la idea era crear una aplicación web para gestionar de manera eficiente el inventario de una tienda de ropa para mascotas. La aplicación permitirá registrar, modificar y eliminar productos en stock, así como realizar búsquedas avanzadas, sin embargo hemos querido agregar funcionalidades adicionales para que la aplicación pueda usarse en un futuro como una tienda online.

Hemos empezado primeramente creando la base de datos ya esta es el pilar fundamental y la base para tomar decisiones sobre el desarrollo del resto de los componentes de la aplicación, a partir de ahí hemos construido el back-end utilizando Spring Boot y Spring Boot Security usando Java como lenguaje de programación y para el front-end hemos utilizado vistas JSP, que están hechas con código HTML, las cuales le hemos dado estilos con Bootstrap, que es un framework que proporciona un conjunto de herramientas, componentes y estilos predefinidos que nos facilita y agiliza tanto la linea de diseño como el funcionamiento responsivo de la aplicación, también hemos hecho uso del lenguaje de programación JavaScript para dar algunas funcionalidades al front.

Durante el proceso de desarrollo se nos han ido presentando dificultades de cara a la base de datos ya que hemos añadido funcionalidades que no estaban previstas en un principio y que nos han obligado a hacer cambios necesarios para poder llevarlas a cabo.

Con vistas a futuro nos gustaría incorporar funcionalidades y agilizar procesos que por falta de tiempo no hemos podido hacer, los cuales son los siguientes:

- Configurar una pasarela de pago para que un usuario pueda finalizar su compra.
- Al agregar un producto al inventario poder cargar la imagen del mismo desde el front.
- Agregar una sección de posts y noticias relacionadas con los productos que se ofrecen.
- Permitir al usuario dejar una reseña o valoración del producto que ha comprado y así saber el grado de satisfacción.
- En la página principal colocar una sección de productos destacados u ofertas que se quieran resaltar.
- El formulario de contacto debe enviar un email a una dirección predeterminada por el administrador de la aplicación.