# Confitería Santa Ana Web

Indice

[Confitería Santa Ana Web 1](#_Toc200790689)

[1. Justificación del proyecto 1](#_Toc200790690)

[2. Funcionalidades en las que se divide. 2](#_Toc200790691)

[3. Módulos del ciclo formativo que intervienen 3](#_Toc200790692)

[4. Tecnologías elegidas y justificación de dicha elección 4](#_Toc200790693)

[5. Etapas por las que ha pasado el proyecto 6](#_Toc200790694)

[6. Problemas encontrados y soluciones aportadas 7](#_Toc200790695)

[7. Nivel de consecución del mismo 8](#_Toc200790696)

[8. Recursos necesarios para su implementación y presupuesto económico 9](#_Toc200790697)

# Justificación del proyecto

El proyecto **"Confitería Santana Web"** surge con la intención de dar respuesta a una necesidad real detectada en el entorno familiar: la modernización de una confitería tradicional que, hasta ahora, gestionaba sus operaciones de manera manual o mediante herramientas poco eficientes y desconectadas entre sí. A través de esta solución web, se busca ofrecer una herramienta completa para la **gestión integral del negocio**, abarcando aspectos como la **venta online**, el **control de inventario**, la **gestión de pedidos y ventas**, así como la **atención al cliente y administración de proveedores**.

La decisión de abordar este proyecto nace de la voluntad de **ayudar a un familiar** a optimizar la gestión de su empresa, facilitándole el salto hacia la digitalización y adaptándose a las nuevas demandas del mercado, que cada vez exige más presencia en línea, automatización y eficiencia. Este tipo de necesidad es compartida por muchos pequeños negocios tradicionales que no disponen de los recursos para contratar soluciones a medida o integrarse en plataformas grandes que no se adaptan a sus particularidades.

Este sistema no solo mejora los procesos internos como la gestión de stock, la facturación o la atención al cliente, sino que también abre nuevas oportunidades al permitir a la empresa vender productos a través de Internet, ampliar su base de clientes y ofrecer una experiencia de compra más cómoda y moderna. Además, fomenta la toma de decisiones basadas en datos al proporcionar información en tiempo real sobre ventas, pedidos y productos más solicitados.

Aunque en una primera fase el desarrollo se enfoca en las necesidades específicas de una **confitería**, el diseño del proyecto es **modular, flexible y escalable**, lo que permite su adaptación a cualquier otro tipo de comercio minorista, como panaderías, tiendas de alimentación, negocios de productos artesanos, floristerías o tiendas de ropa. Esto lo convierte en una solución con un **alto potencial de reutilización y proyección en el mercado**, sobre todo en el contexto actual donde la digitalización de negocios locales es cada vez más urgente y necesaria.

En definitiva, se trata de un proyecto con **utilidad práctica inmediata, impacto directo sobre un negocio real** y potencial de evolución y ampliación a futuro.

# Funcionalidades en las que se divide.

Se divide en múltiples funcionalidades que cubren tanto las necesidades internas de gestión del negocio como los servicios orientados al cliente final. Se ha diseñado una arquitectura modular que permite ofrecer una experiencia completa de comercio electrónico y administración, con especial atención a la facilidad de uso, escalabilidad y eficiencia operativa.

**Funcionalidades de administración:**

* **Dashboard de administración**: Muestra un resumen visual de la actividad del negocio (pedidos, ventas, productos más vendidos, etc.), lo que permite una supervisión ágil y centralizada.
* **Gestión de pedidos**: Permite revisar, actualizar y organizar todos los pedidos realizados. Incluye información detallada de productos, clientes, estado del pedido y método de pago.
* **Gestión de productos**: Alta, edición, eliminación y visualización de productos, incluyendo su descripción, imagen, precio, stock, alérgenos y categoría correspondiente.
* **Gestión de categorías**: Organización jerárquica de los productos para facilitar tanto la búsqueda del usuario como la administración del catálogo.
* **Gestión de alérgenos**: Control sanitario y de información alimentaria, mediante la asociación de alérgenos a los productos, mejorando la seguridad para el consumidor.
* **Gestión de clientes**: Visualización de datos de los usuarios registrados, acceso a su historial de pedidos y posibilidad de actualización de datos.
* **Gestión de ventas**: Registro detallado de todas las ventas realizadas, tanto online como en tienda física, incluyendo desglose de productos y totales.
* **TPV (Terminal Punto de Venta)**: Herramienta interna para realizar ventas rápidas en tienda, con interfaz adaptada a dispositivos táctiles y flujos de trabajo ágiles.
* **Gestión de almacén**: Control de stock actualizado, registro de entradas y salidas, y visualización de movimientos de productos para un control logístico completo.
* **Página de inicio de administración**: Interfaz inicial desde la cual se puede acceder a todas las funcionalidades de gestión.

**Funcionalidades orientadas al cliente final:**

* **Catálogo de productos online**: Página web pública donde los clientes pueden navegar por el catálogo de productos, filtrarlos por categorías y consultar detalles como ingredientes, precio y alérgenos.
* **Carrito de la compra y sistema de pedidos**: Los clientes pueden añadir productos al carrito, seleccionar dirección de envío y confirmar el pedido mediante un sistema guiado e intuitivo.
* **Gestión de usuario y perfil**: Los usuarios pueden registrarse, iniciar sesión, gestionar sus datos personales y consultar el historial de pedidos.
* **Atención al cliente y soporte**: Se proporciona un canal para contacto directo con la empresa, con posibilidad de realizar consultas o notificar incidencias relacionadas con los pedidos.
* **Información corporativa**: Sección destinada a presentar la historia de la empresa, valores, ubicación, contacto y horarios de apertura.
* **Envío a domicilio**: El sistema permite realizar envíos dentro de la península, con configuración de costes de envío y datos logísticos necesarios para la entrega.

# Módulos del ciclo formativo que intervienen

La aplicación de conocimientos y competencias adquiridas a lo largo del ciclo formativo de **Desarrollo de Aplicaciones Web**. A continuación, se detallan los módulos que han intervenido de forma activa en la construcción del sistema:

* **Desarrollo Web en Entorno Cliente**  
  En este módulo se han aplicado tecnologías como **HTML, CSS, TypeScript y Angular** para construir una interfaz de usuario moderna, responsive y funcional. Se han creado componentes reutilizables, implementado formularios dinámicos, gestión de rutas y comunicación con servicios a través de peticiones HTTP.
* **Desarrollo Web en Entorno Servidor**  
  Se ha desarrollado la lógica de negocio en el servidor utilizando **Java** con el framework **Open Liberty** junto con **MicroProfile**, creando servicios RESTful que permiten la comunicación entre el frontend y el backend. También se han implementado validaciones, gestión de errores y control de sesiones.
* **Bases de Datos**  
  Se ha diseñado y gestionado una base de datos relacional en **MySQL**, incluyendo la definición de tablas, relaciones entre entidades (uno a muchos, muchos a muchos), claves foráneas, restricciones, e índices. Además, se han utilizado herramientas ORM como Hibernate para el mapeo de objetos en el backend.
* **Diseño de Interfaces Web**  
  Se ha trabajado en la usabilidad y estética de la aplicación, utilizando **Tailwind CSS y DaisyUI** para garantizar un diseño visual atractivo, accesible y coherente en todos los módulos. También se han seguido principios de diseño responsive para asegurar una correcta visualización en distintos dispositivos.
* **Programación**  
  Se han aplicado conceptos fundamentales de la programación orientada a objetos, estructuras de control, uso de colecciones, modularidad del código y buenas prácticas en la implementación de la lógica tanto en el cliente como en el servidor.
* **Despliegue de Aplicaciones Web**  
  El sistema se ha desplegado utilizando **Docker**, creando contenedores independientes para el frontend, backend y base de datos. Además, se han realizado configuraciones de red, volúmenes y variables de entorno para garantizar la portabilidad y escalabilidad del sistema.

# Tecnologías elegidas y justificación de dicha elección

Se han seleccionado tecnologías modernas, robustas y ampliamente utilizadas en entornos profesionales, con el objetivo de garantizar un producto eficiente, escalable y fácil de mantener. A continuación, se detallan las tecnologías utilizadas en las distintas capas del sistema, junto con la justificación de su elección:

**Frontend**

* **Angular**  
  Se ha utilizado Angular en una de sus versiones más actuales, con enfoque en standalone components. Esta elección se justifica por su arquitectura modular, su potente sistema de routing y formularios, y su integración natural con TypeScript, lo que permite desarrollar aplicaciones SPA (Single Page Application) de forma organizada y mantenible.
* **Tailwind CSS** + **DaisyUI**  
  Tailwind CSS se ha empleado como framework de utilidades CSS por su eficiencia al generar estilos directamente desde clases en el HTML. DaisyUI, como complemento de Tailwind, proporciona componentes accesibles y estilizados que han facilitado el diseño rápido y uniforme de la interfaz. La combinación permite mantener un diseño moderno, responsive y coherente en toda la aplicación.

**Backend**

* **Java con Open Liberty y MicroProfile**  
  Se ha optado por el uso de Open Liberty junto a MicroProfile por su ligereza, compatibilidad con microservicios y cumplimiento de estándares modernos en el desarrollo de APIs REST. Esta elección permite un desarrollo backend robusto, portable y basado en buenas prácticas del entorno empresarial.
* **Hibernate**  
  Hibernate se ha utilizado como framework ORM para facilitar el mapeo entre objetos Java y la base de datos MySQL, reduciendo la necesidad de escribir consultas SQL complejas y mejorando la abstracción del acceso a datos.
* **JAX-RS**  
  Para la creación de servicios RESTful se ha utilizado JAX-RS, permitiendo definir rutas y controladores de forma estructurada y compatible con MicroProfile.

**Base de datos**

* **MySQL**  
  Se ha elegido MySQL por ser un sistema de gestión de bases de datos relacional ampliamente conocido, fiable, gratuito y con buena integración con herramientas como Hibernate. Ha permitido definir un modelo de datos relacional con claves primarias, foráneas y relaciones entre entidades.

**Herramientas complementarias**

* **Postman**  
  Utilizado durante el desarrollo para probar y depurar los endpoints REST del backend, facilitando la verificación de respuestas, errores y flujos de datos entre cliente y servidor.
* **Swagger**  
  Empleado para documentar automáticamente los servicios REST, facilitando la comprensión, uso y pruebas de las APIs por parte de desarrolladores y usuarios técnicos.
* **Git y GitHub**  
  Git ha sido la herramienta de control de versiones utilizada para gestionar el código fuente del proyecto. GitHub ha permitido mantener el repositorio en la nube, hacer seguimiento de los cambios, colaborar y mantener copias de seguridad del proyecto.
* **Linux (entorno cloud para pruebas)**  
  Para el despliegue en entorno real y las pruebas de funcionamiento en la nube, se ha utilizado un sistema Linux que simula el entorno de producción, permitiendo validar el comportamiento del sistema fuera del entorno local.

# Etapas por las que ha pasado el proyecto

Ha seguido un enfoque estructurado y progresivo, dividido en distintas fases que han permitido construir el sistema de forma ordenada y funcional. A continuación, se detallan las principales etapas por las que ha pasado el proyecto:

1. **Análisis y definición del problema**  
   En esta primera fase se identificaron las necesidades del negocio real, en este caso una confitería tradicional, que requería modernizar su sistema de ventas y gestión. Se definieron los objetivos principales, el público objetivo y los procesos que el sistema debía cubrir.
2. **Diseño de la base de datos y la arquitectura**  
   Se elaboró el modelo de datos relacional en MySQL, incluyendo entidades como productos, usuarios, pedidos, ventas, categorías, alérgenos y stock. Paralelamente, se definió la arquitectura general del proyecto, separando claramente las capas de frontend, backend y base de datos.
3. **Diseño de la interfaz y experiencia de usuario (UI/UX)**  
   Se diseñaron las vistas del usuario final y del administrador, priorizando una navegación intuitiva, visualmente agradable y compatible con distintos dispositivos. Se utilizó Tailwind CSS junto a DaisyUI para garantizar un diseño moderno y uniforme.
4. **Desarrollo del backend**  
   Se implementó el servidor en Java usando Open Liberty y MicroProfile, desarrollando los distintos servicios REST que gestionan la lógica de negocio: autenticación, gestión de productos, pedidos, ventas, stock, etc. También se integró Hibernate para el acceso a datos.
5. **Desarrollo del frontend**  
   Se construyó la interfaz en Angular con componentes standalone, conectando con los servicios REST del backend mediante HTTP. Se implementaron funcionalidades como el carrito, formularios de creación y edición, vistas dinámicas y el panel de administración.
6. **Pruebas y validación**  
   Se realizaron pruebas de los endpoints con Postman, así como validaciones manuales del flujo de uso desde el frontend. Se depuraron errores funcionales y de interfaz, asegurando la estabilidad del sistema en los principales casos de uso.
7. **Documentación del proyecto**  
   Durante y después del desarrollo se generó documentación tanto técnica (usando Swagger para las APIs) como de uso, con el objetivo de facilitar el mantenimiento y la comprensión del sistema por parte de futuros desarrolladores o administradores.
8. **Despliegue en entorno de pruebas (Linux Cloud)**  
   Se subió el proyecto a un servidor con sistema operativo Linux para realizar pruebas de funcionamiento en entorno real. Esto permitió comprobar el comportamiento de la aplicación fuera del entorno local y realizar los últimos ajustes necesarios.
9. **Ajustes finales y mejoras**  
   Tras las pruebas iniciales se pulieron detalles visuales, se optimizaron flujos de trabajo y se preparó el proyecto para su presentación final. También se dejaron planteadas posibles mejoras futuras para ampliar la funcionalidad del sistema.

# Problemas encontrados y soluciones aportadas

Se han presentado diversos problemas técnicos y organizativos que han requerido análisis, adaptación y resolución efectiva. A continuación, se detallan los principales retos encontrados junto con las soluciones aplicadas:

1. **Gestión del estado en Angular con signals**  
   Uno de los principales retos en el frontend fue la implementación de la gestión de estado utilizando signals y computed, una característica reciente de Angular. Al principio surgieron problemas de compatibilidad con formularios reactivos y [(ngModel)].  
   **Solución:** Se adaptó el uso de signals con técnicas de enlace personalizado y valueChanges, manteniendo una arquitectura moderna y funcional.
2. **Comunicación entre componentes y servicios**  
   En componentes más complejos como el TPV o la gestión de pedidos, fue necesario compartir estados y eventos entre múltiples vistas.  
   **Solución:** Se utilizaron servicios inyectables y BehaviorSubject cuando fue necesario, junto a computed y effect, para mantener la sincronización del estado sin perder el rendimiento.
3. **Validaciones en formularios complejos**  
   Formularios como el de creación de pedidos o productos exigían validaciones múltiples (campos obligatorios, formatos, lógica condicional).  
   **Solución:** Se implementaron validaciones personalizadas en Angular y controles de errores en el backend, asegurando integridad y buena experiencia de usuario.
4. **Problemas con relaciones entre entidades en el backend**  
   Las relaciones uno-a-muchos y muchos-a-muchos (por ejemplo, productos con categorías y alérgenos, ventas con líneas de pedido) generaban conflictos en la persistencia de datos y en la serialización JSON.  
   **Solución:** Se utilizó correctamente @JsonManagedReference y @JsonBackReference, junto a una configuración precisa de Hibernate, para evitar ciclos infinitos y asegurar la integridad de los datos.
5. **Pruebas de despliegue en entorno real (Linux Cloud)**  
   Al trasladar el proyecto a un entorno Linux para pruebas, surgieron errores por diferencias en el sistema de archivos, rutas y puertos.  
   **Solución:** Se adaptaron las configuraciones del backend y frontend, y se realizaron pruebas incrementales para asegurar que la aplicación funcionaba correctamente fuera del entorno local.
6. **Tiempo limitado y coordinación de tareas**  
   Como proyecto final, ha sido necesario compaginar su desarrollo con otras asignaturas y prácticas profesionales, lo que supuso una limitación de tiempo.  
   **Solución:** Se estableció una planificación semanal de tareas y objetivos parciales, priorizando funcionalidades clave y manteniendo el enfoque en un MVP (producto mínimo viable) funcional y completo.
7. **Gestión de imágenes y formularios con archivos**  
   La carga de imágenes en productos y perfiles presentaba dificultades para previsualizar, eliminar y gestionar correctamente los archivos desde el navegador.  
   **Solución:** Se implementó un sistema de previsualización y limpieza de archivos con FileReader y eventos personalizados, mejorando la experiencia del usuario en los formularios.

# 7. Nivel de consecución del mismo

Ha alcanzado un grado de desarrollo muy avanzado, cubriendo de forma satisfactoria las funcionalidades principales planteadas desde el inicio. Se ha logrado construir una **aplicación web funcional, usable y adaptada a las necesidades reales de una confitería tradicional**, tanto para la parte de administración como para el cliente final.

**Funcionalidades completadas:**

* Sistema completo de gestión de productos, categorías y alérgenos.
* Creación, edición y consulta de pedidos, ventas y usuarios.
* Panel de administración con acceso a estadísticas y datos clave del negocio.
* Interfaz de TPV para ventas en tienda física.
* Control de stock y movimientos de almacén.
* Sistema de roles y autenticación de usuarios.
* Diseño responsive y moderno usando Tailwind CSS y DaisyUI.
* Integración completa entre frontend y backend mediante servicios REST.
* Pruebas funcionales realizadas en entorno local y servidor Linux en la nube.

**Funcionalidades parcialmente implementadas o pendientes:**

* Envío de notificaciones automáticas (por ejemplo, por cambio de estado del pedido o stock bajo).
* Integración con pasarelas de pago para compras en línea.
* Gestión avanzada de direcciones y métodos de pago por parte del usuario.
* Panel público completo con catálogo, carrito y experiencia de compra final para el cliente.

Aunque el núcleo funcional del sistema está plenamente operativo, aún hay espacio para ampliar y mejorar algunas funcionalidades orientadas a la experiencia del usuario final y a la automatización de ciertos procesos.

# Recursos necesarios para su implementación y presupuesto económico

Se requieren una serie de recursos técnicos, humanos y económicos. A continuación, se describen los elementos esenciales y una estimación aproximada del presupuesto necesario para su implementación real en un entorno empresarial.

**Recursos técnicos:**

* **Servidor en la nube** (VPS o instancia Linux): necesario para alojar la base de datos y el backend.  
  *Ejemplo: 1 vCPU, 2 GB RAM, 40 GB SSD*  
  **Coste aproximado:** 10–15 €/mes
* **Dominio web** (.com, .es, etc.): para hacer accesible la web al público.  
  **Coste aproximado:** 10–15 €/año
* **Certificado SSL**: para asegurar la conexión (puede ser gratuito con Let's Encrypt).  
  **Coste aproximado:** 0–15 €/año
* **Frontend (Angular)** y **backend (Java con Open Liberty)**: desarrollados como software propio, sin coste por licencias.
* **Base de datos MySQL**: sistema de código abierto y gratuito.
* **Ordenador para la gestión**: se recomienda disponer de un equipo con al menos 8 GB de RAM, navegador actualizado y conexión estable a Internet.

**Recursos humanos:**

* **Desarrollador web**: responsable del análisis, diseño, implementación y despliegue de la aplicación.
* **Responsable del negocio**: encargado de proporcionar la información funcional, validar el sistema y gestionar el uso diario de la plataforma.

*En este caso, el desarrollo ha sido realizado por un único estudiante como proyecto final del ciclo formativo, lo que ha supuesto un importante ahorro económico respecto a una solución comercial.*

**Presupuesto estimado de implementación:**

| **Recurso** | **Coste aproximado** |
| --- | --- |
| Dominio web | 12 €/año |
| Servidor VPS básico | 12 €/mes (144 €/año) |
| Certificado SSL (Let’s Encrypt) | 0 € |
| Servicios adicionales o correo empresarial | 5–10 €/mes (opcional) |
| **Total estimado (primer año)** | **150–200 €** |

Este presupuesto incluye únicamente los costes básicos para el funcionamiento técnico del sistema. En caso de ampliar funcionalidades (pasarelas de pago, envío automatizado, CRM, etc.) o contratar personal técnico, el coste podría incrementarse.

# Criterios de calidad

Se han seguido una serie de criterios de calidad orientados a garantizar la estabilidad, mantenibilidad, usabilidad y accesibilidad del sistema. Estos criterios han sido aplicados tanto en el desarrollo técnico como en el diseño de la experiencia de usuario.

**1. Calidad del código y buenas prácticas**

* Se ha aplicado **programación modular y estructurada**, separando responsabilidades en componentes, servicios y modelos.
* El código ha sido documentado mediante comentarios claros y descriptivos, facilitando su comprensión y mantenimiento.
* Se ha seguido una **nomenclatura coherente** y se han aplicado principios como DRY (Don't Repeat Yourself) y KISS (Keep It Simple).
* Uso adecuado de interfaces, DTOs y tipado estricto en TypeScript para evitar errores en tiempo de desarrollo.

**2. Documentación técnica**

* La API REST ha sido documentada utilizando **Swagger**, permitiendo consultar de forma clara los endpoints disponibles, parámetros requeridos y respuestas esperadas.
* Se ha elaborado documentación adicional para facilitar el despliegue, las pruebas y la utilización del sistema por parte de futuros desarrolladores o administradores.

**3. Pruebas y verificación**

* Se han realizado pruebas funcionales de todos los flujos principales: creación y edición de productos, gestión de pedidos, ventas, usuarios y navegación en el frontend.
* Uso de **Postman** para probar y validar los endpoints del backend de forma individual y en conjunto.
* Pruebas en entorno Linux para validar la estabilidad del sistema fuera del entorno de desarrollo local.

**4. Diseño y usabilidad**

* La interfaz ha sido diseñada con **Tailwind CSS y DaisyUI**, garantizando un diseño limpio, moderno y visualmente coherente.
* La navegación es clara e intuitiva, con menús laterales, botones accesibles y vistas bien organizadas.
* Se han empleado elementos visuales y formularios validados para guiar al usuario en su experiencia dentro de la aplicación.

**5. Accesibilidad y compatibilidad**

* El diseño es **responsive**, adaptándose correctamente a distintos tamaños de pantalla (ordenadores, tablets, móviles).
* Se han seguido principios básicos de accesibilidad, como contraste suficiente, uso de etiquetas y navegación mediante teclado.
* Compatibilidad garantizada con los principales navegadores modernos (Chrome, Firefox, Edge).

**6. Seguridad**

* Se han implementado controles de acceso y autenticación de usuarios mediante roles, protegiendo el acceso a las funcionalidades administrativas.
* Validaciones tanto en el frontend como en el backend para evitar el envío de datos erróneos o maliciosos.

# Mejoras que se podrían realizar en el futuro

Ha sido diseñado con una estructura modular y escalable, lo que permite su evolución y ampliación mediante nuevas funcionalidades. A continuación se describen una serie de mejoras que podrían implementarse en futuras versiones del sistema para optimizar la experiencia de usuario, la gestión interna y la escalabilidad del negocio:

**Mejoras para el cliente final**

* **Integración con pasarelas de pago**: permitir pagos en línea mediante servicios como Stripe o PayPal.
* **Gestión avanzada de direcciones y métodos de pago**: posibilidad de almacenar múltiples direcciones y formas de pago por usuario.
* **Notificaciones automáticas**: envío de correos o alertas cuando un pedido cambia de estado o es enviado.
* **Sistema de opiniones y valoraciones**: permitir que los usuarios valoren productos y dejen reseñas.
* **Sistema de alérgenos personalizado por usuario**: cada usuario podría configurar sus alérgenos y la web le mostraría alertas sobre los productos que puede o no consumir.
* **Chat en directo**: atención personalizada a través de un sistema de mensajería directa en la web.
* **Seguimiento en directo de pedidos**: el cliente podría visualizar en tiempo real el estado y la ubicación de su pedido.
* **Soporte multidioma y accesibilidad mejorada**: adaptar la web para distintos idiomas y cumplir normativas WCAG.

**Mejoras para la gestión interna**

* **Sistema de gestión de productos activos/inactivos**: opción para activar o desactivar productos en función de su disponibilidad para la venta.
* **Gestión avanzada de proveedores**: sistema completo para controlar contactos, pedidos de reposición, historial de compras y condiciones de suministro.
* **Sistema de impresión de facturas**: generación automática de facturas en PDF tanto para la tienda como para el cliente.
* **Seguimiento del proceso del pedido desde administración**: permitir al personal de tienda actualizar y monitorizar cada etapa del pedido (preparación, empaquetado, envío).
* **Trazabilidad del producto**: registro completo de cada producto desde su entrada en almacén hasta su venta, incluyendo movimientos, lote y fecha.
* **Panel de configuración global**: permitir al administrador modificar desde la propia aplicación aspectos como gastos de envío, horarios, mensajes personalizados, etc.
* **Sistema de promociones, descuentos y cupones**: ofrecer ofertas temporales o descuentos personalizados para fidelizar clientes.
* **Panel estadístico avanzado**: gráficos interactivos, exportación de datos, comparativas de ventas por periodos y análisis de comportamiento del cliente.
* **Integración con sistemas de logística externa**: automatizar el proceso de envío y seguimiento mediante empresas de reparto.

**Otras posibles mejoras**

* **Aplicación móvil (PWA o nativa)**: para clientes o empleados, con funcionalidades adaptadas y sincronización en tiempo real.
* **Integración con sistemas ERP o contables**: conectar con software empresarial para llevar contabilidad, nóminas o facturación de forma automatizada.

# Bibliografía

**Documentación oficial**

* Angular. *Angular Documentation*. Disponible en: https://angular.io/docs
* Tailwind CSS. *Tailwind CSS Documentation*. Disponible en: <https://tailwindcss.com/docs>
* DaisyUI. *DaisyUI Documentation*. Disponible en: <https://daisyui.com>
* Open Liberty. *Official Open Liberty Guides*. Disponible en: https://openliberty.io/guides/
* MicroProfile. *Eclipse MicroProfile Specification*. Disponible en: <https://microprofile.io/>
* Hibernate ORM. *Hibernate Documentation*. Disponible en: https://hibernate.org/orm/documentation/
* MySQL. *MySQL Reference Manual*. Disponible en: <https://dev.mysql.com/doc/>

**Herramientas utilizadas**

* Postman. *Postman Learning Center*. Disponible en: https://learning.postman.com
* Swagger. *Swagger Tools & Docs*. Disponible en: https://swagger.io/tools/
* Git. *Git Documentation*. Disponible en: <https://git-scm.com/doc>
* GitHub. *GitHub Docs*. Disponible en: <https://docs.github.com/>
* Visual Studio Code. *VS Code Docs*. Disponible en: <https://code.visualstudio.com/docs>

**Otros recursos**

* Stack Overflow. Consultas técnicas puntuales. Disponible en: <https://stackoverflow.com/>
* MDN Web Docs. *HTML, CSS y JavaScript documentation*. Disponible en: <https://developer.mozilla.org/>