

PYTHONNN

Samuelson J. Cabello Padilla

PROYECTO FINAL



Samuelson J. Cabello Padilla

RESPONSABLE DEL DESARROLLO DE LA APLICACIÓN

**Fecha:** 21, MARZO, 2024

INDICE

[OBJETIVOS 3](#_Toc164939901)

[STACK TECNOLÓGICO 4](#_Toc164939902)

[DESARROLLO 5](#_Toc164939903)

[CONCLUSIONES 8](#_Toc164939904)

OBJETIVOS

Se nos pide como desarrolladores, crear una aplicación web con Python para una empresa de

suministros informáticos. Esta aplicación debe cumplir con algunos requisitos:

1. Nos piden tener inventariado todos sus productos y cuáles son sus cantidades en el

almacén, de tal forma, que cuando el stock esté al 90% nos avise de pedir al proveedor.

2. En la aplicación web sería ideal tener dos tipos de acceso, uno para clientes y otro para

nosotros, un usuario administrador que tenga acceso a todo.

3. Necesitaremos para nuestros clientes unas gráficas de ventas. Para nosotros, tendremos

unas gráficas comparativas, para saber lo que vendemos y los beneficios que sacamos de

nuestros proveedores. También se podrá buscar una alternativa para las gráficas, calculando

unas estadísticas de ventas y compras y mostrando dichos resultados.

4. Todos los productos deben tener una descripción del producto, así como lo que hay en el

almacén, su precio, lugar donde se encuentra, etc. Aquí podéis tomaros licencias sobre la

información extra que añadir, como número de referencia, colores…

5. Para los proveedores, debemos tener almacenados todos los datos de contacto (nombre de

empresa, teléfono, dirección, cif…), facturación, precios de sus productos, porcentaje de

descuento, IVA, etc.

6. Debemos elaborar la aplicación web de la forma más sencilla para el usuario y lo más práctica

para nosotros en su manejo y obtención de datos importante para la empresa. Hay que

tener en cuenta la Experiencia del Usuario la cual se caracteriza por sencillez, claridad,

intuición.

STACK TECNOLÓGICO

De las opciones que manejaba para el framework he tenido en cuenta 3, a saber:

* Django, por lo completo que es y la riqueza de su documentación, así como la gran comunidad que se ha construido a lo largo de los años.
* Flask, por su ligereza y simplicidad.
* Reflex, por lo nueva que es la tecnología y lo personalizable que es.

Tras las reflexiones oportunas me terminé decidiendo por Django como mi framework de desarrollo web principalmente debido a su enfoque en la productividad y la seguridad. Django viene con una serie de características integradas, como autenticación de usuarios, ORM y administración de sesiones, lo que acelera significativamente el proceso de desarrollo. Además, su robusta arquitectura escalable lo hace adecuado para proyectos de cualquier tamaño. La seguridad es otra razón importante para elegir Django. Viene con medidas integradas para proteger contra vulnerabilidades comunes como CSRF y XSS, lo que proporciona una capa adicional de protección para las aplicaciones web que desarrollo. La comunidad activa de Django y su documentación completa también fueron factores decisivos para mí. La comunidad ofrece una gran cantidad de recursos, desde paquetes adicionales hasta tutoriales y soporte en línea. Y la documentación oficial es exhaustiva y fácil de seguir, lo que facilita el aprendizaje y la resolución de problemas.

En general, la combinación de productividad, seguridad, comunidad activa y documentación completa hacen de Django una opción atractiva y confiable para el desarrollo web en Python.

Para la base de datos he optado por sqlite3 porque ya viene integrada por defecto en Django lo cual agiliza todo el proceso de implementación.

Por último, he escogido Visual Studio Code como IDE para realizar el proyecto, ya que, quería aprender a usar el entorno de desarrollo más utilizado en todo el mundo por desarrolladores. A parte de que cuenta con una gran variedad de extensiones que facilitan las labores de desarrollo y visualización.

DESARROLLO

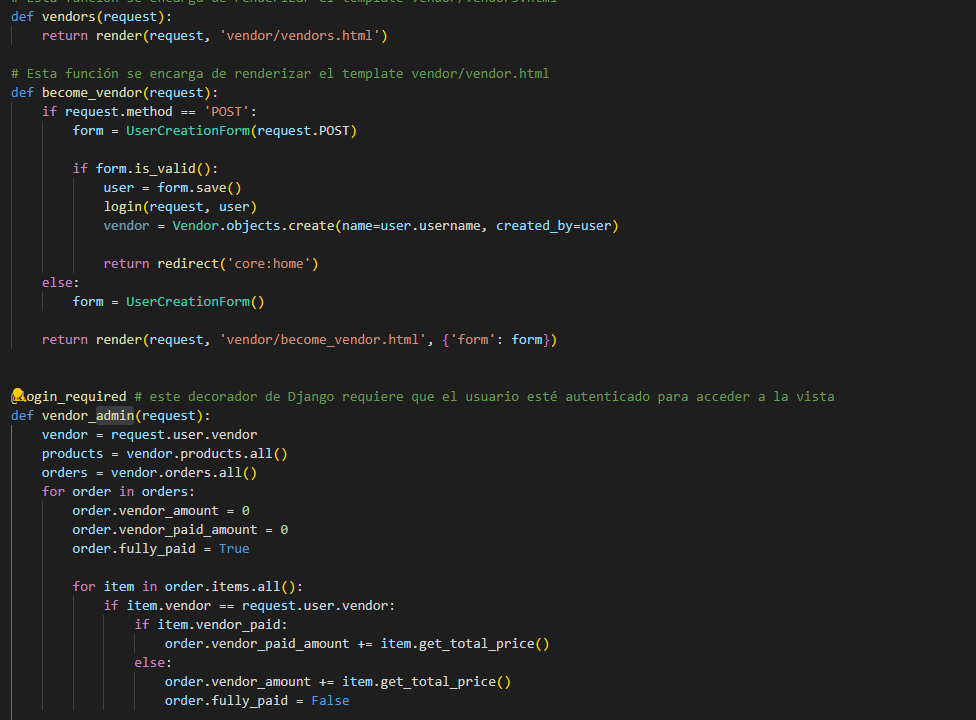
1. Planificación y Diseño de la aplicación:

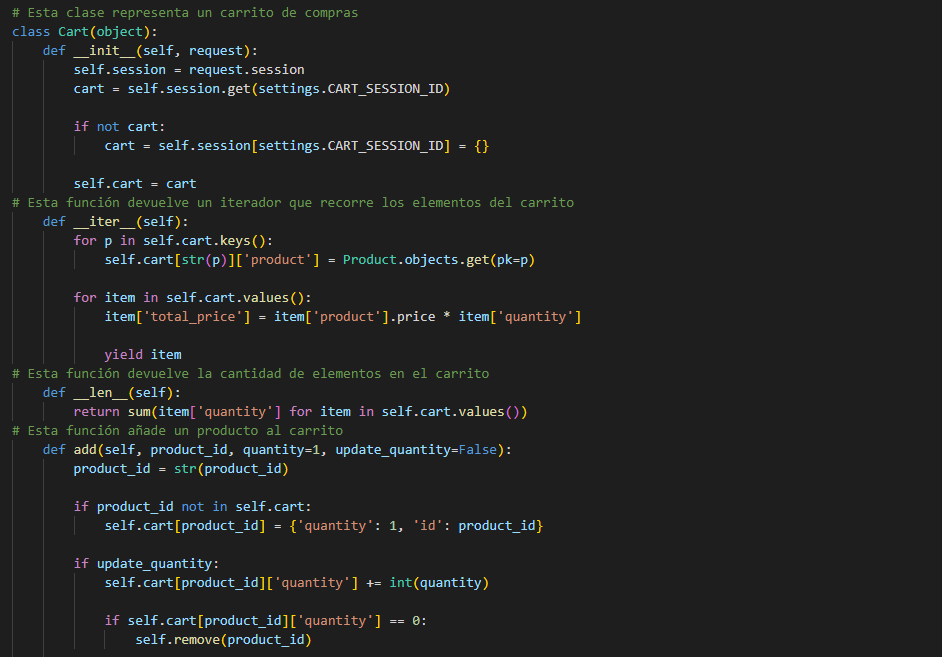
Previamente a codificar, es crucial planificar la estructura y funciones que quería implementar en mi aplicación. Tras analizar los requisitos básicos del proyecto planteé un diseño básico de la interfaz de usuario. Quise hacer algo sencillo e intuitivo.

Al ser una aplicación para proveedores y compradores fue básico establecer desde el principio como iba a ser la base de datos y sus relaciones, así como los productos y sus categorías.

1. Desarrollo del Proyecto:

El primer reto del proyecto fue aprender Django desde cero, supuso una complicación por no haberlo tratado en el curso, así como venir de un framework más sencillo como lo es Flask. Aún así hay una extensa documentación, así como una densa comunidad que aportan tanto conocimiento como respuesta a las dudas que surgen con esta tecnología.

Una vez analizada la documentación y empezado el proyecto el primer paso fue comenzar a diseñar las vistas(views) para poder manejar las solicitudes HTTP y renderizar las plantillas HTML (carpeta “templates”=). En las siguientes imágenes pondré ejemplos de funciones.



Mapear las solicitudes a su vista correspondientes configurando las URLs.



1. Stripe:

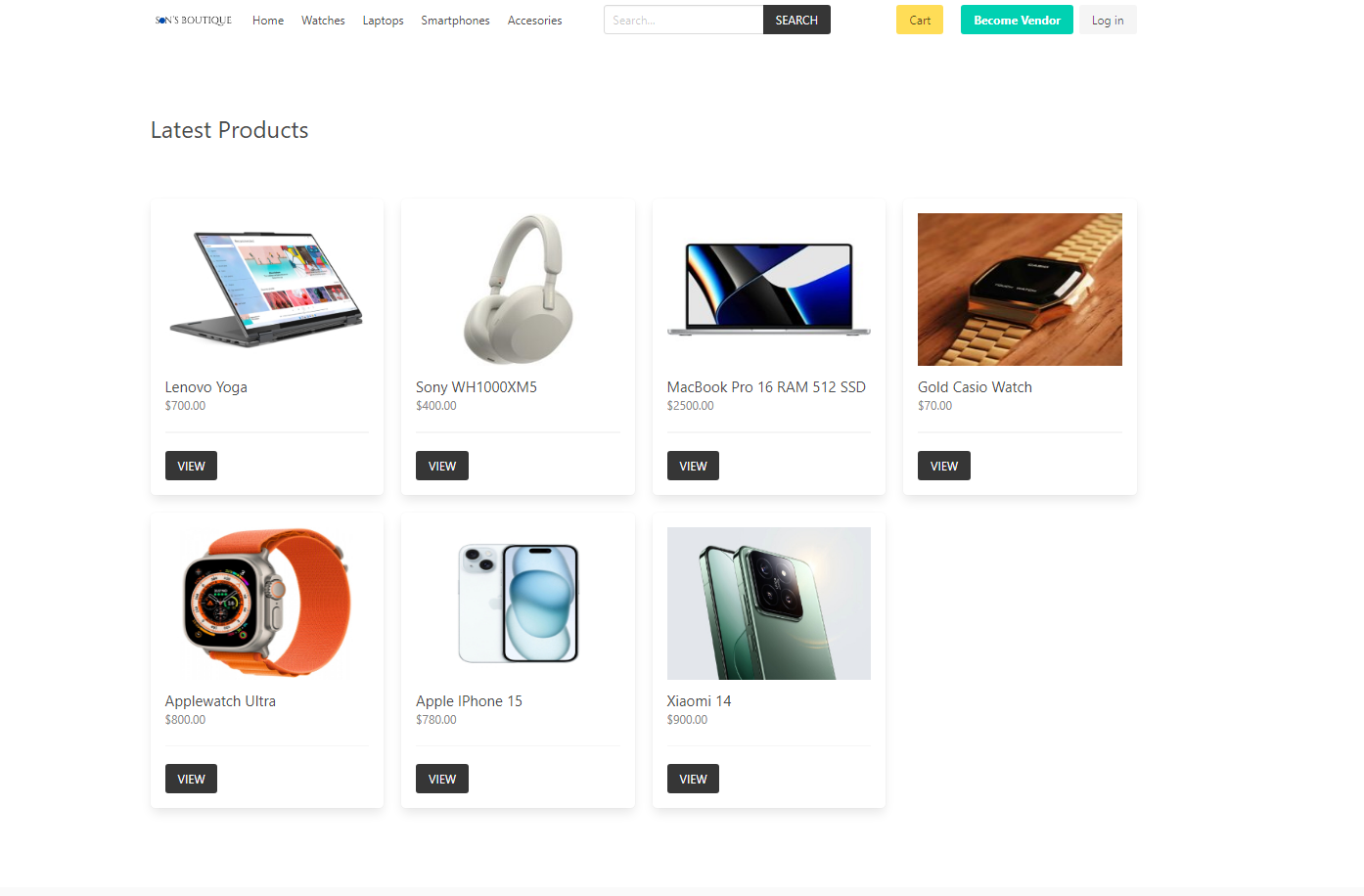
Para el método de pago me decidí por usar la API de Stripe para Python, pues tiene su entera documentación en GitHub y no suele crear conflicto con otras librerías de Python, en concreto Django, asunto de vital importancia para el proyecto.

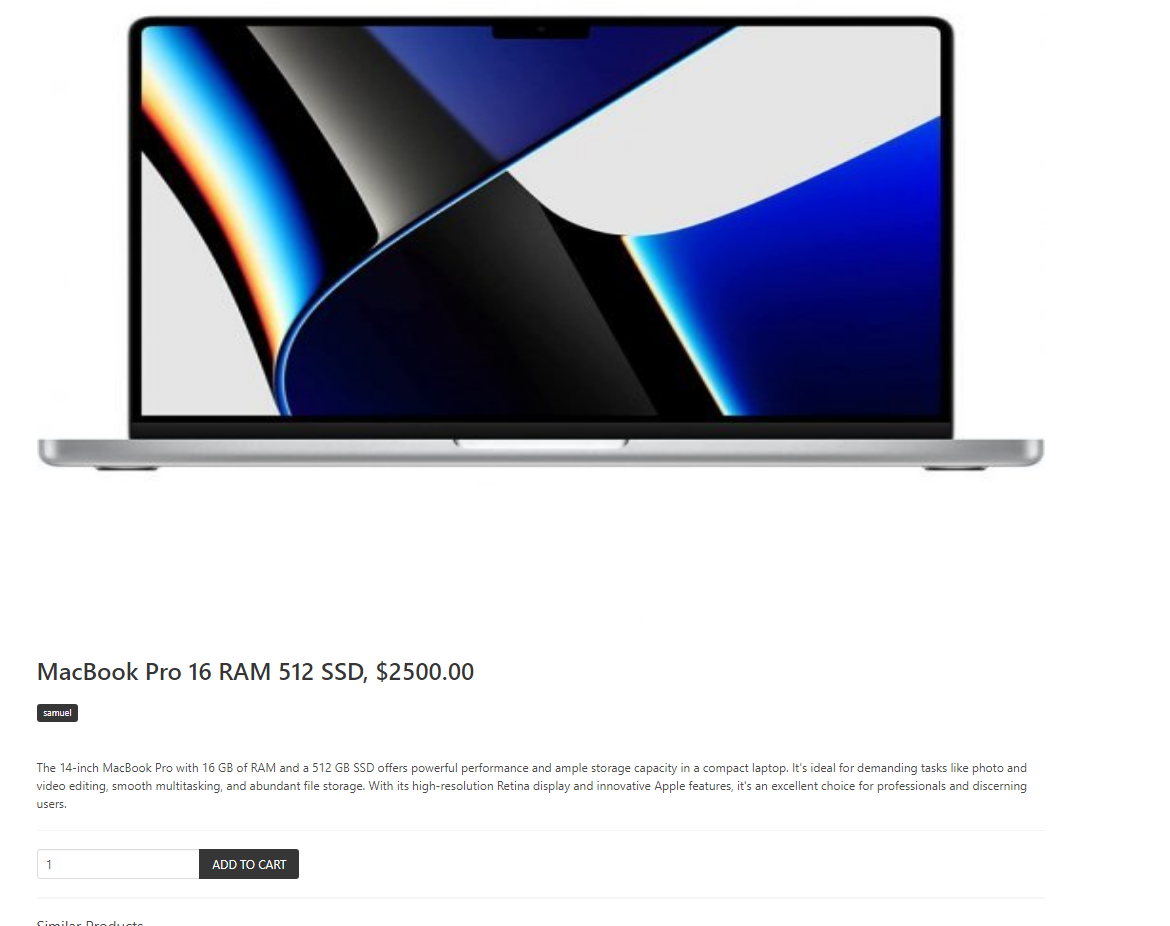
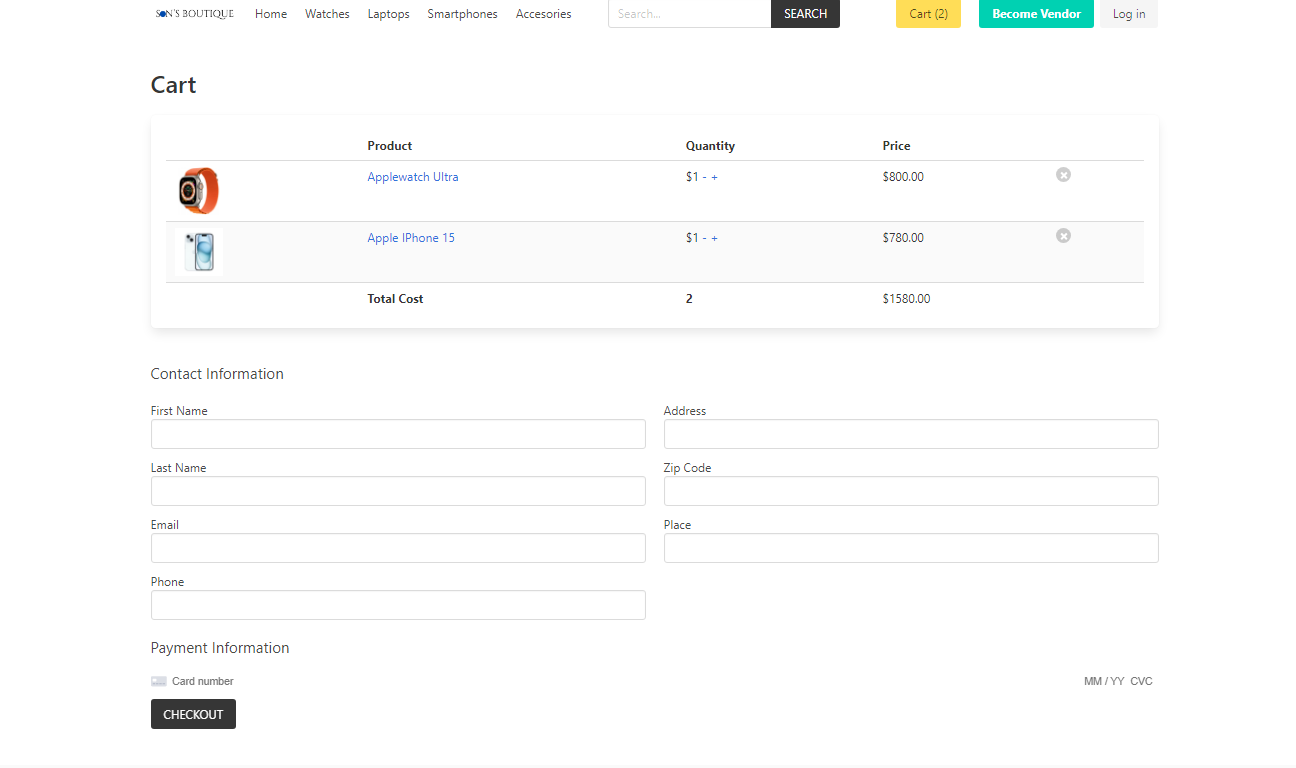


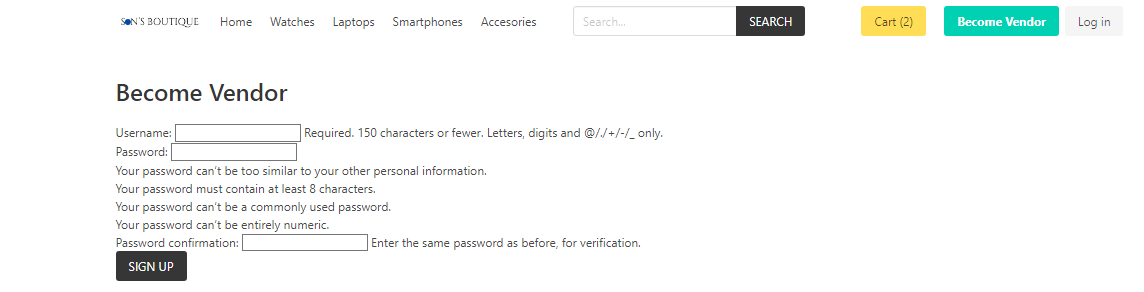
Creé vistas y las funciones pertinentes para manejar los pagos, implementé la integración de Stripe en la plantilla HTML para que se mostrasen tanto los botones de compra como los formularios de pago necesarios (imagen anterior).

1. Despliegue:

El despliegue (siempre en local) fue una herramienta importantísima en todo el desarrollo de la aplicación. Tanto para hacer pruebas como para configurar todo el aspecto gráfico de la app.





1. Pruebas y Depuración:

A la hora de las pruebas y depuración de la aplicación el problema más recurrente fue la relación de la aplicación con la base de datos y los elementos gráficos de las páginas web, así como su responsividad.

Utilicé las herramientas de depuración de Visual Studio Code así como una navegación exhaustiva en la aplicación desplegada para comprobar que todas las funciones y vistas funcionaban correctamente.

CONCLUSIONES

El desarrollo del proyecto presentó desafíos al aprender Django desde cero, pero la abundante documentación y la comunidad activa proporcionaron un sólido soporte. La planificación detallada del diseño de la aplicación fue esencial para establecer una estructura clara y funcionalidades bien definidas.

La integración de la API de Stripe para el procesamiento de pagos agregó una capa adicional de funcionalidad a la aplicación. Esta elección se basó en la robustez y documentación de Stripe, lo que facilitó su implementación en el proyecto.

El despliegue local fue una etapa crucial para realizar pruebas exhaustivas y ajustar el aspecto visual de la aplicación. Utilizando Visual Studio Code como IDE, pude aprovechar su versatilidad y las numerosas extensiones disponibles para facilitar el desarrollo.

Durante las pruebas y la depuración, surgieron desafíos típicos relacionados con la interacción de la aplicación con la base de datos y la apariencia de los elementos gráficos. Sin embargo, con herramientas de depuración adecuadas y un enfoque meticuloso, pude resolver estos problemas de manera efectiva.

En resumen, la combinación de Django, Stripe y Visual Studio Code demostró ser sólida y confiable para el desarrollo de la aplicación web. El proyecto se benefició de un enfoque planificado, una cuidadosa selección de tecnologías y una ejecución diligente, lo que resultó en una aplicación funcional y segura.