

A white graphic consisting of an elliptical orbit with two four-pointed star shapes representing celestial bodies, one at each focus. The word "Gravity" is written in a white, bold, sans-serif font, centered within the orbit.

Gravity

Índice

Contenido

Índice	1
1.Introducción	3
1.1 Resumen	3
1.2 Objetivos	3
2. Planificación	4
2.1 Metodología	4
2.2 Temporalización	5
3. Análisis	5
3.1 Estudio de mercado	5
3.1.1 ManageEngine AssetExplorer	6
3.1.2 LicenseSpring	6
3.1.3 GLPI	7
3.2 Conclusión	7
4. Requisitos de la Aplicación	8
4.1 Funcionales	8
4.2 No Funcionales	8
4.3 Targets principales	9
5. Diseño de la app	9
5.1 Diseño arquitectónico	9
5.2 Diseño de la interfaz de usuario	10
6. Base de datos	11
6.1 Descripción de tablas	11
6.2 Implementación	12
7. Interfaz Gráfica	13
8. Implementación	15
8.1 Tecnologías	15
8.2 Dependencias y arquitectura	16
9. Desarrollo de la aplicación	17
9.1 Estructura del proyecto	17
9.2 Fase 1 Infraestructura y Autenticación	19
9.3 Fase 2 Gestión de Licencias	20
9.4 Fase 3 Funcionalidades avanzadas y usuarios	21
9.5 Fase 4 Sistema de Soporte	22
9.6 Fase 5 Integración de ChatBot con inteligencia artificial	23

10. Pruebas.....	24
11. Conclusiones	26
11.1 Complejidad desarrollo del Front Dinámico	26
11.2 Complejidad de Front responsive	26
11.3 Lecciones Aprendidas	26
12. Posibles mejoras.....	27
12.1 Mejoras de seguridad	27
12.2 Mejoras Funcionales.....	27
12.3 Mejoras de Experiencia de Usuario	27
13. Resumen Final	28
14. Bibliografía	29

1.Introducción

1.1 Resumen

El TFG trata el desarrollo de una aplicación web integral, diseñada específicamente para la gestión, distribución y monetización de licencias de software.

Esta plataforma tiene como objetivo principal proporcionar un sistema completo y fácil de usar tanto para empresas que desean comercializar sus productos de software como para usuarios individuales que buscan una manera sencilla de adquirir y administrar sus licencias.

Al ofrecer funcionalidades clave como la venta, la administración y el control de licencias, junto con herramientas de soporte integradas como un sistema de mensajería y tickets, esta aplicación busca mejorar significativamente la experiencia tanto de proveedores como de consumidores de software.

1.2 Objetivos

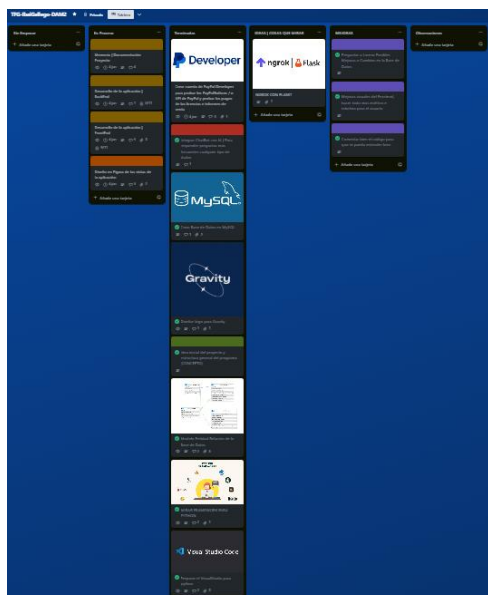
1. **Implementar un sistema de gestión de licencias eficiente:** Esto implica la creación de una buena base de datos y una interfaz intuitiva que permite a los administradores registrar, categorizar y gestionar tanto las diferentes licencias de software disponibles como los usuarios.
2. **Establecer un sistema de control de acceso y uso de licencias:** Será fundamental implementar mecanismos que permiten verificar la validez de las licencias, controlar el número de dispositivos o usuarios autorizados y gestionar posibles renovaciones o cancelaciones.
3. **Integrar herramientas de soporte al usuario:** La aplicación incorpora un sistema de mensajería/tickets de soporte técnico y un Chat Bot con IA, facilitando la comunicación entre los proveedores y los usuarios para resolver dudas lo antes posible.
4. **Ofrecer funcionalidades de monetización seguras y flexibles:** Los pagos se realizan con la API de PayPal, y desde el panel de administración se pueden visualizar los informes de venta de cada licencia.
5. **Desarrollar una interfaz de usuario intuitiva y accesible:** La aplicación cuenta con un diseño atractivo y una navegación sencilla, adaptable a diferentes dispositivos y niveles de experiencia del usuario.

2. Planificación

2.1 Metodología

Para este proyecto, voy a usar **SCRUM**. Esta manera me ayudará a ser organizado y flexible mientras desarrollo la aplicación.

Usar **SCRUM** me permitirá ir avanzando poco a poco, adaptarme si surgen ideas nuevas y asegurarme de que el proyecto final cumpla con lo que necesito.



TRELLO apuntando tareas, y marcando las que se van realizando día a día.

Enlace Trello

<https://trello.com/b/UNoaKey4/tfg-ibaigallego-dam2>

Ilustración 1 – Tablero del Trello

2.2 Temporalización

FASES GENERALES	FASES ESPECÍFICAS	TIEMPO ESTIMADO
Planificación y Diseño	Investigación y análisis de requisitos	4 horas
	Investigación del lenguaje Python	3 horas
	Investigación del Framework Flask	2 horas
	Diseño de la arquitectura de la app	1 hora
	Diseño de la interfaz de usuario	4 horas
Desarrollo de la funcionalidad básica	Implementación de Autenticación	4 horas
	Implementación de configuración de usuario	4 horas
	Implementación de la interfaz de usuario, menú interactivo y la carga del contenido dinámicamente	6 horas
	Implementación del cierre de sesión	1 horas
Desarrollo de la funcionalidad específica	Implementación de las licencias	5 horas
	Implementación del panel de administrador	6 horas
	Implementación del panel de licencias	4 horas
	Implementación del ChatBot con IA	3 horas
	Implementación de tickets para soporte.	5 horas
Pruebas y correcciones	Pruebas de funcionalidad	2 horas
	Corrección de errores y ajustes finales	5 horas
Documentación	Memoria	A lo largo del TFG (6 horas)
	Preparación de la presentación de la aplicación del TFG	3 horas
TOTAL HORAS		68 horas

3. Análisis

3.1 Estudio de mercado

Con el objetivo de entender mejor el entorno competitivo en el que se sitúa Gravity, he analizado tres plataformas relevantes del sector.

Dos de pago y una gratuita para ver si hay mucha diferencia.

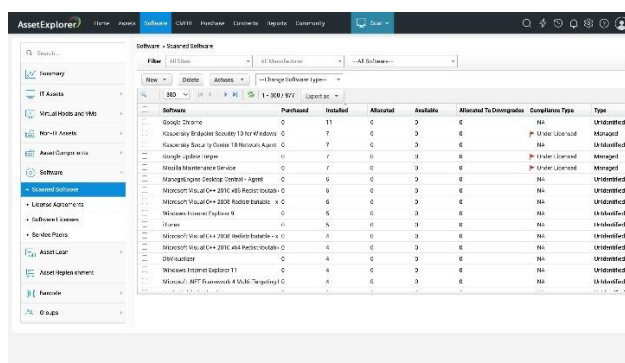
ManageEngine AssetExplorer, **LicenseSpring** y **GLPI**. Estas soluciones, aunque con enfoques variados, comparten funcionalidades similares relacionadas con la gestión de licencias, activos de software y soporte técnico.

3.1.1 ManageEngine AssetExplorer

ManageEngine AssetExplorer es una herramienta profesional enfocada en la gestión de activos de TI, incluyendo el seguimiento y control de licencias de software. Tiene una interfaz relativamente intuitiva, capacidad para detectar automáticamente software y hardware en red, y sus funciones de cumplimiento normativo.

Aunque es una solución robusta, su complejidad puede representar una barrera para pequeñas empresas o usuarios sin experiencia técnica.

Esta aplicación está orientada a organizaciones medianas y grandes.



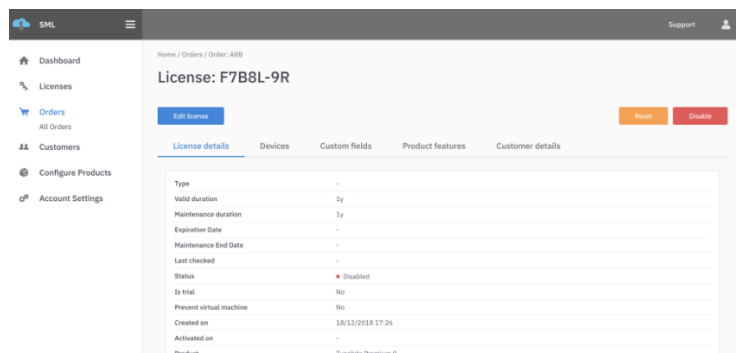
Software	Purchased	Installed	Activated	Available	Activated To/Overage	Compliance Type	Type
Google Chrome	0	11	0	0	0	N/A	Unidentified
Kaspersky Endpoint Security 13 for Windows	0	7	0	0	0	P - Under License	Managed
Kaspersky Small Business Security 13 for Windows	0	7	0	0	0	N/A	Unidentified
Google Update Manager	0	7	0	0	0	P - Under License	Managed
Intel Management Engine	0	7	0	0	0	P - Under License	Managed
Microsoft Visual C++ 2010 x86 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2010 x64 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2012 x86 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2012 x64 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2013 x86 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2013 x64 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2015 x86 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2015 x64 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2017 x86 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2017 x64 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2019 x86 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2019 x64 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2019 x86 Redistributable	0	6	0	0	0	N/A	Unidentified
Microsoft Visual C++ 2019 x64 Redistributable	0	6	0	0	0	N/A	Unidentified

Ilustración 2 - ManageEngine AssetExplorer Interfaz

3.1.2 LicenseSpring

LicenseSpring es una solución en la nube que permite a los desarrolladores de software gestionar licencias y monetizar sus productos de manera eficiente. Ofrece una amplia gama de modelos de licencias.

Una de las características destacadas de LicenseSpring es su enfoque en la automatización del ciclo de vida de las licencias, desde que se activa hasta que se renueva, lo que reduce la carga operativa para los desarrolladores.



License details	
Type	-
Valid duration	1y
Maintenance duration	1y
Expiration date	-
Maintenance End Date	-
Last checked	-
Status	Disabled
Is trial	No
Present virtual machine	No
Created on	18/12/2019 17:26
Activated on	-
Product	TuneBite Premium 9

Ilustración 3 - LicenseSpring interfaz

3.1.3 GLPI

GLPI es una solución de código abierto utilizada para la gestión de activos informáticos y soporte técnico. A diferencia de las anteriores, es gratis y cuenta con una comunidad activa de desarrolladores. Tiene funciones como la creación de tickets de soporte, inventario de hardware y software, y control básico de licencias. La interfaz es menos moderna, pero cumple con los requerimientos funcionales.

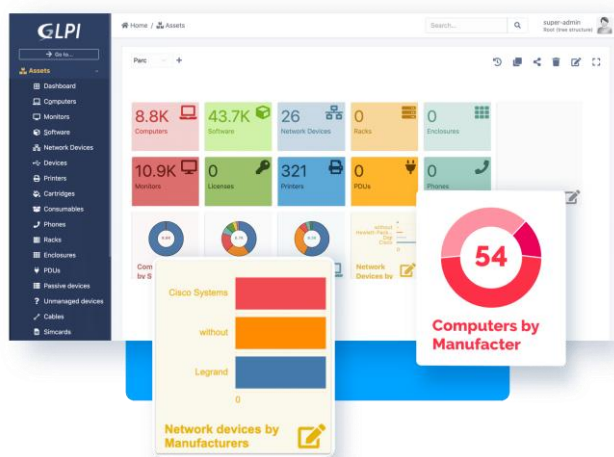


Ilustración 4 - GLPI Interfaz

3.2 Conclusión

Haciendo el análisis del mercado he llegado a la conclusión de que, aunque existen varias plataformas dedicadas a la gestión de licencias de software, la mayoría se centran en soluciones orientadas a empresas medianas o muy grandes, con enfoques técnicos complejos o funcionalidades específicas.

Además, muchas de estas soluciones no tienen integración directa con pasarelas de pago o un sistema de soporte al usuario accesible, cosas esenciales para ofrecer una experiencia al completo tanto para el cliente como para vendedor.

En este contexto, Gravity (Mi aplicación) es una alternativa más accesible, al tener en una sola plataforma la gestión de licencias, la comercialización directa utilizando PayPal, y un sistema de atención al cliente y el ChatBot con IA. Todas esto junto y que tiene un diseño intuitivo, permite cubrir las necesidades de pequeñas empresas, startups y desarrolladores independientes.

4. Requisitos de la Aplicación

4.1 Funcionales

Requisitos Funcionales (Qué hará la aplicación):

- **Gestión de Licencias:** Añadir, categorizar, editar, desactivar/eliminar licencias; buscar y ver detalles de licencias.
- **Control de Acceso y Uso:** Registro/Login de usuarios, compra de licencias, generación/activación/verificación de claves, control de dispositivos/usuarios, notificaciones de expiración, renovación y cancelación.
- **Soporte al Usuario:** Abrir/ver estado/responder tickets, mensajería en tiempo real, Chat Bot con IA (respuestas y escalado a agente).
- **Monetización:** Integración con PayPal, notificaciones de pago, informes de ventas por licencia.
- **Interfaz de Usuario:** Navegación muy fácil, diseño responsive, intuitivo y con mensajes claros.

4.2 No Funcionales

Requisitos No Funcionales (Cómo será la aplicación):

- **Rendimiento:** Carga rápida de páginas, manejo de usuarios concurrentes, consultas rápidas a la base de datos.
- **Seguridad:** Contraseñas encriptadas, HTTPS, protección contra vulnerabilidades, acceso restringido a administración, auditorías.
- **Usabilidad:** Cumplimiento de accesibilidad web, navegación consistente, mensajes de error claros, documentación de ayuda.
- **Escalabilidad:** Arquitectura que permita aumentar recursos fácilmente.
- **Mantenibilidad:** Código bien estructurado y documentado, estándares de codificación, pruebas.
- **Portabilidad/Compatibilidad:** Funcionar en navegadores principales y adaptarse a diferentes pantallas (Responsive).
- **Fiabilidad:** Minimizar errores, registro de actividad (logging).

4.3 Targets principales

Empresas de software bajo la licencia **SaaS** (Software as a Service). **Desarrolladores independientes** que quieren monetizar sus **softwares**.

5. Diseño de la app

El diseño de la aplicación tiene un objetivo claro, que es ofrecer una experiencia de usuario clara y sencilla. Este apartado incluye tanto el diseño arquitectónico, como el diseño de interfaces, detallando las decisiones tomadas durante el desarrollo.

Dejo [aquí](#) un enlace a la página de **Figma** con el prototipo inicial del diseño de la app.

5.1 Diseño arquitectónico

La aplicación sigue una arquitectura cliente-servidor, en la que el cliente (Frontend) se comunica con el servidor (Backend) mediante peticiones HTTPS a través de una API RESTful.

- **Frontend:** Desarrollado como una aplicación web responsive que se adapta a distintos dispositivos.
- **Backend:** Encargado de la lógica de negocio, validación de datos, autenticación, gestión de licencias y conexión con la base de datos. Está construido sobre **Flask**, un Framework de Python.
- **Base de datos:** Almacena información sobre usuarios, licencias y tickets de soporte. Se utiliza MySQL.
- **Servicios Externos:** La aplicación se integra con la API de PayPal para el procesamiento de pagos, por otro lado, tenemos también el ChatBot con IA que utilizará la API de OpenAI.

5.2 Diseño de la interfaz de usuario

He priorizado una interfaz moderna, intuitiva y fácil de navegar, para que cualquier persona pueda utilizarla sin ninguna complicación.

- **Diseño responsive:** La interfaz se adapta correctamente a distintos tamaños de pantalla.
- **Panel de administrador:** Permite a los administradores gestionar licencias, usuarios, mensajes y tickets de soporte.
- **Panel de usuario:** Inicio con información relevante y últimas novedades, configuración, mensajes/tickets y apartado de compra de licencia.
- **Sistema de tickets y mensajería:** Integrado dentro de la interfaz, accesible desde cualquier dispositivo.
- **ChatBot con IA:** Visible desde la interfaz como asistente flotante, capaz de resolver dudas frecuentes.

6. Base de datos

Para la base de datos de la aplicación se ha utilizado MySQL, el modelo de datos se ha diseñado para que sea claro, escalable y alineado con los requisitos de la aplicación, permitiendo una gestión eficaz de los usuarios, licencias, y el soporte mediante tickets.

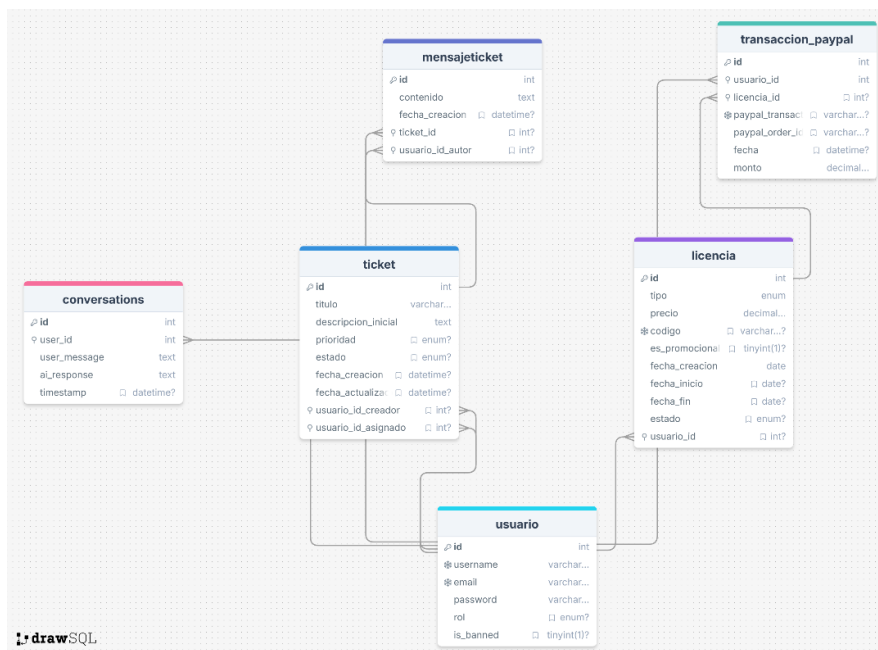


Ilustración 5 - Diagrama de la base de datos

6.1 Descripción de tablas

USUARIO → Guarda los datos de todos los usuarios del sistema (Clientes, vendedores y administradores).

- **Relaciones**

- Puede tener muchas licencias
- Puede crear muchos tickets
- Puede estar asignado como agente a muchos tickets
- Puede escribir muchos mensajes en los tickets

LICENCIA → Almacena información sobre las licencias de software (Tipo, Precio, Estado, código... etc.).

- **Relaciones**

- Cada licencia puede estar asociada a un solo usuario (El que la compró o la recibió)

TICKET → Es el sistema de soporte. Aquí se crean las solicitudes o incidencias de los usuarios.

- **Relaciones**
 - Cada ticket tiene un creador (Usuario)
 - Puede estar asignado a un agente (Otro usuario)
 - Puede tener varios mensajes

MENSAJE-TICKET → Son los mensajes que se escriben dentro de los tickets (Conversación).

- **Relaciones**
 - Cada mensaje pertenece a un ticket
 - Cada mensaje tiene un autor (usuario que lo escribió)

TRANSACCIÓN PAYPAL → Es la tabla donde se almacena la transacción de paypal que hace el usuario cuando compra una licencia.

- **Relaciones**
 - Cada transacción tiene un usuario asignado (Comprador de la licencia).
 - Cada transacción tiene un id de licencia con el que está vinculado.

6.2 Implementación

Durante el desarrollo del proyecto, consideré diferentes opciones para la gestión de la base de datos. Una de las ellas fue SQLite, que como es compatible con Flask y SQLAlchemy es muy fácil de configurar.

Sin embargo, al final opte por utilizar MySQL porque es más profesional, escalable y robusto. Aunque es más complejo, porque requiere una configuración inicial más compleja (Instalar servidor, usuarios, conectarlo con el backend...) tiene mejor rendimiento si hay varios usuarios simultáneos conectados, también soporta más volumen de datos.

MySQL se utiliza más en el mundo real.

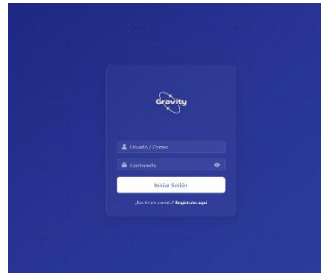
7. Interfaz Gráfica

A continuación, se muestran todas las pantallas acabadas que se utilizan en la aplicación.

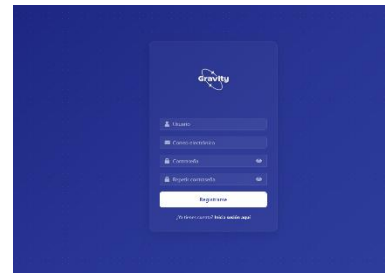
También las pantallas de inicio de sesión del usuario.



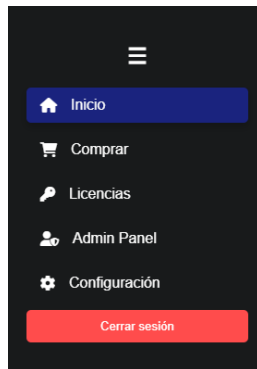
Página 1 – Start Page



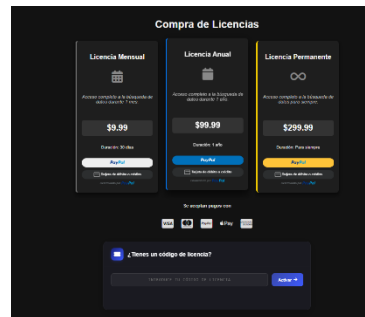
Página 2 - Inicio de Sesión



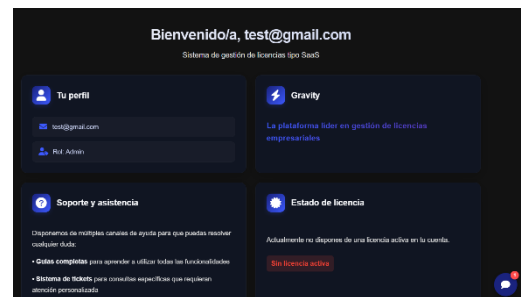
Página 3 - Registro



Página 4 – Menú lateral



Página 5 – Compra licencias



Página 6 – Inicio

Gestión de Licencias

Buscar en la tabla

ID	USUARIO	TIPO	CÓDIGO	ESTADO	FECHA INICIO	FECHA FIN	ACCIONES
1	NA	Mensual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
2	logi	Actual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
3	NA	Actual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
4	test	Actual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
5	logi	Actual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
6	NA	Mensual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
7	test	Mensual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
8	logi	Actual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
9	NA	Mensual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
10	test	Mensual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
11	NA	Mensual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
12	logi	Mensual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
13	NA	Mensual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar
14	logi	Mensual	12345678901234567890	Expirado	26/05/2025	26/05/2025	Ver Eliminar

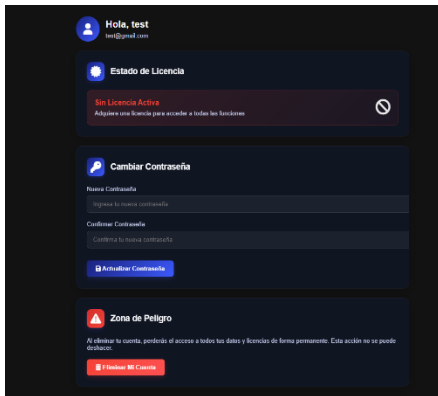
Página 7 – Gestión de licencias

Panel de Administración

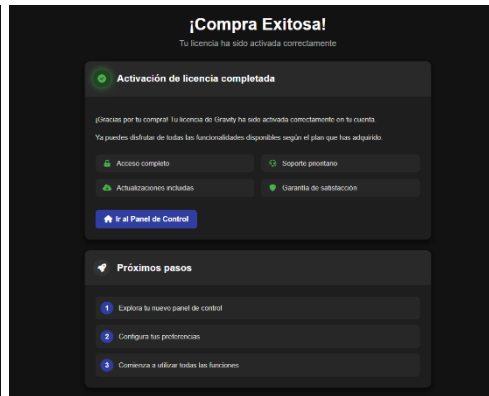
Buscar en la tabla

ID	USUARIO	EMAIL	ROL	LICENCIA	ACCIONES
1	test	test@gmail.com	Administrador	Expirado	Ver Eliminar
2	logi	logi@gmail.com	Usuario	Expirado	Ver Eliminar
3	gravity	gravity@gmail.com	Usuario	Expirado	Ver Eliminar

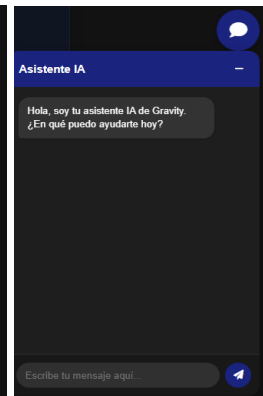
Página 8 – Panel de Administración



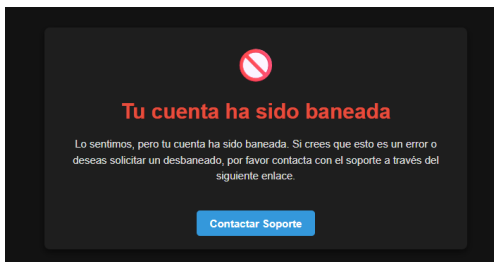
Página 9 – Configuración de cuenta



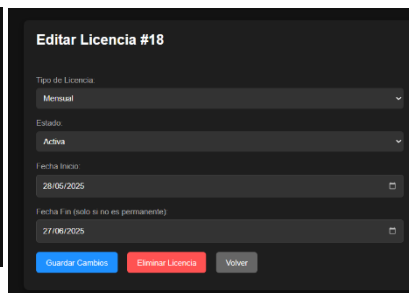
Página 10 – Compra exitosa



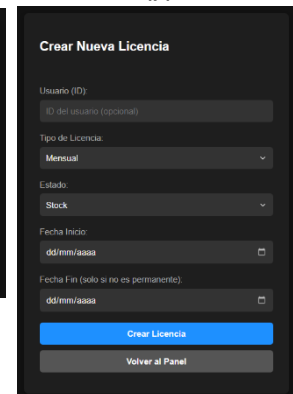
Página 11 – ChatBot IA



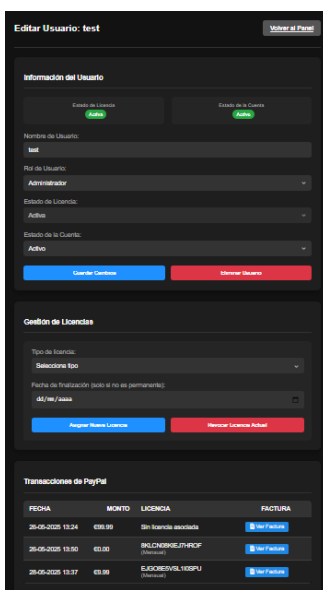
Página 12 – Cuenta bloqueada



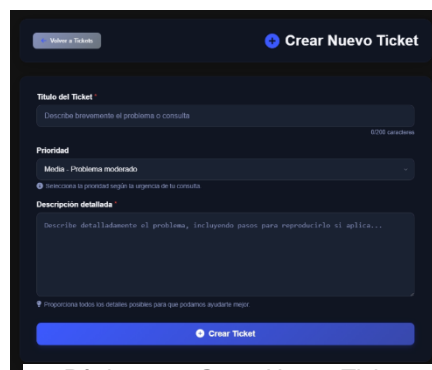
Página 13 – Editar Licencia



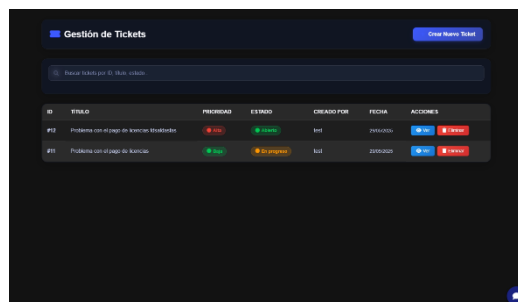
Página 14 – Crear Licencia



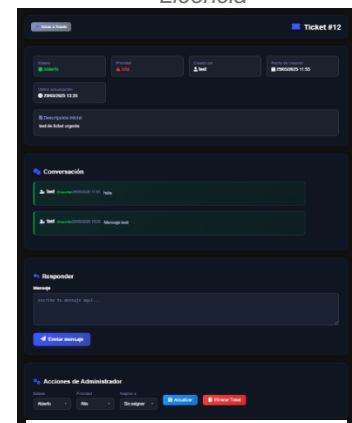
Página 15 – Ver/Editar Usuarios



Página 16 – Crear Nuevo Ticket



Página 18 – Pantalla Tickets/Soporte



Página 17 – Vista de Ticket

8. Implementación

8.1 Tecnologías

A continuación, se detallan las tecnologías, frameworks y herramientas que han sido implementadas en el desarrollo de este sistema de gestión de licencias SaaS, junto con las principales dependencias y librerías utilizadas.

Tecnologías Principales

- **Python**, como lenguaje de desarrollo principal para escribir toda la lógica del servidor backend, gestión de rutas, modelos de datos y servicios de la aplicación.
- **JavaScript**, utilizado para la funcionalidad del frontend, incluyendo interacciones dinámicas, validaciones del lado del cliente, manejo de eventos y comunicación asíncrona con el servidor mediante AJAX.
- **Flask**, como framework web principal de Python para el desarrollo del backend. Proporcionando un sistema robusto de enrutamiento, gestión de sesiones, y arquitectura modular mediante blueprints.
- **HTML5 y CSS**, para la estructura y diseño de la interfaz de usuario, creando una experiencia visual moderna y responsive.
- **Visual Studio Code**, como IDE principal de desarrollo, proporcionando un entorno integrado con extensiones específicas para Python y desarrollo para aplicaciones web.
- **MySQL**, como sistema de gestión de base de datos relacional para el almacenamiento de usuarios, licencias, transacciones y conversaciones.
- **PayPal SDK**, para la integración del sistema de pagos y procesamiento de transacciones de licencias.

8.2 Dependencias y arquitectura

Dependencias y Librerías

Backend (Python)

- **Flask-Login:** Gestiona la autenticación y sesiones de usuarios, proporcionando decoradores para proteger rutas y mantener el estado de login.
- **Flask-SQLAlchemy:** ORM (Object-Relational Mapping) que facilita la interacción con la base de datos MySQL mediante modelos de datos en Python.
- **Werkzeug:** Librería de utilidades que proporciona funciones de seguridad como el hash de contraseñas y validaciones.
- **OpenAI:** API para la integración del chatbot inteligente basado en GPT, permitiendo generar respuestas contextuales y especializadas.
- **MySQL Connector:** Driver de conexión para establecer comunicación entre la aplicación Flask y la base de datos MySQL.

Frontend (JavaScript/CSS/HTML)

- **Font Awesome:** Librería de iconos vectoriales que proporciona una amplia gama de iconos para mejorar la interfaz de usuario.
- **AOS (Animate On Scroll):** Librería que añade animaciones suaves cuando los elementos entran en el viewport durante el scroll.
- **PayPal JavaScript SDK:** Cliente JavaScript para integrar los botones de pago de PayPal directamente en la interfaz web.
- **CSS Grid y Flexbox:** Tecnologías nativas de CSS para crear layouts responsivos y estructuras de diseño modernas.

Arquitectura y Patrones

- **Blueprint Pattern:** Patrón de arquitectura de Flask para organizar la aplicación en módulos separados (auth, admin, licenses, search, etc.).
- **MVC (Model-View-Controller):** Patrón arquitectónico implementado separando modelos de datos, vistas de templates y controladores de rutas.
- **AJAX:** Tecnología para comunicación asíncrona entre el frontend y backend, mejorando la experiencia de usuario sin recargas de página.

9. Desarrollo de la aplicación

Durante el desarrollo de mi aplicación, he optado por utilizar Python como lenguaje principal y el Framework Flask para construir la estructura de la aplicación.

Para el entorno de desarrollo, he utilizado Visual Studio Code, ha sido fundamental a lo largo del proceso. Elegí VS Code por varias razones:

- Integración con Python y Flask: Gracias a la extensión oficial de Python y otras herramientas disponibles, VS Code ofrece autocompletado, depuración y un entorno cómodo para el desarrollo con Flask.
- Me ha resultado muy útil poder ejecutar comandos de Flask, Git o cualquier otro script directamente desde la terminal integrada, sin necesidad de cambiar de ventana.
- Control de Versiones con Git, VS Code cuenta con integración directa con Git, lo que me ha facilitado llevar un control detallado de los cambios realizados en el código, hacer commits y gestionar el repositorio desde el mismo editor.
- Soporte para plantillas HTML, CSS y JavaScript, al trabajar con Flask, también he creado plantillas HTML y estilos en CSS.

9.1 Estructura del proyecto

La estructura del proyecto la he dividido en diferentes partes como se puede apreciar.

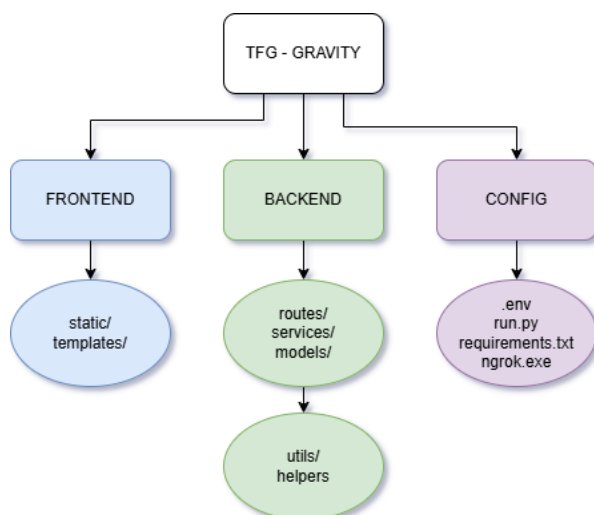


Ilustración 6 Estructura del proyecto

Frontend

- static/ y templates/: contienen los archivos servidos al navegador (HTML, CSS, JavaScript, imágenes).
- Función: gestionan la interfaz de usuario. El servidor los entrega al cliente cuando se accede a la aplicación.

Backend

- routes/: define los endpoints de la API.
- services/: contiene la lógica de negocio.
- models/: define entidades y su relación con la base de datos.
- utils/, helpers/: funciones reutilizables (validaciones, formateos, etc.).
- Función: gestiona datos, reglas de negocio y respuestas a peticiones del frontend.

Configuración

- .env: variables de entorno.
- run.py: arranque de la aplicación.
- requirements.txt: dependencias del proyecto.
- ngrok.exe: túnel para exponer la aplicación localmente.
- Función: configura y permite ejecutar la aplicación en distintos entornos.

9.2 Fase 1 Infraestructura y Autenticación

En la primera fase hice los cimientos técnicos del sistema, implementando la arquitectura base de la aplicación web y los mecanismos fundamentales de seguridad (encriptar contraseñas), @login_required en las páginas y autenticación de usuarios.

Objetivos de la Fase

- Establecer una arquitectura sólida y escalable basada en Flask
- Implementar un sistema de autenticación robusto y seguro
- Configurar la base de datos y las conexiones necesarias
- Crear la estructura modular que permita el desarrollo incremental

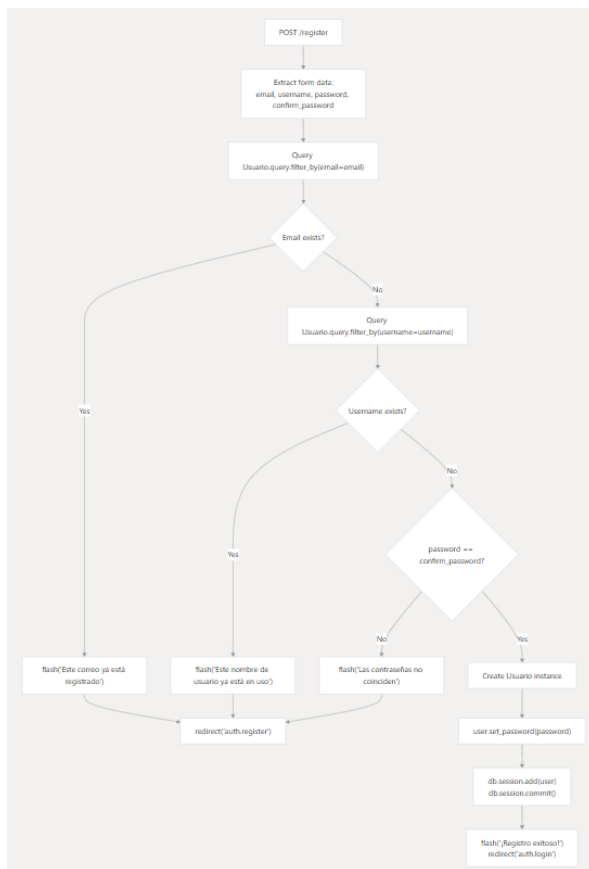


Ilustración 7 Flujo de validación de Registro

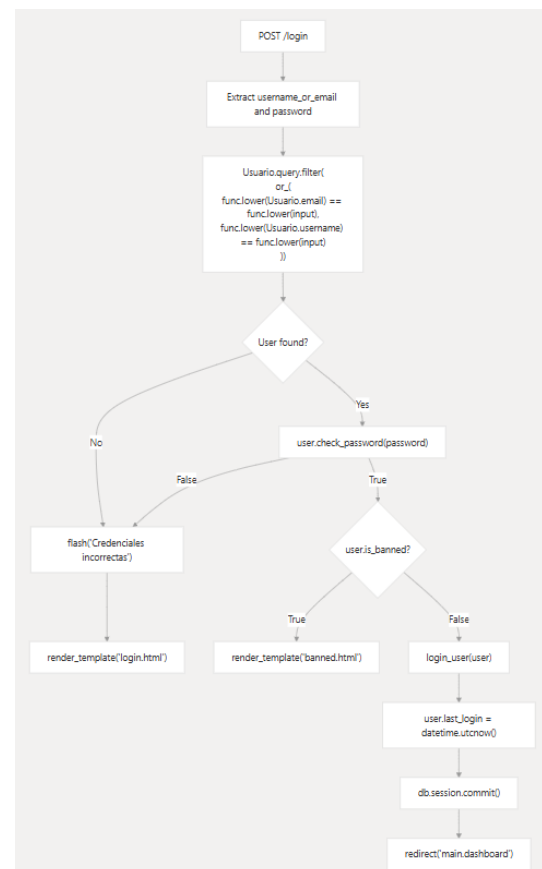


Ilustración 8 Flujo de validación de Inicio de Sesión

9.3 Fase 2 Gestión de Licencias

La segunda fase me centré en implementar la funcionalidad central del sistema: el sistema completo de gestión de licencias SaaS, incluyendo creación, eliminación, modificación, activación, seguimiento y administración de licencias.

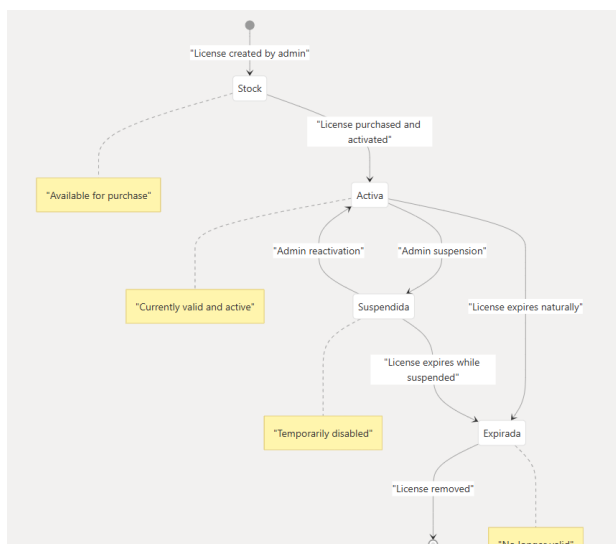


Ilustración 9 - Flujo de Gestión del estado de la licencia

Objetivos de la Fase

- Desarrollar el sistema completo de gestión de licencias
- Implementar la estructura de precios y tipos de licencia
- Crear herramientas administrativas para gestión de licencias
- Establecer el flujo de trabajo de activación de licencias

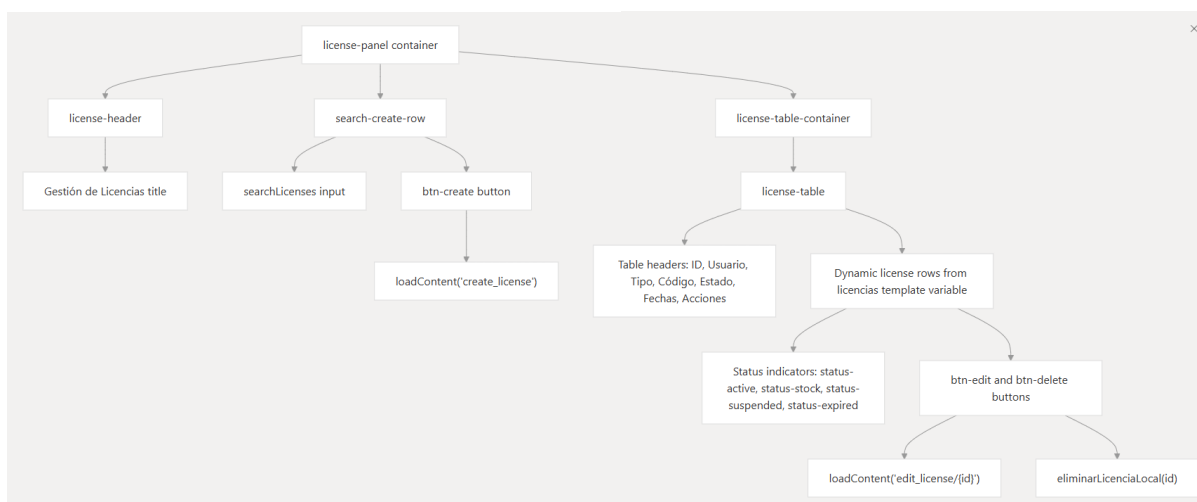


Ilustración 10 – Componentes de la interfaz del panel

9.4 Fase 3 Funcionalidades avanzadas y usuarios

Una vez que tuve la base funcionando, fue hora de añadir las características importantes y más avanzadas. La integración con PayPal fue clave aquí: logré que cuando alguien compraba una licencia, todo sucediera automáticamente, el pago se procesaba, la licencia se activara inmediatamente, y al cliente se le añadía un registro completo de la transacción, que el admin también puede ver.

También pulí todas las herramientas administrativas. Los administradores ahora podían gestionar usuarios fácilmente, editar perfiles cuando sea necesario, controlar roles sin complicaciones, y tener visibilidad completa del historial de transacciones.

Para mejorar la experiencia de usuario, implementé interfaces dinámicas con AJAX, esto significó que las páginas se actualizaban sin necesidad de recargar constantemente, haciendo todo más fluido y profesional.

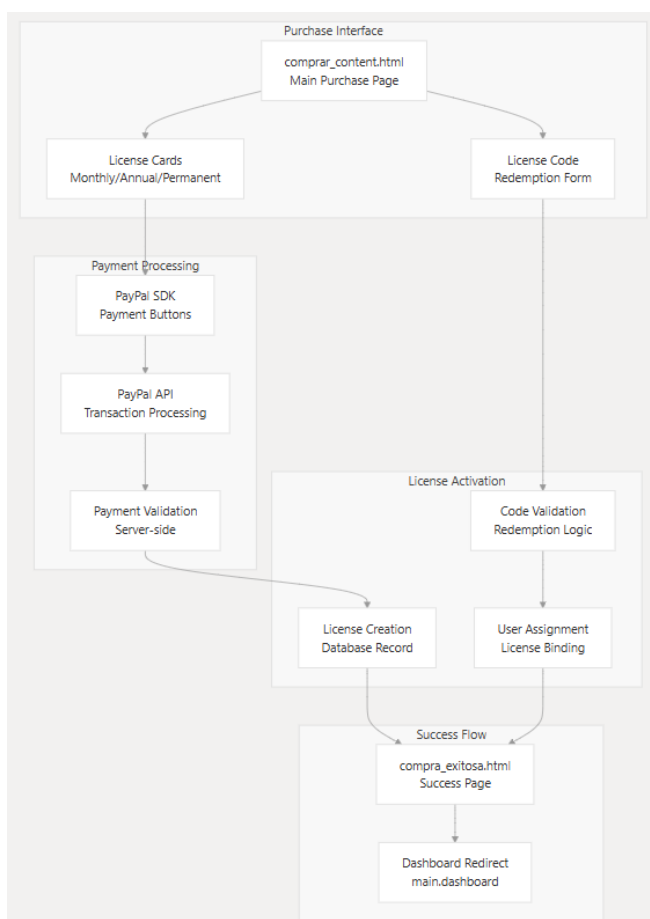


Ilustración 11 – Descripción general del sistema de compra

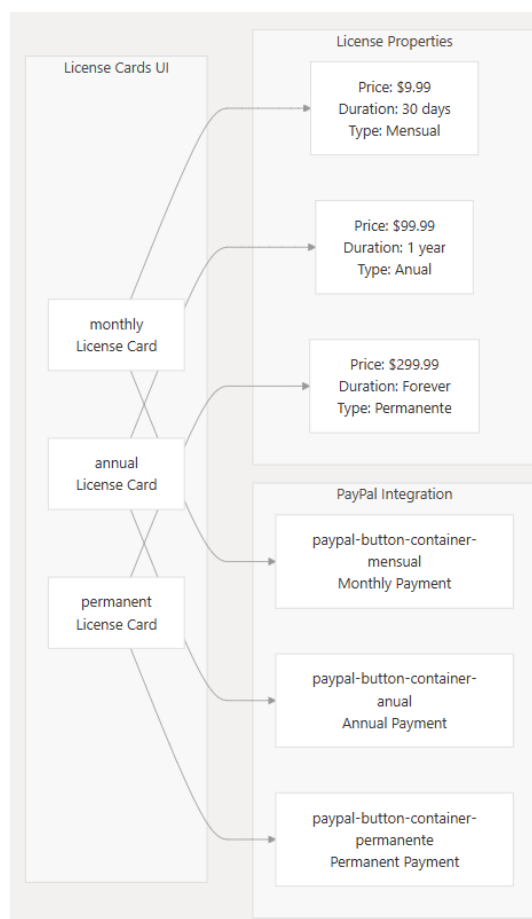


Ilustración 12 – Tipos de licencia y precios

9.5 Fase 4 Sistema de Soporte

Sabía que un buen soporte al cliente podía aportar bastante. Por eso implementé un sistema completo de tickets para poder dar soporte a usuarios o clientes desde el mismo sitio.

Organicé los tickets por prioridad (alta, media, baja) para poder atender primero lo más urgente. Cada ticket tiene un estado claro: abierto cuando llegaba, en progreso cuando alguien está trabajando en él, y cerrado cuando se resuelve el problema.

Lo realmente útil es el sistema de mensajería dentro de cada ticket como una conversación al estilo “WhatsApp” pero más básica. Los clientes pueden seguir la conversación, ver quién les estaba ayudando, y tener acceso a todo el historial de su caso.

Para los administradores: Desarrollé herramientas que les permitieron gestionar eficientemente todos los tickets, asignarlos al personal adecuado, y mantener a los clientes informados con actualizaciones en “tiempo real”.

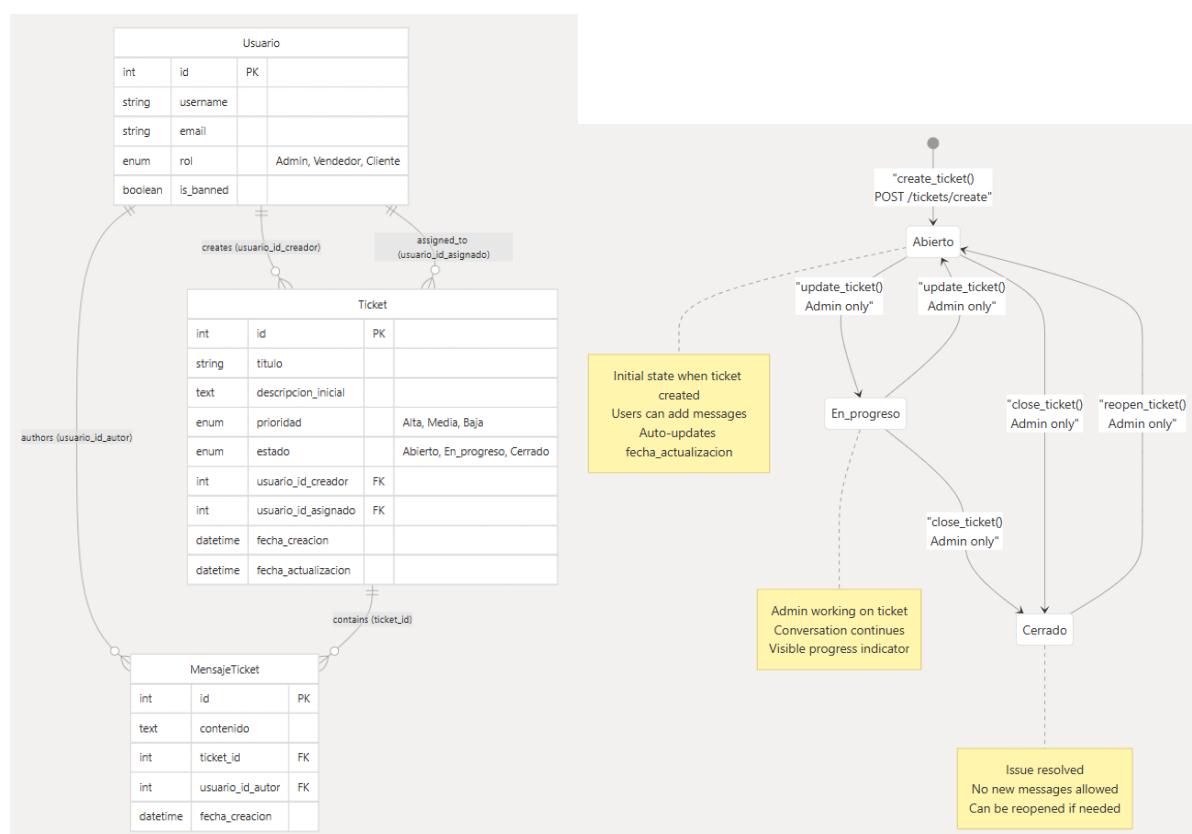


Ilustración 13 – Descripción general del sistema de compra

Ilustración 14 – Ciclo de vida y estado de los tickets

9.6 Fase 5 Integración de ChatBot con inteligencia artificial

Para aumentar el soporte de todo mi sistema, integré un chatbot inteligente que usa la tecnología GPT-4o-mini de OpenAI. No es un chatbot genérico, el ChatBot tiene un contexto específico relacionado con mi aplicación del TFG para poder conocer todo sobre Gravity y cómo ayudar a mis usuarios.

Configuré el asistente con un límite de 150 tokens y temperatura 0.7 para mantener las respuestas directas pero naturales. Lo mejor de todo es que está disponible 24/7, así que mis clientes siempre tienen a alguien que los ayudara, sin importar la hora.

El chatbot tiene conocimiento profundo sobre cómo gestionar licencias, los procesos de compra, la administración de usuarios, y por supuesto, todo funciona perfectamente en cualquier idioma y con manejo inteligente de errores.

Al final he conseguido un asistente virtual completamente funcional, una base de conocimiento cerrada solamente a Gravity, para que no responda con otras cosas, que mejora constantemente y se puede modificar o ampliar su información, gestión completa del historial de conversaciones, y una interfaz de chat moderna que funciona en tiempo real.

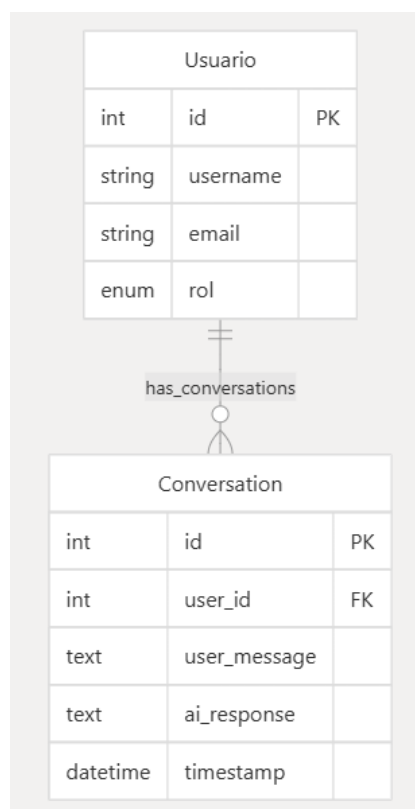


Ilustración 15 – Modelo de la tabla conversation

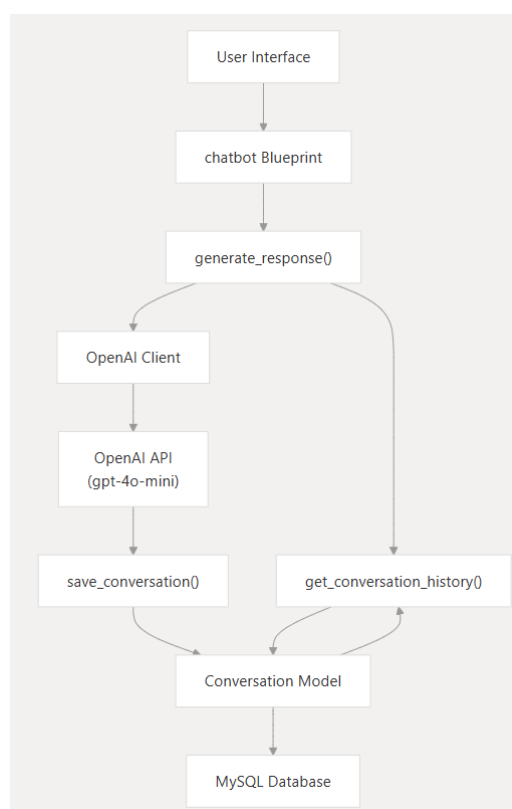


Ilustración 16 – Flujo de servicio de chatbot

10. Pruebas

Categoría	Prueba	Resultado Esperado
Autenticación	Registro válido	Usuario creado, redirección a login
	Email duplicado	Error: "Este correo ya está registrado"
	Username duplicado	Error: "Este nombre de usuario ya está en uso"
	Contraseñas no coinciden	Error: "Las contraseñas no coinciden"
	Login válido	Acceso al dashboard
	Login inválido	Error: "Credenciales incorrectas"
	Usuario baneado	Redirección a página de baneo
	Logout	Sesión cerrada, redirección a login
Dashboard	Acceso página inicio	Renderización correcta
	Carga contenido dinámico	Secciones cargadas correctamente
	Navegación entre secciones	Contenido actualizado sin recarga
Licencias	Ver licencias (usuario)	Solo licencias propias
	Ver licencias (Admin)	Todas las licencias del sistema
	Crear licencia (Admin)	Licencia creada exitosamente
	Crear sin permisos	Error 403 - Acceso denegado
	Editar licencia	Cambios guardados
	Activar con código válido	Licencia activada
	Activar código inválido	Error de código
Tickets	Crear ticket válido	Ticket creado estado "Abierto"
	Crear sin título/descripción	Error: "Datos obligatorios"
	Ver tickets (usuario)	Solo tickets propios
	Ver tickets (Admin)	Todos los tickets
	Añadir mensaje	Mensaje añadido correctamente
	Mensaje sin permisos	Error 403 - Acceso denegado
	Mensaje vacío	Error: "No puede estar vacío"
Chatbot	Mensaje válido	Respuesta generada por IA
	Mensaje vacío	Error 400 - Message cannot be empty
	Historial conversaciones	Últimas 5 conversaciones
	Pregunta sobre Gravity	Información precisa
	Pregunta fuera contexto	Redirección amable
Admin Panel	Ver usuarios	Lista completa usuarios
	Editar usuario	Cambios aplicados
	Banear/desbanear	Estado actualizado
	Eliminar usuario	Usuario y datos eliminados
	Eliminar cuenta propia	Error: No permitido
	Acceso sin permisos	Error 403

Control Acceso	Añadir transacción	Transacción registrada
	Datos incompletos	Error 400
	Ruta sin autenticación	Redirección a login
	Función Admin sin permisos	Error 403
	Usuario baneado	Página de baneo
PayPal	Proceso compra	Redirección a PayPal
	Pago exitoso	Licencia activada automáticamente
	Pago cancelado	Retorno sin activación
Configuración	Actualizar perfil	Cambios guardados
	Cambiar contraseña	Contraseña actualizada
	Eliminar cuenta	Usuario y datos eliminados
Seguridad	Hash contraseñas	Bcrypt aplicado
	Sesiones	Manejo seguro Flask
	Validación datos	WTForms validando

11. Conclusiones

A lo largo del desarrollo de la aplicación del TFG, he aprendido mucho, he disfrutado de todo el proceso, al final he conseguido una implementación exitosa de una plataforma SaaS completa que integra múltiples tecnologías. Pero no todo ha sido fácil, he tenido muchos problemas y complicaciones por el camino.

11.1 Complejidad desarrollo del Front Dinámico

La implementación de la carga dinámica de contenido con AJAX me costó bastante, tuve problemas en la sincronización entre el front y el back, especialmente en la gestión de estados de la interfaz de usuario.

11.2 Complejidad de Front responsive

Hacer todas las pantallas de mi aplicación responsive y que se vean bien en todos los dispositivos fue un reto. Al principio intenté hacerlo yo sin ningún tipo de plantilla ni nada, solo prueba y error, luego ya cogí plantillas de otras pantallas responsive y las implementé en mi aplicación.

11.3 Lecciones Aprendidas

El desarrollo de este TFG ha sido una experiencia con la que me he dado cuenta de muchas cosas y me ha permitido aprender:

- Comprendí la importancia de una buena planificación.
- Tener el código ordenado y limpio y trabajar con git.
- He aprendido mucho a investigar y adquirir conocimientos de manera independiente, buscándome la vida.

12. Posibles mejoras

Propongo diversas mejoras que abarcan desde la seguridad hasta la escalabilidad, pasando por nuevas funcionalidades, una mejor experiencia de usuario, optimizaciones y perfeccionar el funcionamiento de la IA.

12.1 Mejoras de seguridad

Autenticación de Dos Factores (2FA): Estaría bien implementar la verificación en 2 factores, sobre todo para los administradores, añadir capas de seguridad nunca está de más.

12.2 Mejoras Funcionales

Sistema de Notificaciones por Email: Implementar alertas automáticas para eventos críticos, como la expiración de licencias, nuevos tickets de soporte confirmaciones de transacciones y demás.

Dashboard Analítico: Expandir el dashboard actual, se podrían añadir algún gráficos o campos para tener información de las ventas, uso de licencias, tickets abiertos... etc. Sería más cómodo para los administradores.

Sistema de promociones y descuentos: Añadir un sistema completo de promociones, serían códigos de descuento personalizables, ofertas, cupones y así. Aprovecharía el campo de la tabla licencias "es_promocional".

12.3 Mejoras de Experiencia de Usuario

Búsqueda avanzada y filtrado: Podría mejorar las funciones de búsqueda con filtros por rango de fechas, estado de licencias, tipo de usuario... etc. Porque ahora mismo la búsqueda es general.

Sistema de archivos: Habilitar la opción de adjuntar documentos a los tickets de soporte, facilitando la comunicación y proporcionando un historial visual de los problemas.

13. Resumen Final

Bueno, al final, este proyecto ha conseguido montar una **aplicación web para gestionar licencias SaaS** que va bastante bien. La hice usando **Flask y Python**, y la verdad es que la combinación fue una buena elección. Flask me dio una base sólida y escalable para la web, y con Python pude escribir la lógica de todo de forma súper clara y eficaz.

La idea era hacer una solución que destacara en el mercado de licencias para empresas. Para eso, le metí una **interfaz que fuera fácil de usar** (la parte del diseño, vaya) y la conecté con otras apps importantes. Por ejemplo, la integré con **OpenAI** para el chatbot y con **PayPal** para que se pudieran hacer los pagos sin problema.

Aunque fue un reto grande conectar tantos sistemas a la vez, el proyecto demuestra que, con buena planificación y currándoselo, se puede sacar adelante una aplicación web completa y que funcione. Usar Flask, MySQL y todas esas integraciones externas fue clave para conseguir los objetivos y, al final, ayudar a las empresas a gestionar mejor sus licencias.

Una de las cosas que más me costó fue manejar toda la de los **datos interconectados**. En la parte de las **licencias**, tuve que pensar muy bien cómo se relacionaban los usuarios, las licencias y los pagos, y cómo se gestionaban los diferentes estados. Y con el **sistema de tickets**, también tuve que darle muchas vueltas para que todo cuadrara.

En resumen, este proyecto ha sido un desafío que me ha gustado mucho hacer. Me ha permitido aplicar todo lo que sé de desarrollo web (full-stack) y, de paso, mejorar un montón en integrar sistemas complicados, manejar bases de datos y crear APIs.

14. Bibliografía

Tecnologías

- Grinberg, M. (2024). **Flask Web Development** (3rd Edition). O'Reilly Media.
- Pallets Team. (2024). Flask Documentation v3.0. <https://flask.palletsprojects.com/>
- Oracle Corporation. (2024). MySQL 8.0 Reference Manual. <https://dev.mysql.com/doc/>

Seguridad y APIs

- OpenAI. (2024). Documentation. <https://platform.openai.com/docs/>
- PayPal Developer. (2024). PayPal REST API Documentation v2. <https://developer.paypal.com/api/rest/>

Testeos y GitHub

- Pytest Development Team. (2024). pytest Documentation v8.0. <https://docs.pytest.org/>
- GitHub. (2024). **GitHub Actions Documentation**. <https://docs.github.com/en/actions>