

I.E.S LAS SALINAS



PROYECTO FINAL DE GRADO

CURSO 24-25

PAGEZY

CICLO FORMATIVO GRADO SUPERIOR

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

AUTOR: Rubén Cereceda García

TUTOR: Manuel Antonio Benito Mora

Resumen

Pagezy es una plataforma digital que permite a cualquier usuario crear y gestionar su propia tienda online de manera sencilla, rápida y sin conocimientos técnicos avanzados. El sistema proporciona un entorno personalizado en el que cada tienda tiene su propio panel de administración, catálogo de productos, sistema de pedidos y diseño adaptable a las necesidades del negocio.

Esta solución ha sido desarrollada utilizando tecnologías como Python (Flask), bases de datos SQLite y una interfaz moderna basada en HTML, CSS y JavaScript. El objetivo es democratizar el comercio electrónico, facilitando a pequeños emprendedores y comercios locales el acceso a herramientas profesionales para vender por Internet sin depender de plataformas externas.

El proyecto aborda aspectos técnicos como el diseño modular de la aplicación, la escalabilidad del sistema y la gestión dinámica de múltiples bases de datos, una por cada tienda. Además, se ha puesto especial énfasis en la experiencia de usuario tanto en el área pública como en el área administrativa, permitiendo un manejo intuitivo del sistema.

Pagezy representa una oportunidad real para acercar la tecnología a quienes más lo necesitan, ofreciendo una solución accesible, escalable y personalizable.

Abstract

Pagezy is a digital platform that enables any user to easily create and manage their own online store without requiring advanced technical knowledge. The system offers a customizable environment where each store has its own admin panel, product catalogue, order system, and flexible design tailored to business needs.

This solution has been developed using technologies such as Python (Flask), SQLite databases, and a modern interface built with HTML, CSS, and JavaScript. The aim is to democratize e-commerce by empowering small businesses and local entrepreneurs with professional tools to sell online without relying on third-party platforms.

The project covers technical aspects such as modular application design, system scalability, and dynamic management of individual store databases. Special attention has been given to the user experience, both in the public interface and in the admin panel, ensuring intuitive system operation.

Pagezy stands as a real opportunity to bring technology closer to those who need it most, offering an accessible, scalable, and customizable e-commerce solution.

Contenido

1. Justificación.	Página 5
2. Introducción	Página 5
3. Objetivos	Página 6
4. Desarrollo	Página 7
4.1. Tecnologías y herramientas utilizadas	Página 7
4.2. Metodología del desarrollo	Página 7
4.3. Planificación del proyecto	Página 8
4.4. Desarrollo de la aplicación	Página 8
a. Estructura del proyecto	Página 9
b. Funcionalidades principales	Página 10
c. Diseño de la interfaz	Página 13
d. Base de datos	Página 16
e. Pruebas y despliegue	Página 17
5. Viabilidad empresarial	Página 18
5.1. Viabilidad técnica	Página 18
5.2. Viabilidad económica	Página 19
5.3. Viabilidad temporal	Página 19
5.4. Viabilidad de escalabilidad	Página 19
5.5. Viabilidad funcional y de retorno	Página 19
5.6. Análisis DAFO	Página 20
6. Conclusiones	Página 21
7. Bibliografía	Página 22
8. Anexo	Página 23
8.1. Fragmentos de código relevantes	Página 23

Figuras

Figura 1. Estructura de carpetas del proyecto Pagezy

Figura 2. Panel de administración de productos

Figura 3. Carrito de compra público y resumen de pedido

Figura 4. Vista general del panel de administración

Figura 5. Barra de filtrado y búsqueda activa

Figura 6. Gestión de pedidos en el panel admin

Figura 7. Ejemplo de plantillas visuales

Figura 8. Vista de tienda pública con productos

Figura 9. Selector de plantilla y configuraciones visuales

Figura 10. Panel administrativo con navegación por secciones

Figura 11. Modal de confirmación de pedido y controles del carrito

Figura 12. Código de los modelos de las tablas de SQLite

Figura 13. Ejemplo de prueba de flujo de registro

Figura 14. Análisis DAFO

Figura 15. Código de ver_productos

Figura 16. Código de creación de pedidos

Figura 17. Código de filtros de productos

Figura 18. Código de mostrar u ocultar pestañas

1.- Justificación

En la actualidad, el comercio electrónico ha crecido de manera exponencial, impulsado por la digitalización de los hábitos de consumo y la necesidad de adaptar los modelos de negocio a las nuevas tecnologías. Sin embargo, una gran parte de los pequeños comercios, emprendedores y negocios locales sigue enfrentándose a importantes barreras para acceder al mercado digital, debido a la complejidad técnica, los altos costes de las plataformas comerciales o la falta de conocimientos especializados.

Pagezy nace como respuesta a esta problemática. Se trata de una plataforma diseñada para eliminar las barreras técnicas y económicas, ofreciendo una solución sencilla, accesible y eficiente para que cualquier persona, sin necesidad de saber programar, pueda crear y gestionar su propia tienda online. La justificación del proyecto se fundamenta en la necesidad real y creciente de democratizar el acceso a herramientas digitales de comercio, facilitando la transformación digital del tejido empresarial más vulnerable: autónomos, pymes y comerciantes tradicionales.

A través de un sistema modular y adaptable, Pagezy permite generar tiendas personalizadas, cada una con su propia base de datos, panel de administración, catálogo y diseño visual. Esta autonomía individual para cada comercio permite escalar y gestionar cada tienda como una instancia independiente, lo que garantiza flexibilidad, rendimiento y control total por parte del comerciante.

La elección de tecnologías como Flask (Python), SQLite y una arquitectura orientada a micro servicios permite además mantener un desarrollo ágil, seguro y escalable, lo cual refuerza la viabilidad técnica del proyecto. En definitiva, Pagezy se justifica como una herramienta que busca empoderar a quienes tradicionalmente han quedado al margen del comercio electrónico, proporcionando una alternativa real y sostenible para competir en el entorno digital.

2.- Introducción

El auge del comercio electrónico ha transformado radicalmente la forma en que las personas compran y venden productos en todo el mundo. En este contexto, muchas empresas han encontrado en Internet una vía de crecimiento y expansión, pero también ha surgido una brecha tecnológica que deja fuera a pequeñas y medianas empresas, emprendedores o comerciantes sin conocimientos técnicos ni recursos para acceder a plataformas profesionales.

Pagezy surge como una solución a esta problemática. Su propósito principal es ofrecer una plataforma intuitiva, ligera y personalizable que permita a cualquier usuario crear su tienda online en pocos pasos y sin necesidad de experiencia en desarrollo web. A diferencia de otras soluciones del mercado, Pagezy proporciona un entorno completamente autónomo por tienda, con su propia base de datos y panel administrativo, lo que garantiza independencia y control para cada comerciante.

La iniciativa se ha desarrollado como un proyecto modular y escalable, implementado con tecnologías de código abierto, enfocándose en ofrecer funcionalidades clave como gestión de productos, pedidos, clientes, plantillas visuales y carrito de compra dinámico. También se han contemplado aspectos como la seguridad de los datos, la experiencia de usuario y la adaptabilidad a distintos tipos de negocio.

Esta memoria tiene como objetivo documentar el desarrollo del proyecto Pagezy, desde su concepción hasta su implementación técnica, pasando por las decisiones de diseño, estructura del sistema, funcionalidades desarrolladas y análisis de su viabilidad. A lo largo del documento se describen tanto los fundamentos teóricos como los aspectos prácticos de la plataforma, con el fin de evidenciar su utilidad real y su potencial impacto en el ecosistema digital de pequeños comercios.

3.- Objetivos

El desarrollo de Pagezy tiene como finalidad principal brindar una herramienta accesible y funcional para la creación de tiendas online personalizadas, orientada especialmente a pequeños comercios y emprendedores sin experiencia técnica. Para ello, se han definido los siguientes objetivos:

- **Objetivo general**

Desarrollar una plataforma web que permita a cualquier usuario crear y gestionar su propia tienda online de manera autónoma, sencilla y eficiente, sin necesidad de conocimientos en programación.

- **Objetivos específicos**
- Diseñar un sistema modular que permita la creación dinámica de tiendas independientes, cada una con su base de datos y configuración visual personalizada.
- Implementar un panel de administración intuitivo para la gestión de productos, pedidos, clientes y configuraciones de la tienda.
- Desarrollar un sistema de catálogo y carrito de compras dinámico accesible desde la parte pública de la tienda.
- Permitir la personalización visual mediante plantillas adaptables, colores, logotipo y estilo general.
- Utilizar tecnologías de código abierto (Python, Flask, SQLite, HTML, CSS, JS) que permitan mantener la escalabilidad y sostenibilidad del proyecto.
- Garantizar una experiencia de usuario fluida tanto en la parte pública como en la administrativa.
- Desarrollar mecanismos de autenticación de clientes y control de pedidos.
- Facilitar el despliegue de nuevas tiendas a través de una arquitectura que permita la creación y conexión de bases de datos independientes.

4.- Desarrollo

El desarrollo de Pagezy ha seguido una estructura modular que permite su escalabilidad y adaptabilidad. Desde su concepción, se buscó implementar un sistema funcional, mantenible y fácil de usar tanto para los administradores como para los usuarios finales.

El proceso de desarrollo se dividió en varias fases que incluyen la elección de tecnologías, la definición de arquitectura, la implementación de funcionalidades y la validación del producto final mediante pruebas y despliegue.

En las siguientes secciones se describen en detalle las decisiones técnicas tomadas, la planificación del proyecto, su estructura, así como los elementos clave que componen la aplicación.

4.1.- Tecnologías y herramientas utilizadas

Para el desarrollo de Pagezy se optó por tecnologías modernas, ligeras y de código abierto que permiten una implementación ágil, flexible y económicamente viable. A continuación, se detallan las principales herramientas y tecnologías utilizadas:

- Python (Flask): Framework de desarrollo web utilizado para la lógica de servidor, rutas, controladores y conexión con bases de datos. Su simplicidad y robustez lo convierten en una herramienta ideal para el desarrollo rápido de aplicaciones web escalables.
- SQLite: Sistema de gestión de bases de datos relacional, elegido por su facilidad de uso, ligereza y por no requerir instalación adicional. Cada tienda cuenta con su propia base de datos SQLite.
- HTML5 y CSS3 (SCSS): Para la construcción de la interfaz visual tanto del panel administrativo como del sitio público de cada tienda. Se utilizaron estilos personalizados por plantilla.
- JavaScript: Usado para gestionar interacciones dinámicas como el carrito de compra, filtros de productos, navegación sin recarga, y validaciones básicas en el cliente.
- Jinja2: Motor de plantillas de Flask que permite renderizar contenido dinámico en los archivos HTML.
- SQLAlchemy: Utilizado de manera indirecta para conectarse dinámicamente a las bases de datos SQLite y manipular sus tablas sin definir modelos ORM explícitos.
- VS Code: Entorno de desarrollo integrado (IDE) principal usado durante la programación.
- Git y GitHub: Para el control de versiones y el seguimiento del desarrollo del proyecto.

4.2.- Metodología del desarrollo

Para el desarrollo de Pagezy se ha optado por una metodología ágil, basada en iteraciones cortas y entregas parciales funcionales. Aunque no se ha seguido un marco rígido como Scrum o Kanban, se han aplicado principios ágiles para mantener la flexibilidad, responder a cambios y mejorar continuamente el producto conforme se iban validando avances.

El desarrollo se dividió en sprints semanales no formalizados, donde cada bloque funcional era implementado, probado y refinado antes de pasar al siguiente. Esto permitió incorporar mejoras de manera continua, probar nuevas ideas y adaptar funcionalidades a medida que se obtenía retroalimentación durante el proceso.

Durante el ciclo de desarrollo se trabajó en paralelo en distintas áreas del sistema: lógica de negocio en Flask, diseño de interfaz, manejo de base de datos, y funciones del panel administrativo. Esta estrategia permitió mantener una visión integral del sistema y asegurar la coherencia entre módulos.

Además, se hizo uso de control de versiones con Git, lo que facilitó el seguimiento de cambios, la colaboración modular y la documentación del progreso del proyecto.

4.3.- Planificación del proyecto

El desarrollo de Pagezy se llevó a cabo en diferentes fases planificadas con base en prioridades funcionales y dependencias técnicas. A continuación, se resume la planificación seguida:

- Fase de análisis y definición de requisitos: Identificación de las funcionalidades necesarias para una tienda online básica, tanto desde la perspectiva del administrador como del cliente.
- Diseño de la arquitectura técnica: Decisión sobre el uso de una base de datos independiente por tienda, el modelo de rutas en Flask, y la estructura de carpetas para mantener el proyecto modular.
- Desarrollo del núcleo funcional: Implementación del sistema de creación y gestión de productos, pedidos, clientes y configuración visual desde el panel administrativo.
- Desarrollo del sistema de plantillas y parte pública: Creación de interfaces visuales diferenciadas para las tiendas, con navegación pública, vista de productos y carrito de compra.
- Integración de funciones dinámicas con JavaScript: Aplicación de filtros, buscadores, carrito de compra, modales de confirmación y formularios dinámicos.
- Pruebas de funcionalidad y depuración: Validación del sistema con distintos escenarios, corrección de errores e implementación de mejoras de usabilidad.
- Documentación y empaquetado del proyecto: Preparación de la documentación técnica, guías de uso y pruebas finales para entrega del proyecto.

4.4.- Desarrollo de la aplicación

La aplicación Pagezy se ha desarrollado con un enfoque modular y una arquitectura clara, permitiendo mantener el orden y la escalabilidad a medida que se añaden nuevas funcionalidades. Esta sección describe la estructura del proyecto, sus principales componentes, funcionalidades clave, y su integración en una plataforma completa de comercio electrónico.

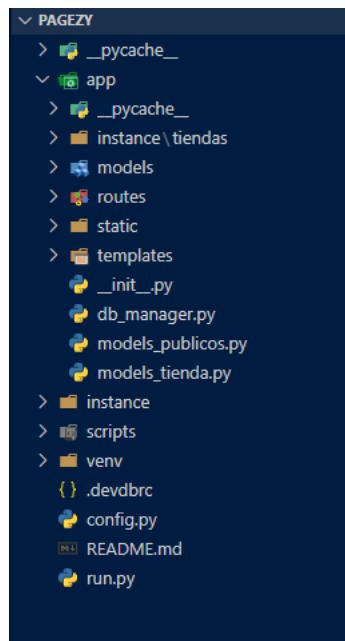
a.- Estructura del proyecto

El proyecto está organizado en una arquitectura típica de aplicaciones Flask, en la que cada funcionalidad se encuentra separada en módulos para facilitar su mantenimiento. La carpeta principal `app/` contiene los elementos esenciales que conforman el sistema:

- `__init__.py`: Inicializa la aplicación Flask, configura rutas globales y carga los Blueprints.
- `db_manager.py`: Lógica de conexión y gestión dinámica de bases de datos SQLite individuales por tienda.
- `models_publicos.py`: Define los modelos ORM relacionados con la gestión de usuarios, tiendas en creación y autenticación.
- `models_tienda.py`: Contiene modelos para representar tablas específicas de cada tienda como productos, pedidos y clientes.
- `routes/`: Contiene los Blueprints divididos por contexto (público, tienda, admin).
- `templates/`: Plantillas HTML organizadas por carpeta y tipo (tiendas públicas, panel de administración, componentes).
- `static/`: Archivos estáticos de cada tienda como imágenes, CSS y JS.

La siguiente imagen muestra la estructura general del directorio del proyecto Pagezy:

Figura 1. Estructura de carpetas del proyecto Pagezy












b.- Funcionalidades principales

- Gestión de productos

Desde el panel de administración, los comerciantes pueden editar o eliminar productos. Cada producto contiene información como nombre, descripción, precio, imagen, stock y categoría. También se puede habilitar o deshabilitar su visibilidad pública.

Figura 2. Panel de administración de productos


Productos									
id	nombre	descripcion	precio	imagen_url	stock	categoria	tipo	caladas	Acciones
2	Blue Razz		12.99		20	Super	Vaper	15000	Eliminar
3	Blueberry Blackcurrant Ice		12.99		20	Super	Vaper	15000	Eliminar
4	Blueberry Cherry Cola		12.99		20	Super	Vaper	15000	Eliminar
5	Blueberry Cherry Cranberry		12.99		20	Super	Vaper	15000	Eliminar
6	Blueberry Ice		12.99		20	Super	Vaper	15000	Eliminar
7	Double Apple		12.99		20	Super	Vaper	15000	Eliminar
8	Lemon Lime		12.99		20	Super	Vaper	15000	Eliminar
9	Mixed Berry		12.99		20	Super	Vaper	15000	Eliminar
10	Peach Mango Pineapple		12.99		20	Super	Vaper	15000	Eliminar

- Carrito de compra y pedidos

En la vista pública, los usuarios pueden explorar los productos disponibles y añadirlos a un carrito de compra dinámico. Este carrito es gestionado desde el navegador mediante JavaScript, permitiendo actualizar cantidades, eliminar productos y confirmar el pedido sin necesidad de recargar la página.

Figura 3. Carrito de compra público y resumen de pedido

Tu carrito




Strawberry Banana

12.99 €

-

1

+




The Elder Scrolls IV: Oblivion Remastered

79.99 €

-

1

+



Red Bull

12.99 €

-

1

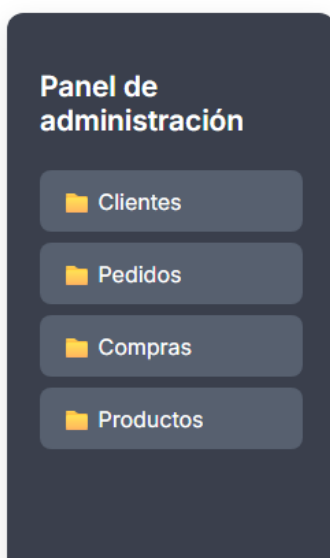
+

Finalizar compra

- Panel de administración personalizado

Cada tienda dispone de su propio panel de administración al que se accede mediante autenticación. Desde este entorno, el usuario puede gestionar clientes, pedidos y configuración visual de la tienda.

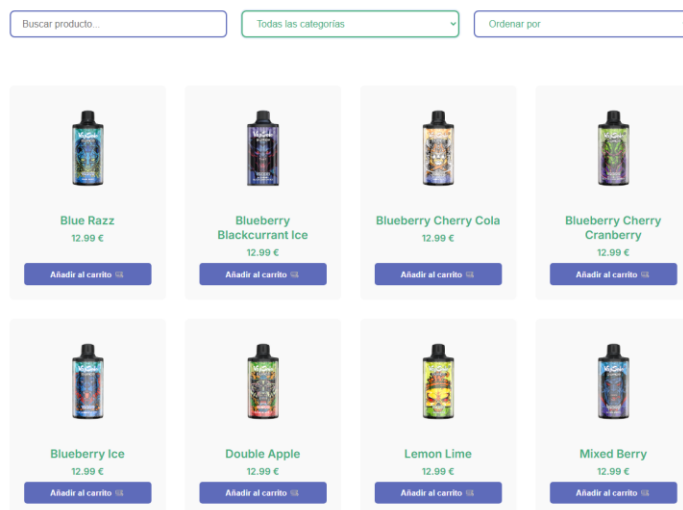
Figura 4. Vista general del panel de administración



- Filtrado y búsqueda de productos

Los productos pueden ser filtrados por nombre, categoría y precio, todo ello mediante una interfaz dinámica con filtros aplicados al vuelo sin recargar la página. Esta funcionalidad mejora notablemente la experiencia de usuario.

Figura 5. Barra de filtrado y búsqueda activa



- Gestión de pedidos

Los administradores pueden consultar los pedidos realizados, cambiar su estado (pendiente, enviado, completado), o eliminarlos. También se relacionan con los datos de cliente y los productos adquiridos.

Figura 6. Gestión de pedidos en el panel admin

id	cliente_id	fecha	estado	total	Actualizar estado	Acciones
1	1	None	Entregado	172.97	<input type="text" value="Cambiar estado"/>	Eliminar
2	1	None	pendiente	159.98	<input type="text" value="Cambiar estado"/>	Eliminar
3	1	None	pendiente	12.99	<input type="text" value="Cambiar estado"/>	Eliminar
5	2	None	Enviado	38.97	<input type="text" value="Cambiar estado"/>	Eliminar
6	1	None	pendiente	64.95	<input type="text" value="Cambiar estado"/>	Eliminar

- Personalización visual

Pagezy permite elegir entre diferentes plantillas visuales para la tienda, además de permitir modificar colores, logos y contenidos personalizados. Esto facilita que cada tienda tenga una identidad propia.

Figura 7. Ejemplo de plantillas visuales



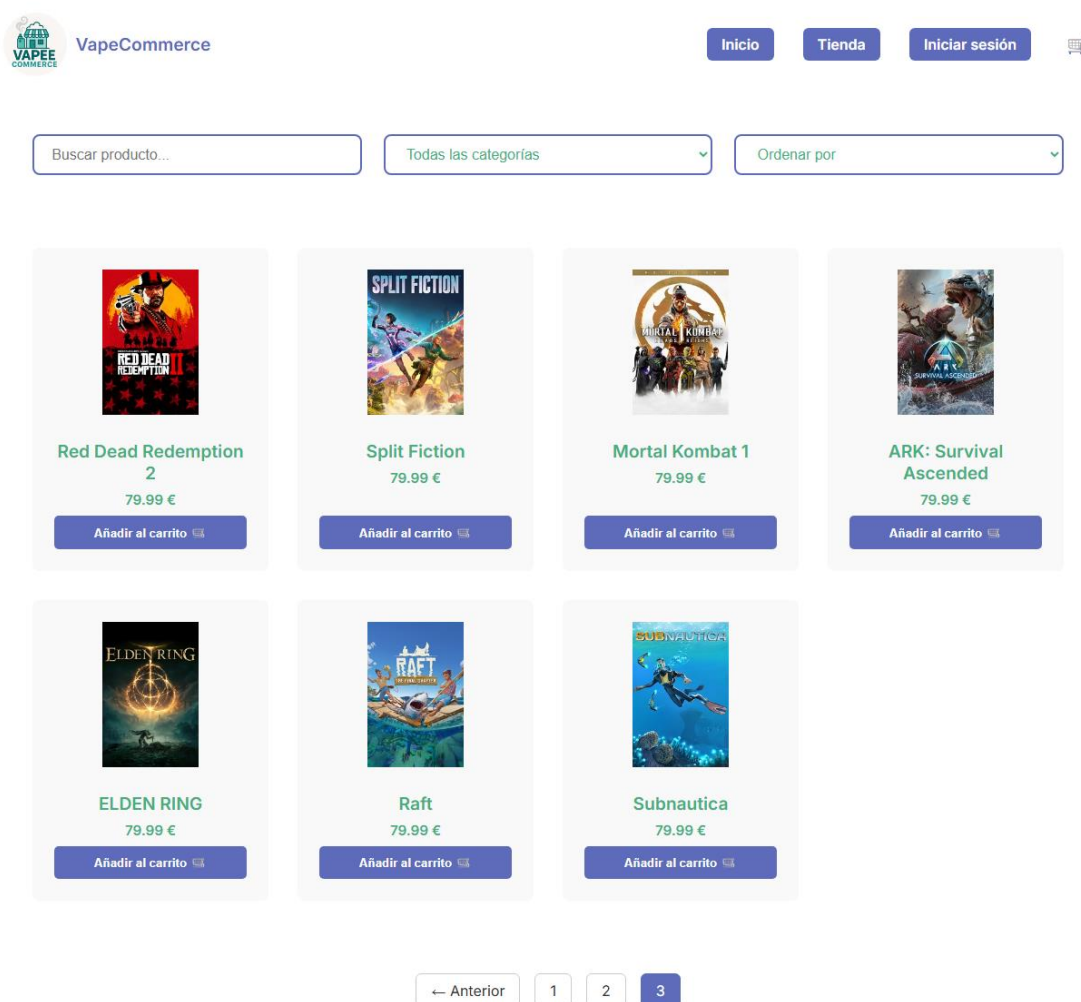
c.- Diseño de la interfaz

La interfaz de usuario es un aspecto fundamental en Pagezy, tanto en su área pública como en el panel de administración. Se ha puesto especial énfasis en crear un entorno amigable, claro y funcional que no requiera conocimientos técnicos por parte del usuario final. Se han utilizado principios de diseño responsive y estructuras limpias para garantizar una experiencia fluida desde cualquier dispositivo.

- Interfaz pública de tienda

Cada tienda cuenta con una interfaz pública que se adapta visualmente a través de plantillas prediseñadas. Estas plantillas definen el layout, los colores y la disposición de los productos, buscando siempre facilitar la navegación y mejorar la conversión. El usuario puede buscar productos, filtrarlos, ver sus detalles y añadirlos al carrito.

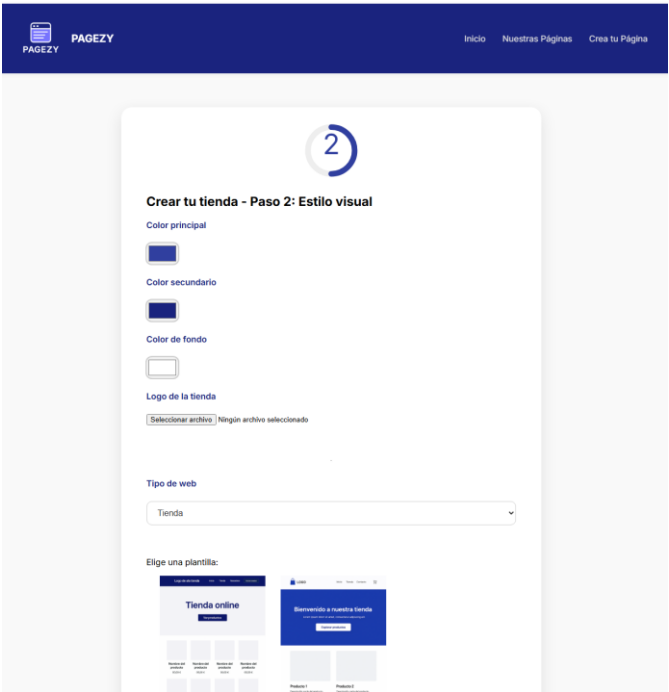
Figura 8. Vista de tienda pública con productos



- Personalización de plantilla

Pagezy permite seleccionar entre distintas plantillas visuales y configurar colores, logotipo y estilo. Esto se gestiona desde el panel de administración, permitiendo a cada comercio establecer su identidad visual sin conocimientos de diseño web.

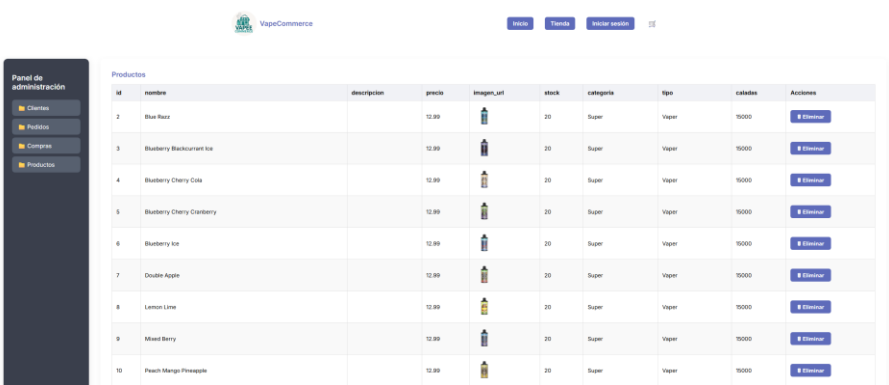
Figura 9. Selector de plantilla y configuraciones visuales



- Panel de administración

La interfaz del panel de administración está organizada en pestañas laterales y secciones visuales. Se ha diseñado con colores neutros y jerarquía clara, facilitando el acceso a funciones clave como productos, clientes, pedidos y personalización. También se ha priorizado la responsividad para permitir su uso desde dispositivos móviles o tablets.

Figura 10. Panel administrativo con navegación por secciones




- Componentes interactivos


Elementos como el carrito, los filtros dinámicos, los modales de pedido o los botones de acción han sido implementados con JavaScript nativo, garantizando ligereza y compatibilidad. Estos componentes permiten una experiencia fluida y moderna para el usuario final.

Figura 11. Modal de confirmación de pedido y controles del carrito

Resumen del pedido ✕



Strawberry Banana
12.99 €
- 1 +



The Elder Scrolls IV: Oblivion Remastered
79.99 €
- 1 +

Total: 92.98 €

Teléfono:

Dirección:

Método de pago:

☒ Efectivo contrareembolso

☐ Tarjeta al transportista

Confirmar pedido

d.- Base de datos

Uno de los aspectos clave en el desarrollo de Pagezy ha sido la decisión de utilizar una base de datos independiente por tienda. Esto significa que cada tienda creada en la plataforma tiene su propio archivo `.sqlite`, lo que garantiza aislamiento de datos, mayor seguridad, flexibilidad y una administración completamente autónoma.

- Arquitectura de la base de datos

Cada base de datos contiene un conjunto de tablas fundamentales, entre las que se encuentran:

- `producto`: información de productos, como nombre, precio, imagen, stock y categoría.
- `cliente`: datos personales de los clientes que realizan pedidos.
- `pedido`: registro de pedidos realizados, su estado y cliente asociado.
- `compra`: relación entre pedido y productos adquiridos, con cantidades.
- `configuracion_visual`: configuraciones de colores, logo y plantilla visual.

Estas tablas son gestionadas de forma dinámica mediante SQLAlchemy a través de `MetaData.reflect()`, lo que permite operar con diferentes bases de datos sin modelos ORM rígidos, adaptándose a cada tienda en tiempo real.

Figura 12. Código de los modelos de las tablas de SQLite

```
class Producto(db.Model):
    __tablename__ = "producto"
    id = db.Column(db.Integer, primary_key=True)
    nombre = db.Column(db.String(100), nullable=False)
    descripcion = db.Column(db.Text)
    precio = db.Column(db.Float, nullable=False)
    imagen_url = db.Column(db.String(255))
    stock = db.Column(db.Integer, default=0)
    categoria = db.Column(db.String(50))
    tipo = db.Column(db.String(50))
    caladas = db.Column(db.Integer)

class Cliente(db.Model):
    __tablename__ = "cliente"
    id = db.Column(db.Integer, primary_key=True)
    nombre = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    telefono = db.Column(db.String(20))
    direccion = db.Column(db.String(255))
    password_hash = db.Column(db.String(255), nullable=False) # Contraseña hasheada

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)

class Administrador(db.Model):
    __tablename__ = "administrador"
    id = db.Column(db.Integer, primary_key=True)
    usuario = db.Column(db.String(50), unique=True, nullable=False)
    contraseña = db.Column(db.String(255), nullable=False)
    email = db.Column(db.String(120), unique=True)
```

- Ventajas del modelo por tienda
- Escalabilidad: es posible administrar muchas tiendas sin que una interfiera con otra.
- Seguridad: el aislamiento de datos reduce el riesgo de accesos no autorizados entre tiendas.
- Simplicidad de despliegue: al tratarse de archivos `.sqlite`, las tiendas pueden gestionarse fácilmente sin necesidad de una base de datos centralizada compleja.
- Mantenimiento individualizado: permite realizar copias de seguridad, restauraciones o migraciones de manera específica por tienda.

e.- Pruebas y despliegue

Durante el desarrollo de Pagezy, se llevaron a cabo distintas fases de prueba para asegurar el correcto funcionamiento del sistema, tanto a nivel funcional como de experiencia de usuario. Estas pruebas permitieron identificar errores, validar flujos de interacción y mejorar aspectos clave del rendimiento y usabilidad.

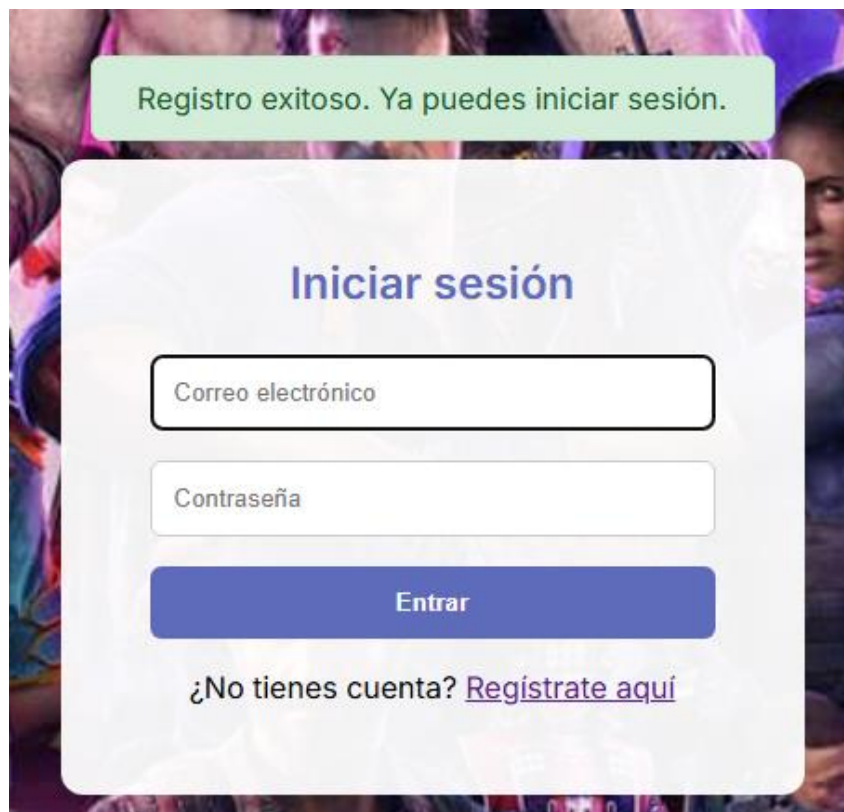
- Pruebas funcionales

Se realizaron pruebas manuales sobre todas las funcionalidades principales:

- Gestión de productos (alta, edición, eliminación).
- Flujo completo del carrito y confirmación de pedidos.
- Registro e inicio de sesión de clientes y administradores.
- Visualización pública de productos, búsqueda y filtrado dinámico.
- Administración de pedidos y actualización de estado.
- Gestión de base de datos independiente por tienda.

Las pruebas se realizaron en distintos navegadores (Chrome, Firefox, Edge) y dispositivos (escritorio y móvil) para garantizar la compatibilidad.

Figura 13. Ejemplo de prueba de flujo de registro



- Despliegue local y despliegue en servidor

Inicialmente, **Pagezy** fue ejecutado en entorno local utilizando el servidor de desarrollo de **Flask**. Esto permitió una implementación rápida y controlada durante las fases de programación, pruebas y validación funcional.

Para su despliegue en producción, el proyecto ha sido estructurado para facilitar su alojamiento en servidores compatibles con aplicaciones **Python**, tales como:

- **Render** (despliegue ya realizado satisfactoriamente en entorno público).
- **Railway** para despliegues rápidos y autoescalables.
- **Heroku**, con adaptaciones mínimas mediante la inclusión de un archivo Procfile y requirements.txt.
- **Servidores VPS** tradicionales usando **WSGI (Gunicorn o uWSGI)** detrás de **Nginx**, permitiendo una configuración robusta con dominios personalizados y certificados **SSL**.

El despliegue en **Render** ha permitido validar que Pagezy puede ejecutarse de forma continua en la nube con mínima configuración adicional. Render ofrece:

- Soporte nativo para proyectos **Flask** y otros frameworks de Python.
- Gestión automática del entorno virtual y dependencias.
- Asignación gratuita de subdominios.
- Certificados HTTPS integrados.
- Escalado automático y reinicios programados.

Además, la modularidad de la arquitectura de Pagezy permite:

- Mantener múltiples tiendas en una sola instancia.
- O bien, escalar cada tienda como un micro servicio independiente si se opta por una arquitectura distribuida en el futuro.

5.- Viabilidad empresarial

Para evaluar la factibilidad de Pagezy como producto viable en el mercado, se han analizado diversos aspectos que abarcan la técnica, la economía, el tiempo de desarrollo, la escalabilidad futura, el retorno esperado y un análisis estratégico de su entorno competitivo.

5.1.- Viabilidad técnica

Pagezy ha sido construido con tecnologías probadas, robustas y ampliamente utilizadas como Flask, SQLite, HTML, CSS y JavaScript. Su arquitectura modular y el uso de bases de datos independientes por tienda permiten que el sistema sea fácilmente desplegable, escalable y mantenible. No requiere infraestructura compleja ni servidores especializados, lo que lo hace accesible para desarrolladores y emprendedores.

5.2.- Viabilidad económica

La plataforma está pensada para ser de bajo coste tanto en desarrollo como en mantenimiento. No se requieren licencias de software, ya que todo está basado en herramientas de código abierto. Además, el modelo económico puede basarse en suscripciones mensuales por tienda, servicios premium de personalización o despliegues dedicados, lo que abre múltiples líneas de monetización con inversiones mínimas.

5.3.- Viabilidad temporal

El desarrollo completo de Pagezy se ha logrado en un período razonable, gracias a una planificación eficiente y al uso de herramientas ligeras. Esto demuestra que el tiempo de implementación para nuevas versiones, tiendas o funcionalidades es bajo, facilitando la entrega continua y mejora del producto.

5.4.- Viabilidad de escalabilidad

Al estar diseñado con una arquitectura de micro servicios y bases de datos independientes, Pagezy puede escalar horizontalmente sin grandes complicaciones. Es posible añadir nuevas tiendas, mejorar el rendimiento con cachés o separar servicios sin afectar a las demás instancias. Esto permite alojar cientos o miles de tiendas en un entorno distribuido o en contenedores Docker.

5.5.- Viabilidad funcional y de retorno

Desde una perspectiva funcional, el sistema cubre todas las necesidades básicas de una tienda online y puede adaptarse a nuevos requerimientos sin rediseñar el sistema base. El retorno de inversión se justifica tanto por la reducción de costes para el cliente como por el bajo coste operativo para el proveedor. La experiencia del usuario ha sido validada con pruebas funcionales, asegurando su utilidad práctica.

5.6.- Análisis DAFO

- Fortalezas
 - Plataforma ligera y personalizable
 - Bajo coste de despliegue
 - Aislada por tienda, altamente segura
 - Código abierto y modificable
- Debilidades
 - No incluye pasarela de pago integrada aún
 - Panel de administración básico
 - Requiere conocimientos mínimos para el uso
 - No cuenta con app móvil
- Oportunidades
 - Expansión en el mercado de micro tiendas
 - Integración con marketplaces externos
 - Personalización como servicio
- Amenazas
 - Competencia de plataformas consolidadas
 - Cambios rápidos de tecnología web
 - Requiere promoción activa para escalar

Figura 14. Análisis DAFO



6.- Conclusiones

El desarrollo de Pagezy ha demostrado que es posible construir una plataforma completa de creación de tiendas online de forma ágil, escalable y sin necesidad de herramientas propietarias o entornos complejos. La solución propuesta cumple con todos los requisitos básicos de una plataforma de comercio electrónico moderna, incluyendo la gestión de productos, pedidos, clientes, personalización visual y experiencia de usuario dinámica.

Uno de los principales logros ha sido el enfoque modular del sistema, donde cada tienda funciona como una unidad independiente, tanto en términos de visualización como de almacenamiento. Esto garantiza seguridad, escalabilidad y flexibilidad, aspectos clave para cualquier solución que pretenda crecer y adaptarse a distintos modelos de negocio.

Además, Pagezy ha sido concebido con un diseño accesible, tanto para usuarios con conocimientos técnicos básicos como para comerciantes que necesitan una solución rápida, funcional y personalizable. La posibilidad de desplegar tiendas con pocos pasos y personalizar su diseño visualmente convierte a Pagezy en una alternativa viable frente a plataformas más grandes pero menos accesibles.

A nivel personal y académico, este proyecto ha permitido aplicar conocimientos adquiridos durante la formación, integrar tecnologías diversas y enfrentar retos de planificación, diseño y programación que simulan un entorno profesional real.

En definitiva, Pagezy no solo constituye un proyecto técnico sólido, sino también una propuesta funcional con potencial de aplicación real en el mercado.

7.- Bibliografía

Browse fonts. (n.d.). Google Fonts. Retrieved May 23, 2025, from <https://fonts.google.com/>

GitHub · Build and ship software on a single, collaborative platform. (n.d.).

Javascript - Javascript en español. (n.d.). Lenguajejs.com. Retrieved May 23, 2025, from <https://lenguajejs.com/javascript/>

Jinja — Jinja documentation (3.1.X). (n.d.). Palletsprojects.com. Retrieved May 23, 2025, from <https://jinja.palletsprojects.com/en/stable/>

Lenguaje CSS - CSS en español. (n.d.). Lenguajecss.com. Retrieved May 23, 2025, from <https://lenguajecss.com/>

Lenguaje HTML - HTML en español. (n.d.). Lenguajehtml.com. Retrieved May 23, 2025, from <https://lenguajehtml.com/>

Sass: Syntactically awesome style sheets. (n.d.). Sass-lang.com. Retrieved May 23, 2025, from <https://sass-lang.com/>

SQLite Home Page. (n.d.). Sqlite.org. Retrieved May 23, 2025, from <https://www.sqlite.org/>

Visual Studio Code - code editing. Redefined. (n.d.). Visualstudio.com. Retrieved May 23, 2025, from <https://code.visualstudio.com/>

Welcome to flask — flask documentation (3.1.X). (n.d.). Palletsprojects.com. Retrieved May 23, 2025, from <https://flask.palletsprojects.com/en/stable/>

Cloud application platform. (n.d.). Render. Retrieved May 23, 2025, from <https://render.com/>

8.- Anexo

8.1.- Fragmentos de código relevantes

- Fragmento 1 – Registro de rutas para productos en Flask

Figura 15. Código de ver_productos

```
@tienda_bp.route("/<nombre_tienda>/productos")
@tienda_bp.route("/<nombre_tienda>/productos/<int:pagina>")
def ver_productos(nombre_tienda, pagina=1):
    tienda_publica = TiendaPublica.query.filter_by(nombre_tienda=nombre_tienda).first_or_404()
    engine = create_engine(f"sqlite:///{"tienda_publica.ruta_db}")
    metadata = MetaData()
    metadata.reflect(bind=engine)

    config_table = metadata.tables["configuracion_visual"]
    producto_table = metadata.tables["producto"]
```

- Fragmento 2 – Creación de pedidos y reducción de stock

Figura 16. Código de creación de pedidos

```
for p in data["carrito"]:
    producto_id = int(p["id"])
    cantidad_comprada = int(p["cantidad"])

    # Insertar en la tabla compra
    conn.execute(
        compra_table.insert().values(
            pedido_id=pedido_id,
            producto_id=producto_id,
            cantidad=cantidad_comprada
        )
    )

    # Descontar stock del producto
    conn.execute(
        producto_table.update()
        .where(producto_table.c.id == producto_id)
        .values(stock=producto_table.c.stock - cantidad_comprada)
    )
```

- Fragmento 3 – Filtros dinámicos en JavaScript

Figura 17. Código de filtros de productos

```
const form = document.getElementById('filtros-form');
const productosContainer = document.getElementById('productos-container');
let timeout = null;

form.addEventListener('input', () => {
  clearTimeout(timeout);
  timeout = setTimeout(() => {
    const params = new URLSearchParams(new FormData(form));
    fetch(window.location.pathname + '?' + params.toString(), {
      headers: { 'X-Requested-With': 'XMLHttpRequest' }
    })
    .then(res => res.text())
    .then(html => {
      productosContainer.innerHTML = html;
    });
  }, 300);
});
```

- Fragmento 4 – Panel de administración con pestañas

Figura 18. Código de mostrar u ocultar pestañas

```
document.addEventListener("DOMContentLoaded", function () {
  const tabs = {
    clientes: document.getElementById("btn-clientes"),
    pedidos: document.getElementById("btn-pedidos"),
    compras: document.getElementById("btn-compras"),
    productos: document.getElementById("btn-productos"),
  };

  const sections = {
    clientes: document.getElementById("seccion-clientes"),
    pedidos: document.getElementById("seccion-pedidos"),
    compras: document.getElementById("seccion-compras"),
    productos: document.getElementById("seccion-productos"),
  };

  Object.keys(tabs).forEach(key => {
    tabs[key].addEventListener("click", () => {
      Object.values(sections).forEach(s => s.style.display = "none");
      sections[key].style.display = "block";
    });
  });

  // Mostrar sección clientes por defecto
  tabs.clientes.click();
});
```