PROYECTO INTEGRADOR

Tecnicatura Superior en Desarrollo de Software

IFTS N°24

Alumna: Carla Rossi
Profesor: Julio Leppen

Titulo: “Desarrollo de aplicación de gestión de productos”

Agradecimientos

A mamá y a Mar por ser mis pilares y estar siempre.

Resumen

Este documento recopila toda la información vinculada al desarrollo del Proyecto Integrador, que consistió en la creación de un sitio web destinado a la gestión de productos para un negocio dedicado a la venta de cosmética y fragancias. La aplicación permite realizar operaciones de alta, baja, modificación y búsqueda de productos, ofreciendo así una solución funcional y accesible para el control del inventario mediante una interfaz intuitiva y amigable.

El principal objetivo fue aplicar de forma práctica los conocimientos adquiridos durante la carrera y explorar metodologías que permitan organizar y ejecutar un proyecto de forma individual y profesional.

Durante el proceso, se abordaron conceptos fundamentales como la manipulación del DOM, validación de formularios, almacenamiento de datos, autenticación segura y diseño web responsivo. Este proyecto no solo permitió consolidar habilidades técnicas, sino también fortalecer la capacidad de análisis, planificación y resolución de problemas en un entorno de desarrollo real.

Abstract

This document compiles all the information related to the development of the Integrative Project, which involved the creation of a website designed for product management in a business dedicated to the sale of cosmetics and fragrances. The application enables operations such as adding, deleting, editing, and searching for products, thus providing a functional and accessible solution for inventory control through an intuitive and user-friendly interface.

The main objective was to apply the knowledge acquired throughout the program in a practical way and to explore methodologies that support the organization and execution of an individual and professional-level project.

Throughout the development process, fundamental concepts were addressed, including DOM manipulation, form validation, data storage, secure authentication, and responsive web design. This project not only helped consolidate technical skills, but also strengthened analytical thinking, planning, and problem-solving abilities within a real-world development environment.

Política de confidencialidad y seguridad de datos

La aplicación desarrollada en este proyecto contempla el tratamiento de datos personales tanto de usuarios como de productos del sistema. Por tal motivo, se priorizó desde el inicio del desarrollo el cumplimiento de principios de seguridad, privacidad y confidencialidad de la información.

Todos los datos ingresados por los usuarios registrados —como nombre, apellido, provincia, celular, correo electrónico e imagen de perfil— son almacenados en una **base de** datos MySQL local. Si bien este entorno es de uso académico y de desarrollo, se tomaron medidas esenciales para proteger la integridad de los datos.

Las contraseñas se gestionan utilizando la librería **bcrypt**, la cual permite cifrarlas de forma segura. Esto significa que, en lugar de guardar las contraseñas tal como las escribió el usuario, el sistema almacena de manera cifrada. De esta manera, si alguien accediera a la base de datos, no podría recuperar las contraseñas reales.

El acceso al sistema está restringido únicamente a usuarios autenticados, mediante un mecanismo de validación por token JWT (JSON Web Token**)**. Esto asegura que solo los administradores autorizados puedan operar sobre la información sensible, como la gestión del inventario de productos.

Los formularios implementan validaciones tanto del lado del cliente (JavaScript**)** como del servidor (Node.js), con el fin de evitar la carga de datos incorrectos, minimizar vulnerabilidades comunes y proteger la integridad del sistema.

Además, el sistema no comparte información con terceros, ni utiliza los datos personales con fines comerciales o publicitarios. Toda la información recolectada se utiliza exclusivamente para el funcionamiento interno de la aplicación.

Este compromiso con la seguridad busca brindar una experiencia confiable, respetando principios éticos, técnicos y de buenas prácticas en el manejo de datos en entornos de desarrollo.

Declaración y consentimiento de usuarios

Al utilizar esta aplicación, el usuario declara haber leído, comprendido y aceptado los términos relacionados con el tratamiento de sus datos personales, detallados en la sección de Confidencialidad y Seguridad. El uso del sistema implica la aceptación explícita de estas condiciones.

Durante el proceso de registración, se solicita al usuario información personal utilizada exclusivamente para identificar a los administradores habilitados a operar dentro del sistema y no serán compartidos con terceros ni utilizados con fines comerciales.

El usuario podrá, en cualquier momento, solicitar la modificación o eliminación de sus datos, así como la baja de su cuenta.

Este compromiso busca garantizar el respeto por la privacidad, el cumplimiento de buenas prácticas en el manejo de datos y la transparencia en el funcionamiento de la aplicación.

Índice

1. Lista de abreviaturas y siglas 9
2. Glosario o Lista de términos clave 11
3. Objetivo del proyecto 13
4. Alcance y planteo del proyecto 14
   4.1 Requerimientos funcionales 16
   4.2 Requerimientos no funcionales 18
   4.3 Restricciones 20
5. Criterios de aceptación 21
6. Metodología 24
7. Diagramas 27
   7.1 Mapa de navegación 27
   7.2 Diagrama entidad-relación 28
   7.3 Diagrama de clases 30
   7.4 Diagrama de casos de uso 32
   7.5 EDT 34
8. Viabilidad del Proyecto 38
9. Conclusión y validación personal. 45
10. Entregables del proyecto 46
11. Listado de Software y Herramientas Utilizadas 47
12. Anexo 49
13. Bibliografías 57

1. Lista de abreviaturas y siglas

API: *Application Programming Interface* (Interfaz de Programación de Aplicaciones).

CRUD: *Create, Read, Update, Delete* (Crear, Leer, Actualizar, Eliminar).

JWT: *JSON Web Token* (Token Web basado en JSON).

HTML: *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto).

CSS: *Cascading Style Sheets* (Hojas de Estilo en Cascada).

JS: *JavaScript*.

DOM: *Document Object Model* (Modelo de Objetos del Documento).

SQL: *Structured Query Language* (Lenguaje de Consulta Estructurado).

GET, POST, PUT, DELETE: Métodos del protocolo HTTP utilizados para la comunicación cliente-servidor.

Git: Sistema de control de versiones distribuido.

GitHub: Plataforma para alojar proyectos con control de versiones mediante Git.

Multer: Middleware de Node.js para el manejo de archivos en formularios.

bcrypt: Librería para la encriptación segura de contraseñas.

Node.js: Entorno de ejecución para JavaScript del lado del servidor.

Express: Framework minimalista de Node.js para construir aplicaciones web.

EmailJS: Servicio para enviar correos electrónicos desde el frontend.

JSON: *JavaScript Object Notation* (Formato ligero para intercambio de datos).

UI: *User Interface* (Interfaz de Usuario).

UX: *User Experience* (Experiencia del Usuario).

2. Glosario o Lista de términos clave

Autenticación: Proceso mediante el cual un sistema verifica la identidad de un usuario, generalmente a través de credenciales como nombre de usuario y contraseña.

Base de datos relacional: Modelo de base de datos que organiza la información en tablas relacionadas entre sí mediante claves primarias y foráneas.

Bcrypt: Algoritmo de hash utilizado para almacenar contraseñas de forma segura mediante encriptación.

CRUD: Acrónimo de Create, Read, Update, Delete. Representa las operaciones básicas para la gestión de datos en una aplicación.

Frontend: Parte visual de una aplicación web con la que interactúa el usuario, generalmente construida con HTML, CSS y JavaScript.

HTTP: Protocolo de transferencia de hipertexto utilizado para la comunicación entre clientes y servidores en la web.

JWT (JSON Web Token): Estándar abierto que permite el intercambio seguro de información como objetos JSON, usado frecuentemente para autenticación.

Multer: Middleware de Node.js que permite procesar archivos enviados mediante formularios HTML.

Node.js: Entorno de ejecución de JavaScript que permite crear aplicaciones del lado del servidor.

Repositorio: Lugar donde se almacena, organiza y gestiona el código fuente de un proyecto, como los alojados en GitHub.

Responsive Design: Diseño web adaptable a diferentes tamaños de pantalla (PC, tablet, móvil).

SQL: Lenguaje estructurado de consulta utilizado para crear y gestionar bases de datos relacionales.

Token: Cadena de caracteres que funciona como identificador seguro en sesiones autenticadas, comúnmente generado con JWT.

Validación de formularios: Proceso mediante el cual se comprueba que los datos ingresados por el usuario cumplan con los requisitos esperados antes de enviarse al servidor.

Middleware**:** Componente intermedio que procesa las solicitudes antes de que lleguen al controlador principal.

Hash**:** Proceso de transformación de una entrada (como una contraseña) en una cadena fija, irreversible y segura.

Controlador (Controller)**:** Parte del backend que gestiona la lógica de negocio y responde a las peticiones del usuario.

Media Queries: condiciones que permiten que el diseño de una página web cambie automáticamente según el ancho, alto, resolución o tipo de dispositivo del usuario.

3. Objetivo del proyecto

El objetivo principal de este proyecto es desarrollar una aplicación web destinada a la gestión de productos para un negocio dedicado a la venta de cosmética y fragancias. Se busca permitir un control eficiente y detallado del inventario, facilitando operaciones de alta, baja, modificación y búsqueda de productos, mediante una interfaz accesible y responsiva.

Además, el proyecto tiene como propósito aplicar de forma práctica los conocimientos adquiridos durante la formación en desarrollo de software, incluyendo aspectos como la creación de bases de datos relacionales, el diseño de interfaces, la validación de formularios, la autenticación segura de usuarios y la organización modular del código. Esto permite integrar teoría y práctica en un proyecto individual con enfoque profesional y escalable.

4. Alcance y planteo del proyecto

El alcance del proyecto define qué se va a hacer y qué no se va a hacer. Es una descripción precisa de los límites del sistema, las funcionalidades que se van a desarrollar, los objetivos que se deben cumplir y los entregables esperados.

En este caso, el sistema permite llevar un control detallado del inventario, utilizando una base de datos estructurada con tablas específicas como productos, categorías, subcategorías, marcas, tipos, provincias, países y usuarios.

La aplicación ofrece operaciones de alta, baja, modificación y búsqueda de productos, los cuales cuentan con información completa como nombre, categoría, subcategoría, tipo, marca, descripción, precio e imagen. Toda esta información se organiza y relaciona con sus respectivas tablas auxiliares, lo que permite un manejo ordenado y escalable de los datos.

Para acceder a la plataforma, los usuarios deben autenticarse mediante un sistema de login seguro que utiliza contraseñas encriptadas (bcrypt) y validación por tokens (JWT). Cada usuario cuenta con un perfil personalizado que incluye nombre, apellido, provincia, pronombre, celular, mail e imagen de perfil. Esto permite identificar a los administradores autorizados para operar el sistema.

El sistema cuenta con formularios con validaciones integradas, tanto para los datos de usuario como para los de productos. Además, la carga de imágenes se realiza mediante la librería Multer, asegurando que los productos y perfiles cuenten con representaciones visuales adecuadas.

Desde el punto de vista visual, la interfaz fue desarrollada con HTML, CSS y JavaScript, prestando especial atención a la accesibilidad y la facilidad de uso, incluso para usuarios sin experiencia técnica. La manipulación del DOM y la validación de campos fueron aspectos clave durante el desarrollo.

Gracias al uso de herramientas como Node.js, MySQL, Git y GitHub, se logró una estructura modular basada en controladores, rutas y middlewares, lo que favorece el mantenimiento y la expansión futura del sistema.

4.1 Requerimientos funcionales

Los requerimientos funcionales definen lo que el sistema debe hacer. Son una parte fundamental de cualquier proyecto de software, ya que describen las funcionalidades específicas que la aplicación debe ofrecer para satisfacer las necesidades del usuario o del negocio.

4.1.1 Gestión de Inventario de Productos

Se permiten operaciones de alta, baja (física y lógica), modificación y búsqueda de productos.
Los productos están vinculados a las siguientes tablas auxiliares: categorías, subcategorías, tipos, marcas (cada marca está asociada a un país).
Atributos de cada producto: nombre, categoría, subcategoría, tipo, marca, descripción, precio e imagen.

4.1.2 Sistema de Autenticación de Usuarios

Implementación de login seguro, con contraseñas encriptadas mediante bcrypt.
Cada usuario posee un perfil personalizado con los siguientes datos: nombre, apellido, provincia, pronombre, celular, correo electrónico e imagen.
Solo los usuarios autenticados pueden acceder al panel de gestión del sistema.

4.1.3 Manejo de Tablas Auxiliares

Las siguientes tablas mantienen la integridad referencial y facilitan la carga dinámica en formularios: categorías, subcategorías, tipos, marcas, países, provincias y pronombres.

4.1.4 Formularios Web

El formulario de registro de usuarios cuenta con validación de campos y carga de imagen de perfil.

El formulario de contacto está integrado con EmailJS, permitiendo el envío seguro de mensajes desde el frontend.

4.1.5 Carga de Imágenes Local

El sistema permite cargar y almacenar imágenes asociadas a productos y perfiles de usuario.

4.1.6 Consumo de API

El sistema realiza una llamada GET a una API propia para mostrar contenido dinámico en el sitio, demostrando la capacidad de integrar datos desde el backend hacia el frontend en tiempo real.

4.1.7 CRUD Completo

Se utiliza un soporte para realizar operaciones CRUD sobre todas las tablas del sistema, empleando los métodos HTTP GET, POST, PUT y DELETE.

4.2 Requerimientos no funcionales

Los requisitos no funcionales son aquellas características que definen cómo debe comportarse el sistema. No describen funciones específicas, sino atributos como la seguridad, la usabilidad, la escalabilidad, entre otros.

4.2.1 Seguridad

Las contraseñas son almacenadas de forma segura mediante hashing con bcrypt.

Hay protección de rutas y sesiones mediante tokens JWT.

Solo administradores autenticados pueden acceder al sistema de gestión (no hay roles múltiples).

4.2.2 Accesibilidad y Usabilidad

La interfaz es clara y responsiva, con atención a la usabilidad incluso para usuarios sin experiencia técnica.

Las validaciones están del lado del cliente (JavaScript) para todos los formularios.

4.2.3 Diseño Responsive

El diseño es adaptable a tres tamaños de dispositivos: escritorio, tablet y móvil, mediante media queries en CSS.

4.2.4 Rendimiento y Escalabilidad

La carga local de imágenes mediante Multer es aceptable en entornos pequeños o medianos. No se utiliza almacenamiento en la nube, lo que limita la escalabilidad a futuro.

4.2.5 Mantención y Estructura del Código

Se utilizó Flexbox como sistema de maquetación principal para estructurar y alinear elementos en la interfaz. Se incorporan íconos y tipografías externas.

La arquitectura modular está basada en controladores, rutas y middlewares (Node.js)

El uso de herramientas de control de versiones (Git y GitHub) para aseguran la trazabilidad.

4.2.6 Dependencia Tecnológica

Las tecnologías utilizadas son: Node.js, Express, MySQL, Multer

Cambiar cualquiera de estas tecnologías implica una reestructuración importante del sistema.

4.2.7 Repositorio y Despliegue

El código fuente debe estar disponible en un repositorio GitHub.

El sitio web se despliega en un servidor online navegable (Netlify) para su evaluación.

4.3 Restricciones

4.3.1 Acceso restringido a administradores

Solo los usuarios autenticados con permisos de administrador pueden acceder a las funciones de gestión de productos. Por el momento, no se contemplan múltiples roles (por ejemplo, clientes o empleados con distintos niveles de acceso).

4.3.2 Sin integración con medios de pago o carrito de compras

El sistema está orientado exclusivamente a la gestión interna del inventario. No incluye funcionalidades para ventas en línea, pagos electrónicos ni procesamiento de pedidos.

4.3.3 Carga local de imágenes

Las imágenes se cargan y almacenan de forma local en el servidor mediante Multer. No se utiliza un sistema de almacenamiento en la nube, lo que limita la escalabilidad del sistema.

4.3.4 Dependencia de tecnologías específicas

El sistema depende directamente de tecnologías como Node.js, MySQL y Multer. Cambiar cualquiera de estas herramientas requeriría reestructurar partes clave del código y la arquitectura.

5. Criterios de aceptación

Para considerar que cada funcionalidad del sistema está correctamente implementada, se deben cumplir los siguientes criterios de aceptación:

5.1 Gestión de Productos (CRUD)

El sistema permite crear, modificar, eliminar (física y lógicamente) y buscar productos.

Cada producto debe incluir todos sus atributos: nombre, categoría, subcategoría, tipo, marca, descripción, precio e imagen. También deben estar correctamente relacionados con las tablas auxiliares (categorías, tipos, marcas, países, etc.).

La información se guarda y recupera correctamente desde la base de datos relacional SQL.

5.2 Autenticación de Usuarios

El sistema implementa un login seguro usando bcrypt para el cifrado de contraseñas.

Solo usuarios autenticados mediante token JWT pueden acceder a las funciones administrativas. Cada usuario cuenta con un perfil personalizado (nombre, apellido, provincia, pronombre, celular, mail, imagen).

5.3 Validación de Formularios

Todos los formularios deben tener validación en frontend y backend.

No se debe permitir enviar formularios incompletos o con datos inválidos.

El sistema debe mostrar mensajes claros de error y confirmación.

5.4 Carga de Imágenes

La aplicación debe permitir la carga de imágenes tanto para productos como para perfiles de usuario, utilizando Multer. Las imágenes deben visualizarse correctamente una vez cargadas.

No se deben aceptar archivos que no sean imágenes o que tengan extensiones no válidas.

5.5 Diseño Responsivo

La interfaz debe adaptarse correctamente a tres tamaños de dispositivos: PC, tablet y celular, usando media queries garantizando la legibilidad, accesibilidad y navegación fluida en todos los dispositivos.

La maquetación debe hacerse con Flexbox y/o Grid.

5.6 Frontend y UI

El sitio debe contener al menos 6 páginas HTML bien estructuradas.

Debe incluir un formulario de contacto y un formulario de registración con al menos 5 campos, incluyendo: Un checkbox o radiobutton, un select, y una imagen.

Incluir iconos e integración de fuentes.

5.7 API

El sistema debe realizar una llamada GET a una API pública o API propia para mostrar contenido dinámico.

5.8 Seguridad

Las contraseñas deben estar encriptadas, nunca en texto plano.

Las rutas protegidas requieren autenticación con token válido.

El sistema no debe exponer datos sensibles ni permitir acceso sin validación.

5.9 Usabilidad

La interfaz debe ser intuitiva y clara, incluso para usuarios sin conocimientos técnicos.

Todos los elementos visuales deben respetar el diseño definido: colores, tipografías, iconografía, jerarquía visual.

5.10 Despliegue y Repositorio

El sitio debe estar publicado online en un servidor accesible (Netlify, Vercel, Replit, etc.).

El código fuente debe estar alojado en un repositorio GitHub (público o privado con acceso al docente).

El backend debe estar integrado con el frontend en un sistema funcional.

6. Metodología

El desarrollo del proyecto se llevó a cabo siguiendo una metodología tradicional estructurada por fases, que permitió organizar y ejecutar las tareas de forma secuencial y controlada. Esta metodología fue seleccionada por su adecuación a un proyecto individual, donde el alcance, los recursos y los objetivos estaban bien definidos desde el inicio.

Las principales etapas del desarrollo fueron:

6.1 Recolección de Requerimientos

Se identificaron las funcionalidades clave del sistema, los actores involucrados (usuarios administradores) y los datos a manejar. Esta etapa incluyó el análisis del dominio y la definición de los objetivos funcionales y no funcionales.

6.2 Diseño

Se trabajó en dos niveles:

Diseño visual (UI/UX): se definieron colores, tipografías, disposición de elementos y estructura navegacional utilizando herramientas como Figma.

Diseño técnico: se elaboraron diagramas de estructura, navegación, entidad-relación y clases UML para organizar el backend y la base de datos.

6.3 Desarrollo

El sistema fue construido de forma modular utilizando Node.js y Express en el backend, y HTML, CSS y JavaScript en el frontend.

Se implementaron:

- Rutas y controladores para cada operación CRUD.
- Autenticación con bcrypt y JWT.
- Carga de imágenes mediante Multer.
- Validaciones en frontend y backend.
- El código fue versionado mediante Git y alojado en GitHub.

6.4 Pruebas

Se realizaron pruebas funcionales y de seguridad para asegurar el correcto funcionamiento del sistema. Las pruebas se enfocaron en:

- Validación de formularios.
- Accesos autorizados y no autorizados.
- Visualización y persistencia de datos.
- Compatibilidad en diferentes dispositivos (responsive design).
- Despliegue

El sitio fue publicado en un servidor online, lo que permitió comprobar su accesibilidad y funcionamiento real. Se usó una infraestructura gratuita basada en Netlify para el frontend, y base de datos local para el desarrollo.

6.5 Documentación

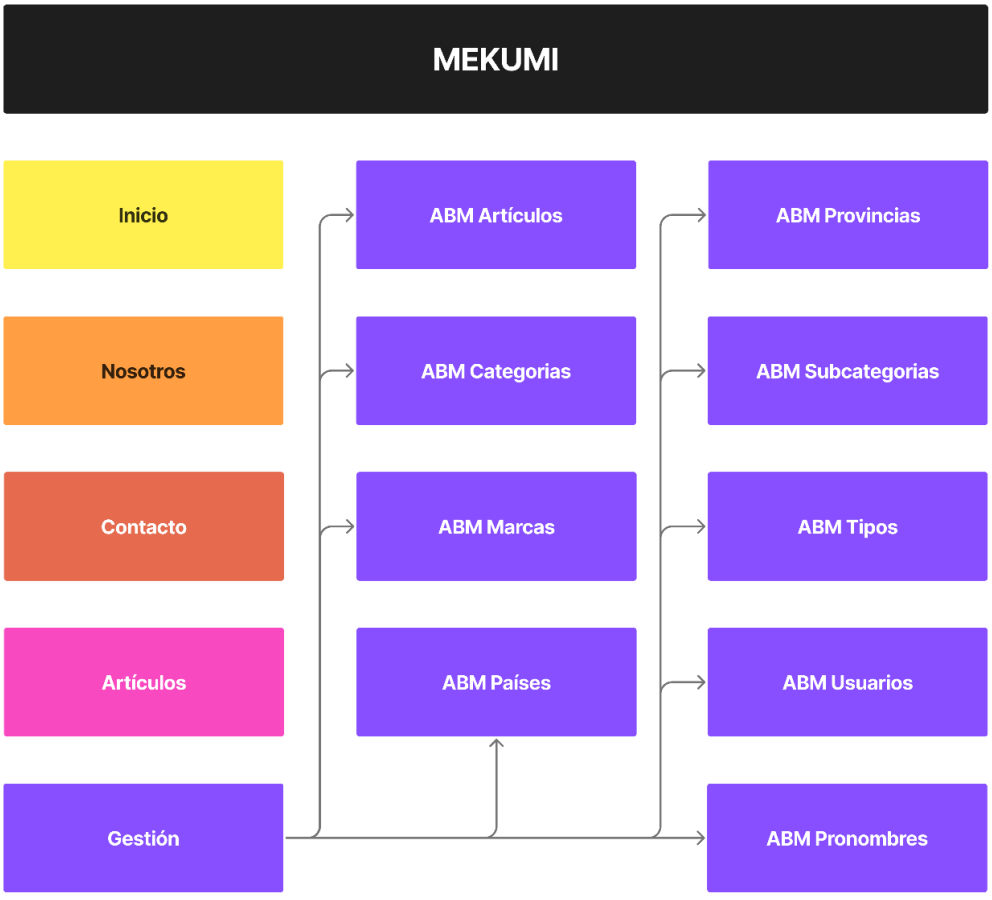
Se generó la documentación técnica y del usuario, incluyendo manuales, diccionario de la EDT, diagramas y criterios de aceptación, para dejar constancia clara del proceso y facilitar futuras modificaciones o expansiones del sistema.

7. Diagramas

7.1 Mapa de navegación (Figma)

El mapa de navegación es un diagrama visual que muestra cómo se conectan las distintas páginas o pantallas del sitio web. Representa el flujo de navegación entre secciones. Ayuda a definir la arquitectura de la información, planificar el recorrido del usuario y garantizar que el sistema sea intuitivo.

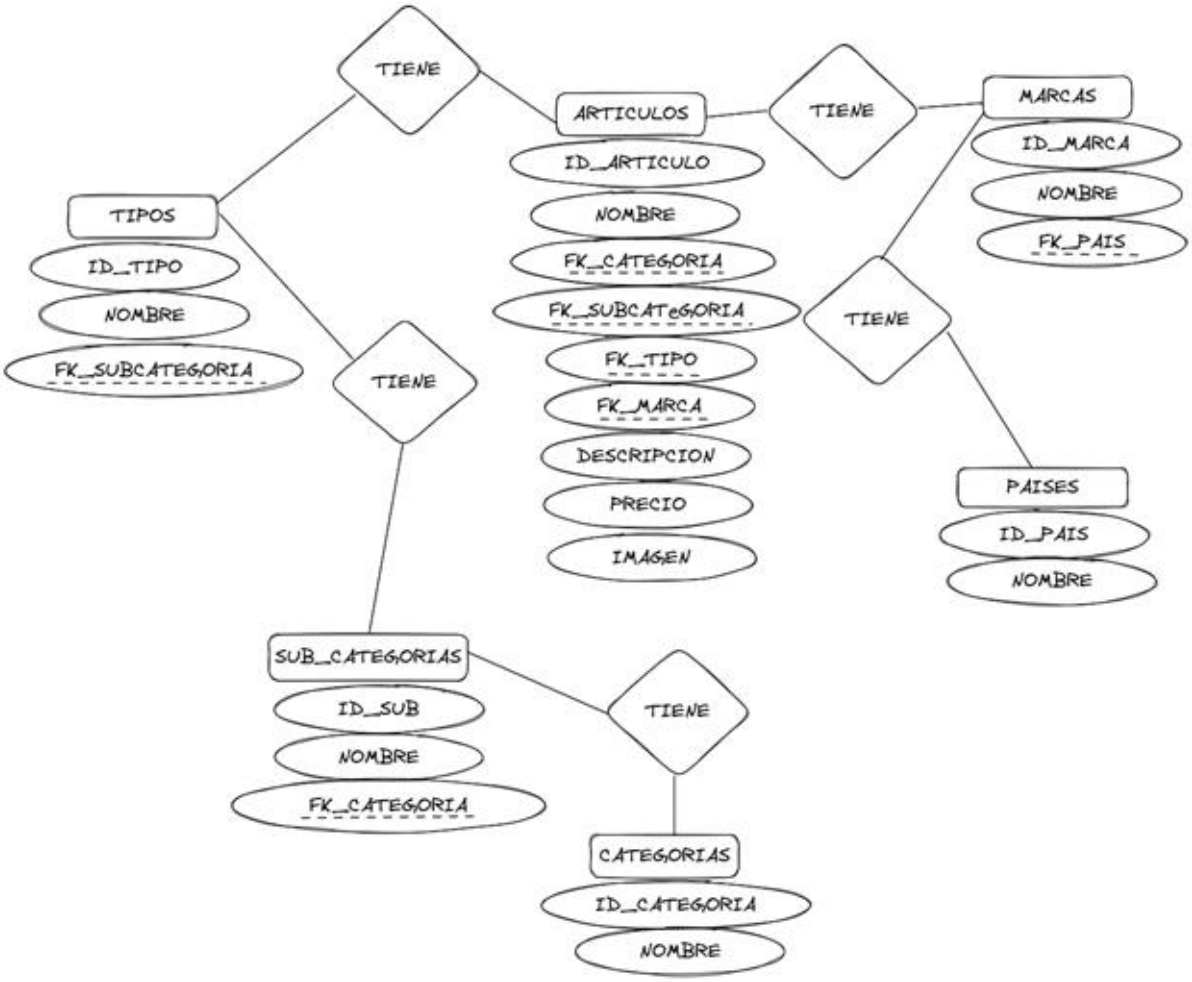
Se usó en el proyecto para organizar visualmente las secciones clave: inicio, productos, formulario de contacto, login, registro y perfil. Esto facilitó la maquetación inicial y la jerarquía del contenido.

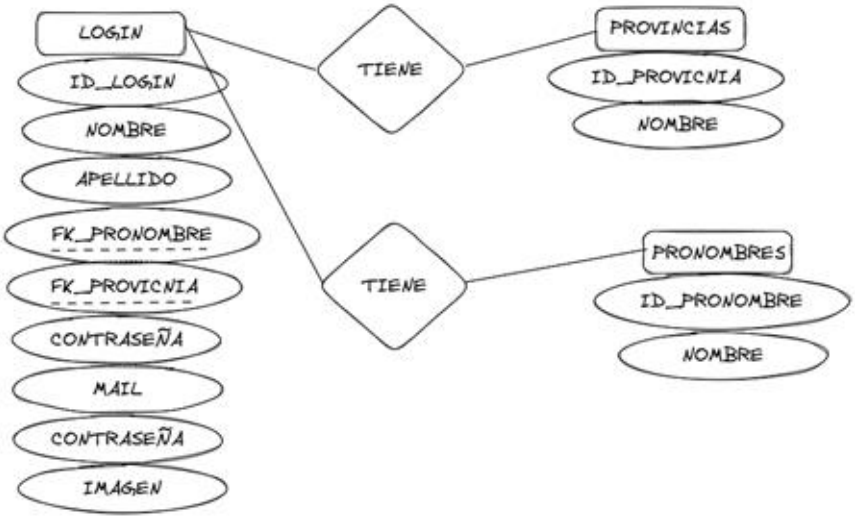


7.2 Diagrama entidad-relación (excalidraw)

El Diagrama entidad-relación es la representación gráfica de la estructura de una base de datos relacional, con tablas, atributos, claves primarias y claves foráneas. Permite visualizar las relaciones entre los datos y asegurar la integridad referencial del sistema.

Se utilizó para definir cómo se relacionan productos con categorías, marcas, tipos, subcategorías, usuarios, provincias, países y pronombres. Fue clave para construir correctamente la base de datos en MySQL.

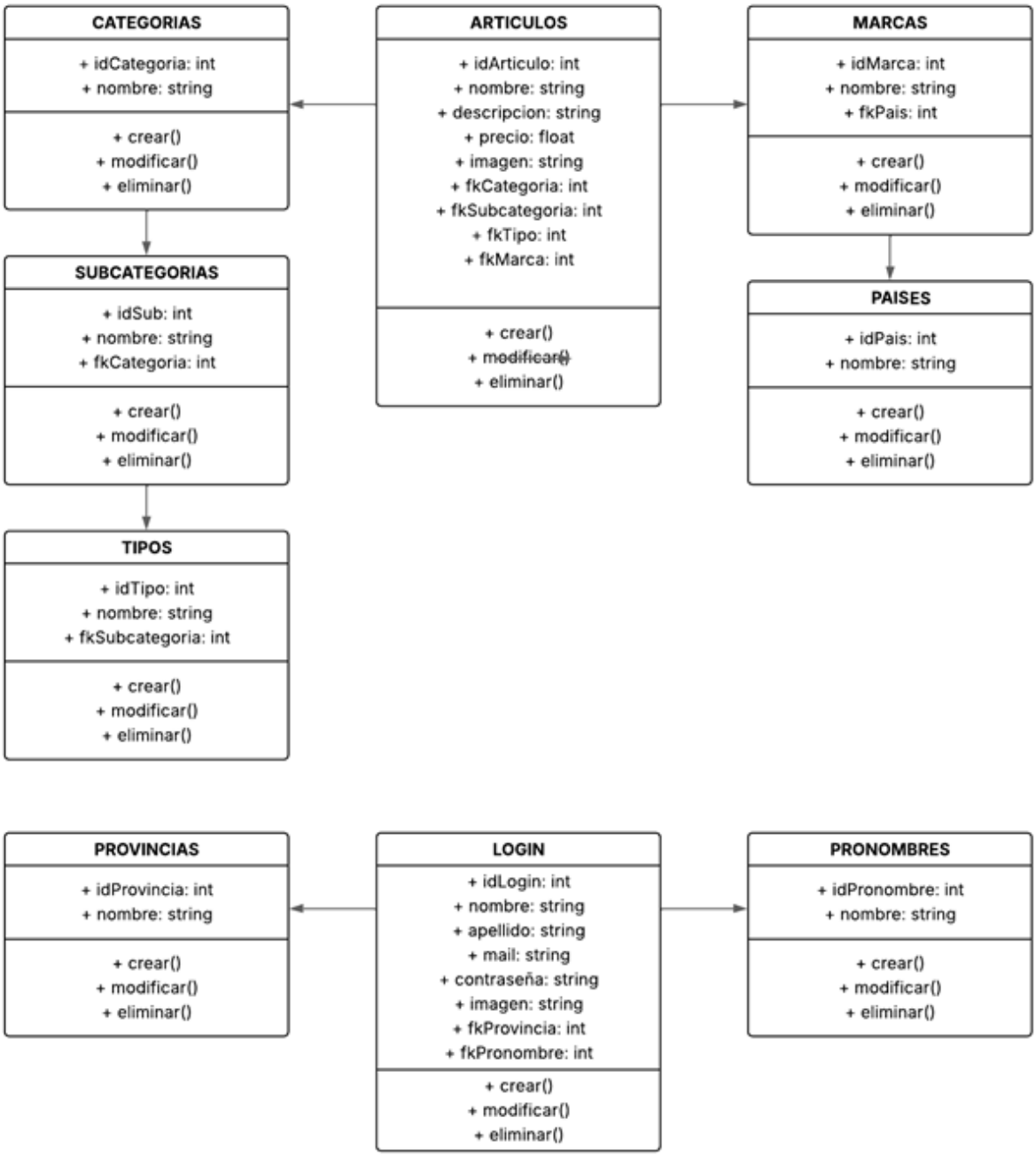

TIENE
ARTICULOS
TIENE
MARCAS
ID_MARCA
NOMBRE
FK_PAIS
ID_ARTICULO
NOMBRE
FK_CATEGORIA
FK_SUBCATEGORIA
FK_TIPO
FK_MARCA
DESCRIPCION
PRECIO
IMAGEN
TIPOS
ID_TIPO
NOMBRE
FK_SUBCATEGORIA
TIENE
TIENE
PAISES
ID_PAIS
NOMBRE
SUB_CATEGORIAS
ID_SUB
NOMBRE
FK_CATEGORIA
TIENE
CATEGORIAS
ID_CATEGORIA
NOMBRE


LOGIN
ID_LOGIN
NOMBRE
APELLIDO
FK_PRONOMBRE
FK_PROVICNIA
CONTRASEÑA
MAIL
CONTRASEÑA
IMAGEN
TIENE
PROVINCIAS
ID_PROVICNIA
NOMBRE
TIENE
PRONOMBRES
ID_PRONOMBRE
NOMBRE

7.3 Diagrama de clases (lucidchart)

El diagrama de clases es un diagrama UML que representa las clases del sistema, con sus atributos, métodos y relaciones. Permite modelar la estructura lógica del sistema desde el enfoque de la programación orientada a objetos.

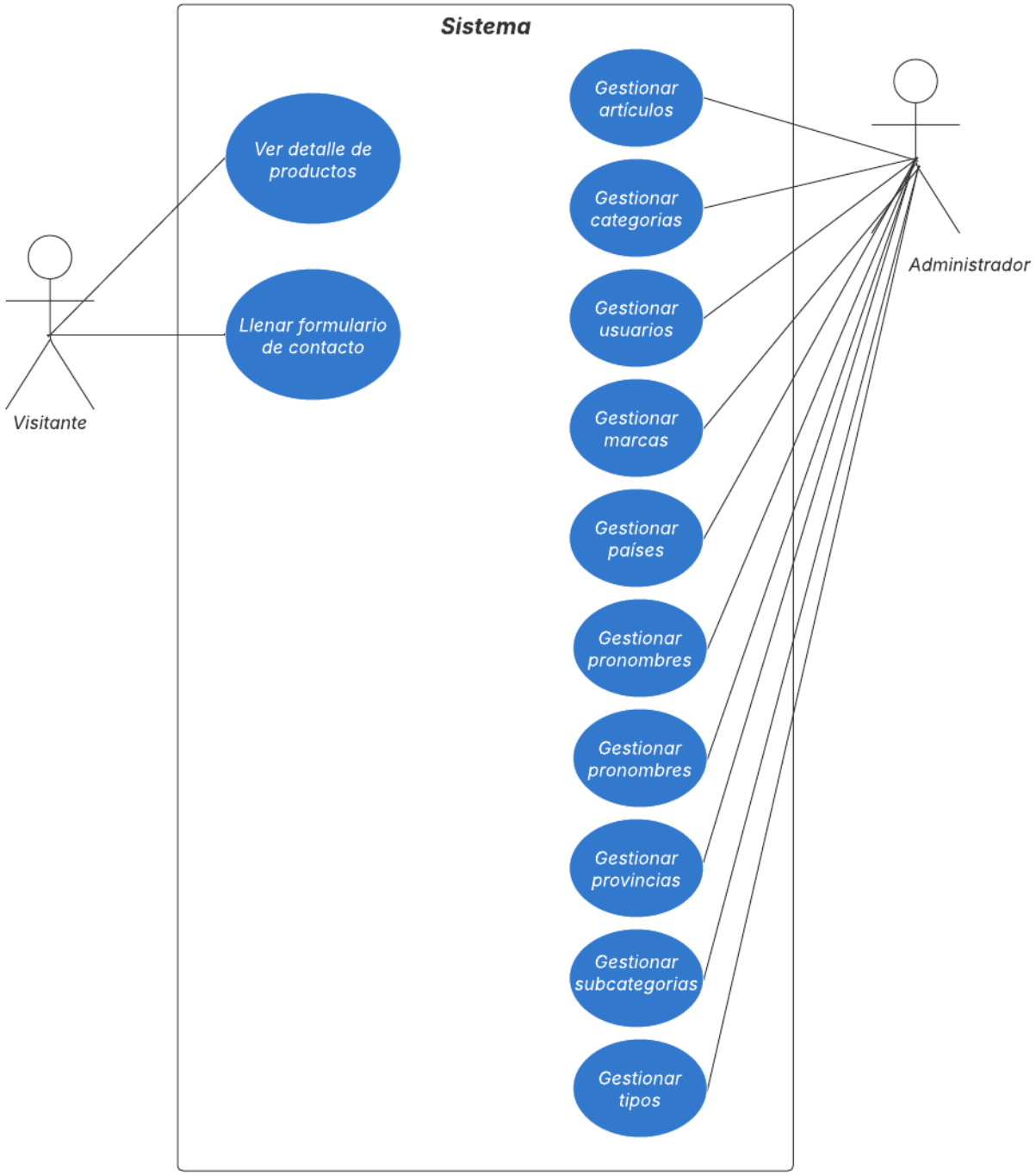
Sirvió para poder reflejar las entidades principales del sistema y sus atributos, junto con las funciones básicas que se pueden realizar (crear, editar, eliminar). Ayudó a organizar el desarrollo del backend.


CATEGORIAS
+ idCategoria: int
+ nombre: string
+ crear()
+ modificar()
+ eliminar()
ARTICULOS
+ idArticulo: int
+ nombre: string
+ descripcion: string
+ precio: float
+ imagen: string
+ fkCategoria: int
+ fkSubcategoria: int
+ fkTipo: int
+ fkMarca: int
+ crear()
+ modificar()
+ eliminar()
MARCAS
+ idMarca: int
+ nombre: string
+ fkPais: int
+ crear()
+ modificar()
+ eliminar()
SUBCATEGORIAS
+ idSub: int
+ nombre: string
+ fkCategoria: int
+ crear()
+ modificar()
+ eliminar()
PAISES
+ idPais: int
+ nombre: string
+ crear()
+ modificar()
+ eliminar()
TIPOS
+ idTipo: int
+ nombre: string
+ fkSubcategoria: int
+ crear()
+ modificar()
+ eliminar()
PROVINCIAS
+ idProvincia: int
+ nombre: string
+ crear()
+ modificar()
+ eliminar()
LOGIN
+ idLogin: int
+ nombre: string
+ apellido: string
+ mail: string
+ contraseña: string
+ imagen: string
+ fkProvincia: int
+ fkPronombre: int
+ crear()
+ modificar()
+ eliminar()
PRONOMBRES
+ idPronombre: int
+ nombre: string
+ crear()
+ modificar()
+ eliminar()

7.4 Diagrama de casos de uso (lucidchart)

El diagrama de caso de uso es un diagrama UML (Lenguaje Unificado de Modelado) que describe las acciones que los usuarios pueden realizar con el sistema (casos de uso), y qué actores están involucrados. Ayuda a identificar y delimitar las funcionalidades del sistema desde la perspectiva del usuario.

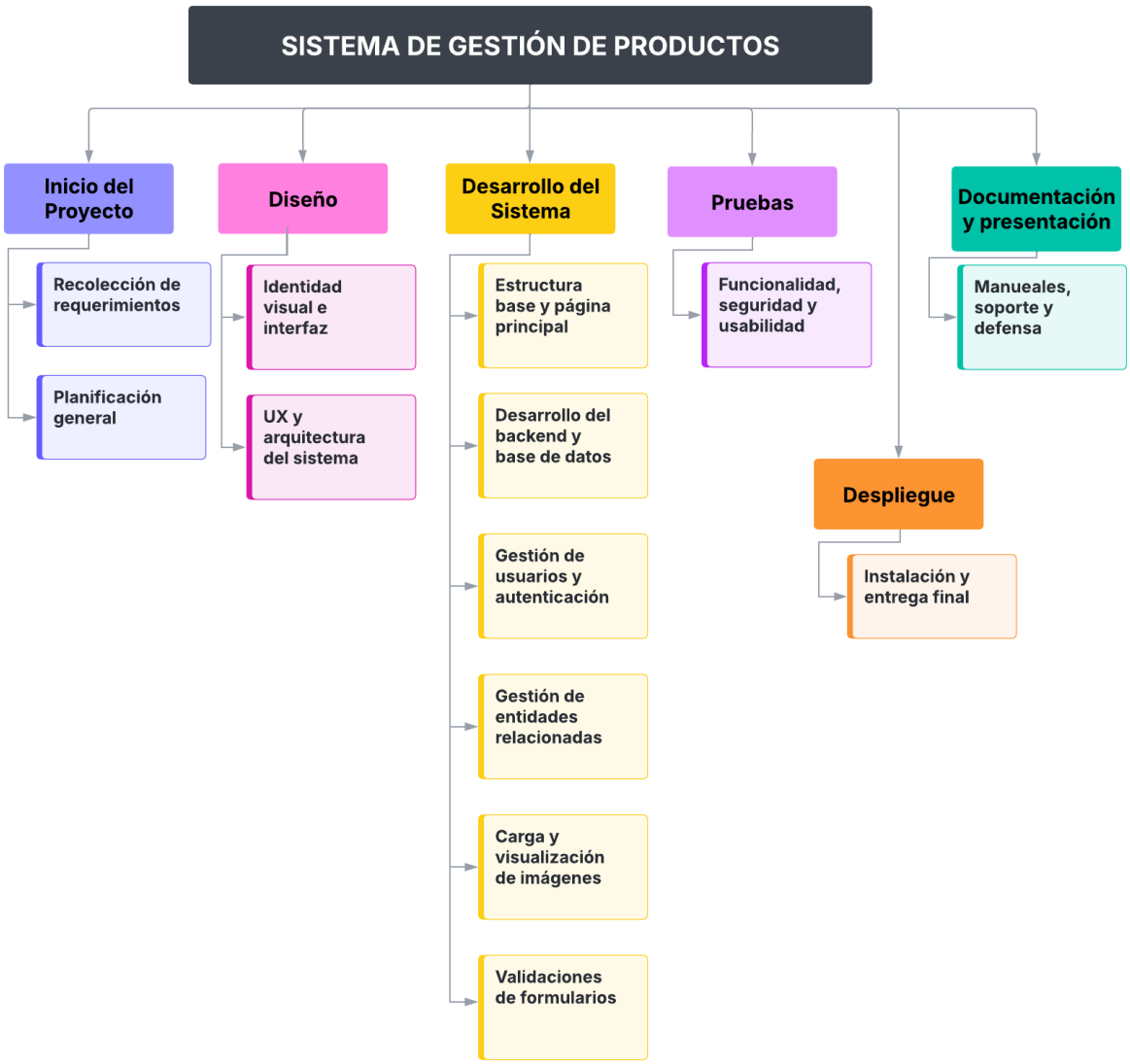
Se usó para definir claramente qué puede hacer un administrador (gestionar productos, usuarios, tablas auxiliares) y qué puede hacer un visitante (ver productos, llenar formulario). No se implementaron roles múltiples, por lo que el acceso está restringido.


Sistema
Ver detalle de productos
Llenar formulario de contacto
Visitante
Gestionar artículos
Gestionar categorias
Gestionar usuarios
Gestionar marcas
Gestionar países
Gestionar pronombres
Gestionar pronombres
Gestionar provincias
Gestionar subcategorias
Gestionar tipos
Administrador

7.5 EDT (lucidchart)

El diagrama de Estructura de desglose de trabajo (EDT) es una representación jerárquica que divide el proyecto en tareas y sub-tareas organizadas por fases. Permite planificar, organizar y controlar el desarrollo del proyecto, dividiéndolo en partes manejables.

En este caso se usó para organizar el trabajo en fases: inicio, diseño, desarrollo, pruebas, despliegue y documentación. Esto permitió un enfoque estructurado, desde la recolección de requerimientos hasta la entrega final.



7.5.1 Diccionario de la EDT

El diccionario EDT detalla las actividades planificadas para el desarrollo del sistema de gestión de productos, desde la recolección de requerimientos hasta la entrega final. Este documento fue fundamental para estructurar el trabajo, asignar prioridades y garantizar una visión global y ordenada del proceso de desarrollo.

1. Inicio del Proyecto

1.1 Recolección de requerimientos

Se identifican las funcionalidades necesarias del sistema, los actores (usuarios administradores) y los datos a manejar. Se documentan los objetivos del sistema según el alcance.

1.2 Planificación general

Se define el cronograma de trabajo, herramientas, fases del desarrollo y entregables previstos. Incluye la distribución del tiempo por tareas principales.

2. Diseño

2.1 Identidad visual e interfaz

Se define la estética visual de la aplicación, incluyendo colores, fuentes, iconografía, disposición de elementos, y el estilo visual general (UI).

2.2 UX y arquitectura del sistema

Se diseña la estructura de navegación y la experiencia del usuario final, priorizando accesibilidad, organización lógica y facilidad de uso.

3. Desarrollo del Sistema

3.1 Estructura base y página principal

Se construye la base del sitio web y se implementa la página de inicio, con enlaces funcionales y diseño inicial.

3.2 Desarrollo del backend y base de datos

Se programan las rutas, controladores y consultas para realizar operaciones CRUD sobre todas las tablas del sistema. Se implementa y relaciona la base de datos MySQL.

3.3 Gestión de usuarios y autenticación

Se desarrolla el sistema de login con cifrado de contraseñas (bcrypt) y validación JWT. Solo administradores pueden acceder al panel de gestión.

3.4 Gestión de entidades relacionadas

Se habilita la carga, edición y eliminación de datos auxiliares como categorías, subcategorías, tipos, marcas, países, provincias y pronombres.

3.5 Carga y visualización de imágenes

Se integra Multer para permitir la carga local de imágenes en productos y perfiles de usuario, mostrando correctamente las imágenes en la interfaz.

3.6 Validaciones de formularios

Se implementan validaciones tanto en frontend como en backend para evitar errores de ingreso y garantizar la integridad de los datos.

4. Pruebas

4.1 Funcionalidad, seguridad y usabilidad

Se testean las funcionalidades del sistema, la seguridad del acceso y el uso en distintos dispositivos para verificar la experiencia del usuario.

5. Despliegue

5.1 Instalación y entrega final

Se publica la aplicación en un servidor online y se verifica su funcionamiento completo. El código queda disponible en un repositorio.

6. Documentación y Presentación

6.1 Manuales, soporte y defensa

Se generan los manuales técnicos y de usuario, se prepara la presentación del proyecto y se deja constancia del desarrollo realizado.

8. Viabilidad del Proyecto

8.1 Análisis de Costos

El presente apartado tiene como finalidad estimar los costos asociados al desarrollo, implementación y posible despliegue profesional del sistema, en caso de que el proyecto se llevara a producción. Si bien actualmente se trata de una aplicación de uso académico, se contemplan los recursos técnicos y de infraestructura necesarios para garantizar su funcionamiento a escala real.

8.1.1 Costos de Desarrollo

El desarrollo del sistema demandó aproximadamente **160 horas de trabajo**,. Si se considera una tarifa promedio de $7.000 por hora (valor estimado para un desarrollador freelance con conocimientos full-stack), el costo estimado de desarrollo sería:

160 h × $7.000 = $1.120.000

Este monto incluye también el tiempo dedicado al diseño de interfaz, experiencia de usuario (UX/UI) y creación del logo e identidad visual.

8.1.2 Infraestructura

Rubro	Costo anual	Descripción
Dominio Web	$ 11.500	Dominio.com.ar (pago anual único)
Hosting (Netlify)	$ 0	Plan gratuito
Base de datos (MySQL)	$ 0	Alojada localmente
EmailJS	$ 0	Uso bajo plan gratuito
Total infraestructura	$11.500	

8.1.3 Mantenimiento y soporte técnico

En un sistema en producción, el mantenimiento implica:

- Corrección de errores.
- Actualizaciones de dependencias.
- Optimización de base de datos y mejoras visuales.

Se estima un mínimo de 10 horas mensuales de soporte técnico, a un valor de $7.000/hora:

10 h/mes × $7.000 × 12 meses = $840.000/año

8.1.4 Resultado general

Categoría	Costo estimado anual
Desarrollo (una vez)	$ 1.120.000
Infraestructura básica	$ 11.500
Mantenimiento (12 meses)	$ 840.000
Total estimado	$ 1.971.500

8.1.5 Conclusión

Aunque el proyecto se desarrolló con recursos gratuitos y una infraestructura local, este análisis permite visualizar los costos reales asociados a un sistema de gestión de productos si fuera desplegado en un entorno profesional. Considerando únicamente el desarrollo, la infraestructura básica y el mantenimiento anual, se obtiene una estimación clara del nivel de inversión necesario. A través del uso de herramientas accesibles y servicios gratuitos de calidad, es posible mantener un sistema funcional con una inversión razonable, lo que lo vuelve una solución viable para pequeñas y medianas empresas del sector comercial.

8.2 Análisis de Escalabilidad del Proyecto

Aunque el sistema fue concebido como una solución de gestión interna de productos, su arquitectura modular y uso de tecnologías modernas permiten **una evolución gradual hacia una aplicación web comercial integral**. A continuación, se detallan los aspectos clave que permitirían escalar el proyecto en distintas direcciones:

8.2.1 Escalabilidad Funcional

Para transformar el sistema en una tienda virtual con funcionalidades de venta en línea, se podrían incorporar las siguientes extensiones:

- Carrito de compras
- Agregar una sección para clientes donde puedan agregar productos al carrito, modificar cantidades y confirmar pedidos.
- Implementar persistencia de carrito por sesión o por usuario autenticado.
- Relación directa con el sistema de stock.
- Pasarela de pagos.
- Integrar servicios de pago como MercadoPago o MODO.
- Validar transacciones, generar comprobantes y registrar historial de compras.
- Cifrado de datos sensibles y cumplimiento con normas de seguridad.
- Gestión de roles de usuario.
- Incorporar múltiples roles: Administrador, Vendedor, Cliente.

- Control de permisos para definir qué acciones puede realizar cada tipo de usuario.
- Diseño de interfaces diferenciadas según rol.
- Módulos adicionales
- Registro y seguimiento de pedidos.
- Reportes de ventas y estadísticas de productos.
- Valoraciones y reseñas por parte de los clientes.
- Notificaciones por mail ante cambios de estado de pedido.

8.2.2 Escalabilidad Técnica

Para soportar el crecimiento funcional, también deben considerarse mejoras en la infraestructura y arquitectura técnica:

- Almacenamiento en la nube: Migrar imágenes y archivos a servicios en la nube, lo cual mejora el rendimiento y la escalabilidad.
- Optimización de base de datos: Implementar paginación**,** índices y normalización avanzada para soportar consultas con miles de registros.
- Agregar auditoría de cambios y backups automáticos.
- Preparar la plataforma para soportar múltiples idiomas y monedas.

8.2.3 Escalabilidad de Usuarios y Seguridad

A medida que aumente el número de usuarios y operaciones simultáneas, será necesario:

Usar sistemas de autenticación escalables con refresh tokens y expiración.

Implementar control de sesiones, bloqueo de cuentas tras intentos fallidos y verificación de email.

8.2.4 Conclusión

El sistema actual constituye una base sólida para escalar a un nivel profesional. Con la incorporación de un carrito, pasarela de pagos, distintos roles y mejoras técnicas, se puede evolucionar hacia una plataforma de e-commerce completa. Esta escalabilidad está favorecida por el uso de tecnologías abiertas y buenas prácticas de desarrollo, lo cual permite:

- Crecer sin reescribir el sistema desde cero.
- Añadir nuevas funcionalidades sin comprometer la estabilidad actual.
- Adaptarse a las necesidades de una pyme o incluso de una tienda online profesional.

8.3 Posibilidades de Expansión / Comercialización

El sistema desarrollado presenta un alto potencial de expansión más allá del entorno académico, tanto en términos de funcionalidades como de aplicación comercial real. Su arquitectura modular, uso de tecnologías estándar y orientación a la gestión de inventario lo convierten en una base adaptable a múltiples contextos del sector comercial.

8.3.1 Adaptabilidad a otros rubros

Aunque fue diseñado para una tienda de cosmética y fragancias, la estructura del sistema permite adaptarlo fácilmente a otros sectores como indumentaria, librerías, ferreterías, electrónica, alimentos, etc.

Simplemente modificando algunas tablas auxiliares (categorías, tipos, marcas), el sistema puede reutilizarse sin necesidad de rediseño completo.

8.3.2 Posibilidad de convertirlo en tienda virtual

El sistema puede evolucionar hacia una **tienda online** completa mediante la incorporación de:

- Carrito de compras
- Registro de clientes
- Gestión de pedidos
- Integración con pasarelas de pago.
- Sistema de seguimiento de envíos y notificaciones por email
- Estas mejoras permitirían transformar el sistema en una plataforma de ventas en línea, ampliando su valor de uso comercial.

8.3.3 Comercialización como servicio (SaaS)

Con algunas modificaciones y mejoras en escalabilidad, el sistema podría ofrecerse como un producto SaaS (Software as a Service), es decir, como una solución lista para usar por otros negocios, bajo un modelo de suscripción. Esto implicaría:

- Alojamiento en la nube
- Módulo de registro para múltiples clientes
- Panel administrativo independiente por empresa

- Escalabilidad por demanda (planes pagos según volumen)

8.4 Conclusión

La aplicación desarrollada no solo cumple con los objetivos académicos, sino que **está** preparada para evolucionar hacia una solución útil y aplicable en el mundo real. Con un desarrollo adicional y enfoque estratégico, puede convertirse en una herramienta comercializable que aporte valor a pequeñas y medianas empresas del sector minorista.

9. Conclusión y validación personal.

El desarrollo de este proyecto integrador representó tanto un desafío como una valiosa oportunidad para aplicar de forma práctica los conocimientos adquiridos a lo largo de la carrera. Desde los primeros bocetos hasta la publicación del sistema, cada etapa permitió profundizar en aspectos clave del desarrollo web, tanto en frontend como en backend, consolidando una experiencia de aprendizaje integral.

A nivel técnico, se logró construir una solución funcional, segura y escalable para la gestión de productos, cumpliendo con todos los requerimientos funcionales y no funcionales definidos. Se implementaron tecnologías modernas como Node.js, Express, MySQL, JWT, bcrypt y Multer, integrando eficazmente todas las capas del sistema. La arquitectura modular, la validación de formularios, la autenticación de usuarios, la carga de imágenes y el consumo de API reflejan una aplicación robusta y adaptable a futuras mejoras o ampliaciones.

Asimismo, el uso de herramientas como Git y GitHub, la planificación estructurada, la documentación completa y el despliegue final en un servidor online demuestran un enfoque profesional y ordenado del trabajo. Más allá de los logros técnicos, este proyecto también fortaleció habilidades personales como la organización, la resolución de problemas, la toma de decisiones y la mejora continua frente a errores o imprevistos.

En definitiva, el proyecto no solo cumplió con los objetivos académicos, sino que también evidenció la capacidad de diseñar, desarrollar y entregar una solución tecnológica concreta, aplicable tanto en el ámbito educativo como en entornos reales del mercado profesional.

10. Entregables del proyecto

Los entregables representan los productos y resultados finales que evidencian el desarrollo completo del proyecto, tanto desde el aspecto técnico como académico. Son elementos que permiten evaluar el cumplimiento de los objetivos planteados y asegurar la trazabilidad del trabajo realizado.

Los entregables principales de este proyecto son:

- Documento final del proyecto: informe técnico que incluye el planteo del problema, los requerimientos, el diseño, el desarrollo, las decisiones técnicas, pruebas, restricciones y conclusiones.
- Presentación oral o exposición: instancia de defensa donde se explica el proceso de desarrollo, las tecnologías utilizadas y la funcionalidad del sistema ante docentes o evaluadores.
- Producto funcional: sistema web publicado en un servidor (Netflify), con frontend y backend integrados y operativos.
- Repositorio de código fuente: acceso al código completo del sistema mediante una plataforma de control de versiones (GitHub), que permite su revisión, validación o reutilización.

Cada uno de estos entregables refleja una etapa fundamental del proceso y permite validar tanto el aprendizaje adquirido como la aplicación concreta de conocimientos técnicos, organizativos y comunicacionales.

11. Listado de Software y Herramientas Utilizadas

Durante el desarrollo del proyecto se utilizaron diversas herramientas, lenguajes y servicios que permitieron llevar a cabo cada etapa de diseño, programación, prueba y despliegue. A continuación, se detalla el conjunto de tecnologías empleadas:

11.1 Lenguajes de Programación y Tecnologías Web

- HTML5. Estructura semántica del sitio.
- CSS3. Estilos visuales, diseño responsivo con Flexbox y media queries.
- JavaScript. Validaciones, manipulación del DOM, lógica de interacción.
- Node.js. Entorno de ejecución del backend.
- Express.js. Framework para la creación de rutas y middlewares en el servidor.
- SQL (MySQL). Lenguaje de consulta para bases de datos relacionales.

11.2 Librerías y Middleware

- bcrypt. Encriptación segura de contraseñas.
- JWT (jsonwebtoken). Gestión de autenticación mediante tokens.
- Multer. Carga y validación de archivos/imágenes desde formularios.
- EmailJS. Envío de correos electrónicos desde el frontend.
- FontAwesome. Iconografía visual.
- Google Fonts y Sergio Trendy. Tipografías externas personalizadas.

11.3 Herramientas de Desarrollo

- Visual Studio Code (VS Code). Editor de código principal.
- Postman. Testeo de rutas API (GET, POST, PUT, DELETE).
- MySQL/ phpMyAdmin. Diseño, consulta y gestión de base de datos.
- Git. Control de versiones local.
- GitHub. Repositorio remoto para control de versiones y colaboración.

11.4 Herramientas de Diseño y Diagramación

- Figma. Prototipado y mapa de navegación visual.
- Lucidchart. Diagramas UML: casos de uso, clases, EDT.
- Excalidraw. Diagrama entidad-relación (base de datos).

11.5 Plataformas de Despliegue y Hosting

- Netlify. Hosting del frontend (versión pública navegable).

12. Anexo


INSTITUTO DE FORMACIÓN TÉCNICA SUPERIOR NÚMERO 24
(GRUPO) N2
Material Design
By Google
SEMINARIO DE PROFUNDIZACIÓN Y ACTUALIZACIÓN
PROF. LAMAS EDGARDO
NUESTRO EQUIPO
CARLA ROSSI
DANIELA BERRIOS
JUAN PAREDES
KEREN CURURO
PAULO FORTES ROCO
19 DE JUNIO DE 2025

¿Qué es Material Design?

Material Design es un sistema de diseño creado por Google en 2014 orientado principalmente a Android. Su concepto se inspira en el papel y la tinta, lo que le da un aspecto visual más realista y con profundidad, diferenciándolo del diseño plano que predominaba antiguamente. De ahí el nombre: "Material", haciendo referencia a una interfaz que se comporta como un material físico, con profundidad, bordes, sombras y movimiento, pero en un entorno digital.

El objetivo principal de Material Design es ofrecer una experiencia de usuario coherente en distintos dispositivos, plataformas y formas de interacción. Así, sin importar si un usuario accede a un producto de Google desde un teléfono, una computadora o una tablet, la experiencia visual y de uso será consistente.

Este sistema no solo establece cómo deben verse los elementos de la interfaz, sino también cómo deben comportarse. Incluye guías detalladas sobre tipografía, cuadrículas, espaciado, colores, imágenes, animaciones y transiciones. Además, fomenta el uso de jerarquía visual y un diseño intencional, permitiendo a los diseñadores crear interfaces claras, intuitivas y enfocadas en mejorar la experiencia del usuario.

Gracias a Material Design, las aplicaciones logran ser visualmente atractivas y fáciles de usar, manteniendo una identidad visual unificada.

Línea del tiempo

La iniciativa de Material Design fue liderada por Matías Duarte, diseñador de experiencia de usuario y figura clave en el desarrollo del sistema Android. Google comenzó a trabajar en este nuevo enfoque en el 2014.

La presentación oficial de Material Design ocurrió en Junio de 2014 en el evento Google I/O donde se presentó cómo este lenguaje de diseño no solo era visualmente atractivo, sino que estaba cuidadosamente estructurado con principios matemáticos para garantizar consistencia y claridad.

Ese mismo año fue lanzado Android Lollipop (versión 5.0) y fue el primer sistema operativo de Google construido desde cero con Material Design como guía. A partir de ese momento, Google comenzó a rediseñar todas sus aplicaciones más populares (Gmail, Google Maps, Drive y YouTube) para adaptarlas al nuevo estándar visual. También se publicaron guías y herramientas para que los desarrolladores de terceros pudieran adoptarlo en sus propias aplicaciones, tanto móviles como web.

A medida que se fue adaptando este estilo, comenzaron a surgir librerías como Materialize, Material Components for the Web y Material-UI (para React), que facilitaron su implementación más allá del entorno directo de Google.

En 2018, Google presentó Material Design 2, con un enfoque más limpio, menos dependiente de las sombras pronunciadas, y con elementos más personalizables. La idea era permitir que las marcas pudieran adaptarlo a sus propias identidades visuales sin perder las bases de usabilidad y accesibilidad que lo caracterizaban.

La siguiente gran evolución ocurrió con Android 12 en 2021, cuando Google introdujo Material You, un rediseño más centrado en la personalización. Esta nueva versión permitió que la interfaz del sistema se adapte dinámicamente al gusto del usuario, cambiando colores y formas según el fondo de pantalla y otras preferencias personales.

Actualmente, Material Design sigue siendo el corazón del diseño visual de Google, y continúa evolucionando con nuevas guías, componentes y principios, reflejando una intención clara: ofrecer experiencias digitales que sean no solo consistentes, sino también intuitivas, accesibles y visualmente atractivas para todos los usuarios.

¿Cómo incorporar Material Design a tu proyecto integrador?

Plataforma de Ejemplo: Web Don Giovanni

Materialize CSS es un framework de diseño web basado en Material Design, el sistema visual creado por Google. Su objetivo es ayudarte a construir sitios web modernos, ordenados, responsivos y accesibles, sin necesidad de crear todo el diseño desde cero.

Incluir Materialize en tu proyecto

Debes agregar los archivos de estilo y script de Materialize CSS en tu HTML.

¿Dónde? En la parte `<head>` y antes del cierre del `<body>` de tus archivos HTML.

¿Cómo? Usando links a la CDN (no necesitás instalar nada si no querés).

Organizar tu estructura usando el sistema de grillas

Material Design trabaja con un sistema de columnas. Usá clases como `row` y `col s12 m6` para ubicar tus elementos de forma ordenada y responsive.

¿Dónde? En todas las secciones: menú, reservas, formulario de contacto, etc.

Aplicar componentes visuales

Materialize te ofrece muchos componentes ya listos para usar. Los más destacados para este proyecto serán:

Tarjetas: para mostrar los platos del menú con imagen y descripción.

Botones: para enviar formularios o navegar.

Formularios: para el contacto y las reservas.

Selectores de fecha y hora: ideales para el sistema de reservas.

Personalizar colores y fuentes

Materialize permite cambiar los colores principales (botones, fondos, textos) para que combinen con la identidad visual del restaurante.

¿Cómo? Usando clases predeterminadas (`red darken-3`, `brown lighten-1`) o escribiendo tus propios estilos CSS.

Asegurar diseño responsive

Todo en Materialize ya es responsive, pero igual es importante probar en celulares, tablets y computadoras.

¿Cómo? Redimensionando la ventana del navegador o utilizando las herramientas de desarrollo del navegador (inspector).

Ventajas de utilizar Material Design

Ecosistema de diseño completo

Material Design no es solo un conjunto de reglas visuales, sino un ecosistema completo. Ofrece guías detalladas para casi cualquier situación de diseño, incluyendo casos de uso complejos que suelen pasar desapercibidos en sistemas menos exhaustivos. Esto brinda a los diseñadores una estructura sólida sobre la cual trabajar.

Consistencia en la experiencia de usuario

Garantiza una experiencia visual y de uso coherente en todas las plataformas y dispositivos (móviles, tabletas, escritorio). Los usuarios reconocen patrones comunes, lo que mejora la usabilidad y refuerza la familiaridad con los productos.

Documentación actualizada y soporte constante

Google mantiene activamente Material Design y proporciona documentación exhaustiva y actualizada. Este nivel de soporte es superior al que ofrecen muchos otros sistemas de diseño, facilitando el aprendizaje y la implementación.

Flexibilidad en la implementación K

A pesar de sus reglas claras, Material Design deja espacio para la personalización. Los diseñadores pueden adaptar componentes y estilos para responder a las necesidades específicas del producto o de la identidad visual de la marca.

Jerarquía visual y diseño accesible D

Promueve el uso adecuado de tipografía, cuadrículas, espaciado y color, lo que permite crear interfaces con jerarquía visual clara. Además, fomenta prácticas de diseño accesible, facilitando el uso por parte de personas con distintas capacidades.

Componentes reutilizables D

Existen librerías y frameworks (Material-UI, Angular Material, etc.) que facilitan la implementación de componentes reutilizables, agilizando el desarrollo y garantizando coherencia en la interfaz.

Retroalimentación de usuario integrada D

Las guías de Material Design incluyen el uso de animaciones sutiles, transiciones fluidas y retroalimentación háptica (cuando es posible), lo que mejora la experiencia de interacción y aporta una sensación de respuesta inmediata.

13. Bibliografías

- Normas APA 7ma Edición

- ChatGPT

https://donweb.com/es-ar/dominio-com-ar