

Proyecto *Pybuys*



DESARROLLO APLICACIONES MULTIPLATAFORMA (DAM)

Realizado por

Víctor González Cobos

Tutores: Miguel Ángel González Santos y Gregorio Granja Granja

Fecha entrega: 25/05/2023



Índice

1	Estudio del problema y análisis del sistema.....	2
1.1	Descripción del software	2
1.2	Motivación y finalidad	4
1.3	Requisitos	4
1.4	Herramientas técnicas.....	6
2	Recursos necesarios para el desarrollo	6
2.1	Para el desarrollo	6
2.2	Para su ejecución.....	7
2.2.1	Servidor	7
2.2.2	Cliente	7
3	Desarrollo del proyecto	7
3.1	Estimación de tiempos	7
3.2	Trabajo realizado.....	8
3.2.1	Análisis.	8
3.2.2	Diseño	8
3.2.3	Codificación	8
3.2.4	Pruebas	8
3.3	Diseño de datos	9
3.4	Diagrama de clases.....	10
3.5	Diagrama de flujo	11
3.6	Interacción con el usuario	12
3.6.1	Casos de uso.....	12
3.7	Diseño de la interfaz.....	28
3.8	Código fuente.....	33
4	Fase de pruebas	41
5	Conclusiones finales	42
6	Documentación del sistema desarrollado.....	43
6.1	Manual de instalación.....	43
6.2	Manual de uso.....	43
7	Bibliografía	43
8	Anexos	44

1 Estudio del problema y análisis del sistema

1.1 Descripción del software

Resumen:

PyBuys es una solución de comercio electrónico integral diseñada para satisfacer las necesidades de las pequeñas y medianas empresas. Con PyBuys, estas empresas pueden establecer su presencia en línea de manera rápida y eficiente, ofreciendo sus productos a una audiencia más amplia.

Detalles:

Desarrollado sobre la sólida plataforma de Django y utilizando Python como lenguaje de programación, PyBuys ofrece una amplia gama de características que se adaptan a las necesidades de las empresas en crecimiento. Las características principales incluyen la gestión de cuentas de usuario, la navegación y búsqueda de productos, la gestión del carrito de compras, así como la posibilidad de añadir, modificar y eliminar productos.

Una de las fortalezas de PyBuys radica en su robusto sistema de gestión de cuentas. Permite a los nuevos usuarios crear cuentas fácilmente, iniciar sesión y modificar la información de su cuenta con facilidad. Además, la plataforma valida la información ingresada por los usuarios para asegurar su veracidad y seguridad.

El sistema de gestión de productos de PyBuys es otra característica destacada. No sólo permite a los usuarios buscar y seleccionar productos, sino que también permite a los administradores y al personal gestionar el inventario de manera eficaz. Esto incluye la adición de nuevos productos, la modificación de la información de los productos existentes y la actualización de las existencias de los productos.

También ofrece la búsqueda de productos por su nombre, y una gran gestión de compras y ventas para el staff y administradores.

Pybuys ofrece una increíble modularidad que permite a cualquier pequeña o mediana empresa iniciar su camino como e-commerce en cuestión de segundos, si bien está enfocada a productos tecnológicos nada impide que pueda ser usada para cualquier otro tipo de productos.

Ha sido desarrollada en Python, ofrece una solución para guardar toda tu información en MySQL (fácilmente migrable a cualquier otro tipo de bdd gracias a su modularidad). Ha sido creada para que el servidor tenga Windows.

Campos de aplicación y conexión con el ciclo:

Los campos de aplicación de PyBuys son variados y amplios, centrados principalmente en el sector del comercio electrónico. Esta plataforma puede ser utilizada por cualquier pequeña o mediana empresa (PYME) que desee establecer o expandir su presencia en línea. Esto puede abarcar una amplia gama de industrias, desde la moda y los electrodomésticos hasta la tecnología y los libros, entre otros.

El potencial de PyBuys también se extiende a empresas que buscan digitalizar sus operaciones internas, como la gestión de inventario y el seguimiento de las ventas. La plataforma puede ayudar a estas empresas a optimizar sus operaciones, mejorar la eficiencia y reducir los errores.

En términos de la conexión con el ciclo de formación de Desarrollo de Aplicaciones Multiplataforma (DAM), PyBuys se alinea estrechamente con varias áreas clave de estudio.

- **Back end:** El sistema se basa en Python y Django, lo que permite la implementación de una lógica de negocio sólida, la gestión de las interacciones del usuario, y la conexión con la base de datos. PyBuys utiliza MySQL para almacenar y gestionar los datos de los usuarios, los productos y las transacciones, ofreciendo a los estudiantes la oportunidad de aplicar y expandir sus conocimientos en el manejo de bases de datos.
- **Front end:** PyBuys se centra en proporcionar una interfaz de usuario intuitiva y atractiva, lo que requiere habilidades de desarrollo front end. Aunque Django maneja gran parte del trabajo de back end, los desarrolladores de PyBuys necesitan entender cómo presentar los datos a los usuarios de una manera que sea fácil de entender y de navegar.
- **Full stack:** Finalmente, PyBuys representa un ejemplo de desarrollo full stack, ya que combina tanto el desarrollo de front end como de back end para crear una aplicación web completa.

1.2 Motivación y finalidad

El comercio electrónico ha experimentado un crecimiento exponencial en los últimos años, y se espera que continúe en esta trayectoria en el futuro previsible. Sin embargo, a pesar del gran potencial que ofrece el comercio electrónico, muchas pequeñas y medianas empresas (PYMES) todavía luchan por establecer su presencia en línea y aprovechar este canal de ventas.

La motivación detrás de PyBuys es ayudar a estas PYMES a superar los desafíos asociados con la creación de una tienda en línea y facilitar su entrada al mercado digital. A través de su interfaz amigable, su conjunto de características bien pensadas y su flexibilidad de configuración, PyBuys permite a las PYMES establecer su tienda en línea de manera rápida y eficiente, sin necesidad de tener un conocimiento técnico profundo.

La finalidad de PyBuys es ser un catalizador para el crecimiento de las PYMES en el espacio digital. Al proporcionar una solución de comercio electrónico asequible y fácil de usar, PyBuys puede ayudar a estas empresas a aumentar su alcance de mercado, mejorar sus ventas, y finalmente, mejorar su rentabilidad.

Además, PyBuys está diseñado para ser escalable y adaptable a las necesidades cambiantes de las empresas. A medida que la empresa crece, PyBuys puede crecer con ella, lo que permite a las PYMES mantener su tienda en línea actualizada y relevante para su base de clientes en constante expansión.

En resumen, la motivación y finalidad de PyBuys es ser el socio de confianza para las PYMES en su viaje de comercio electrónico, ofreciéndoles una plataforma robusta y versátil que puede facilitar su transición al espacio digital y apoyar su crecimiento continuo.

1.3 Requisitos

- **Requisito de Sistema (RS1):** El sistema debe permitir a los nuevos usuarios crear su propia cuenta.
 - Requisito Funcional (RF1): La plataforma debe validar que el correo electrónico ingresado tenga un formato válido y que no esté ya en uso.
 - Requisito No Funcional (RNF1): La creación de la cuenta debe ser rápida y no debe demorar más de unos segundos.

- **RS2:** El sistema debe permitir a los usuarios registrados iniciar sesión.
 - Requisito Funcional (RF2): La plataforma debe verificar que la combinación de correo electrónico y contraseña sea correcta antes de permitir el inicio de sesión.
- **RS3:** El sistema debe permitir a los usuarios modificar la información de su cuenta.
 - Requisito Funcional (RF3): La plataforma debe validar que todos los campos de información del usuario sean completados antes de permitir la modificación.
 - Requisito No Funcional (RNF3): Las modificaciones a la cuenta del usuario deben ser guardadas y reflejadas inmediatamente en la plataforma.
- **RS4:** El sistema debe permitir a los usuarios agregar, modificar y eliminar productos de su carrito.
 - Requisito Funcional (RF4): La plataforma debe verificar que hay stock suficiente de un producto antes de permitir que se añada al carrito.
 - Requisito No Funcional (RNF4): Las modificaciones al carrito de un usuario deben ser guardadas y reflejadas inmediatamente en la plataforma.
- **RS5:** El sistema debe permitir a los usuarios del staff o administradores añadir nuevos productos.
 - Requisito Funcional (RF5): La plataforma debe validar que todos los campos de información del producto sean completados antes de permitir la adición de un nuevo producto.
 - Requisito No Funcional (RNF5): La adición de nuevos productos debe ser registrada y reflejada inmediatamente en la plataforma.
- **RS6:** El sistema debe permitir a los usuarios del staff o administradores añadir o quitar stock.
 - Requisito Funcional (RF6): La plataforma debe validar que la cantidad de stock a añadir o quitar sea un número válido y que no exceda la cantidad de stock disponible.
 - Requisito No Funcional (RNF6): Las modificaciones al stock de un producto deben ser registradas y reflejadas inmediatamente en la plataforma.
- **RS7:** El sistema debe permitir a los usuarios del staff o administradores visualizar las compras de stock realizadas.
 - Requisito Funcional (RF7): La plataforma debe mostrar todas las compras realizadas en orden cronológico, incluyendo la fecha, producto y cantidad.
 - Requisito No Funcional (RNF7): La visualización de las compras de stock debe ser actualizada en tiempo real para reflejar las últimas transacciones.

- **RS8:** El sistema debe permitir a cualquier usuario registrado buscar un producto.
 - Requisito Funcional (RF8): La plataforma debe proporcionar resultados de búsqueda relevantes basados en el término de búsqueda ingresado por el usuario.
 - Requisito No Funcional (RNF8): El sistema de búsqueda debe ser rápido y debe proporcionar resultados en tiempo real a medida que el usuario escribe su búsqueda.

1.4 Herramientas técnicas

Se usará Python como lenguaje de programación, trabajando sobre el framework de django.

Para el desarrollo web se usó javascript, css y html, concretamente se usó el framework de css TAILWIND.

Como IDE se usará Visual Studio Code.

Para lograr una conexión a bdd y gestionarlo se usará MySQLWorkbench.

Se usará GIMP para cualquier edición de imagen requerida.

Se usará Git y Github como herramienta de versionado.

Se utilizarán herramientas de edición de documentos:

- Excel
- Word

Y como herramienta para grabar el manual de usuario se usará el OBS.

2 Recursos necesarios para el desarrollo

2.1 Para el desarrollo

Hardware:

- Sistema Operativo: Windows 10
- Procesador: Intel Core i5
- Memoria: 12 GB de RAM
- Almacenamiento: 100 GB de espacio libre en disco

Software:

- Sistema Operativo: Windows 10
- Python 3.8 o superior
- MSSQL Server Express Edition
- Dependencias instaladas: Django, django-mssql-backend, Pillow

2.2 Para su ejecución

2.2.1 Servidor

Hardware:

- Sistema Operativo: Windows 10 o superior.
- Procesador: CPU de 2 o 3 núcleos
- Memoria: 4 GB de RAM
- Almacenamiento: 10 GB de espacio libre en disco

Software:

- Sistema Operativo: Windows 10 o superior
- Python 3.7 o superior
- MSSQL Server Express Edition
- Dependencias instaladas: Django, django-mssql-backend, Pillow

2.2.2 Cliente

Hardware:

- Navegador web actualizado (Chrome, Firefox, Safari, etc.)

Software:

- Navegador web actualizado (Chrome, Firefox, Safari, etc.)

3 Desarrollo del proyecto

3.1 Estimación de tiempos

Fase de desarrollo	Tiempo estimado	Tiempo realizado
Análisis	13	10
Diseño	17	20
Codificación	24	20
Pruebas	27	10
Total	81	60

Para desarrollo total del proyecto he necesitado alrededor de unas 60 horas totales.

3.2 Trabajo realizado.

3.2.1 Análisis.

Recoger requisitos, plantear herramientas a usar y estimar tiempos. Elegir framework a usar y elección de lenguaje de programación.

3.2.2 Diseño

Diseñar base de datos, diagrama de clases, casos de uso. Estructurar como iba a ser la codificación en una aplicación de django y como podía adaptar los requisitos a este framework.

Diseño de interfaces y creación de logo.

3.2.3 Codificación

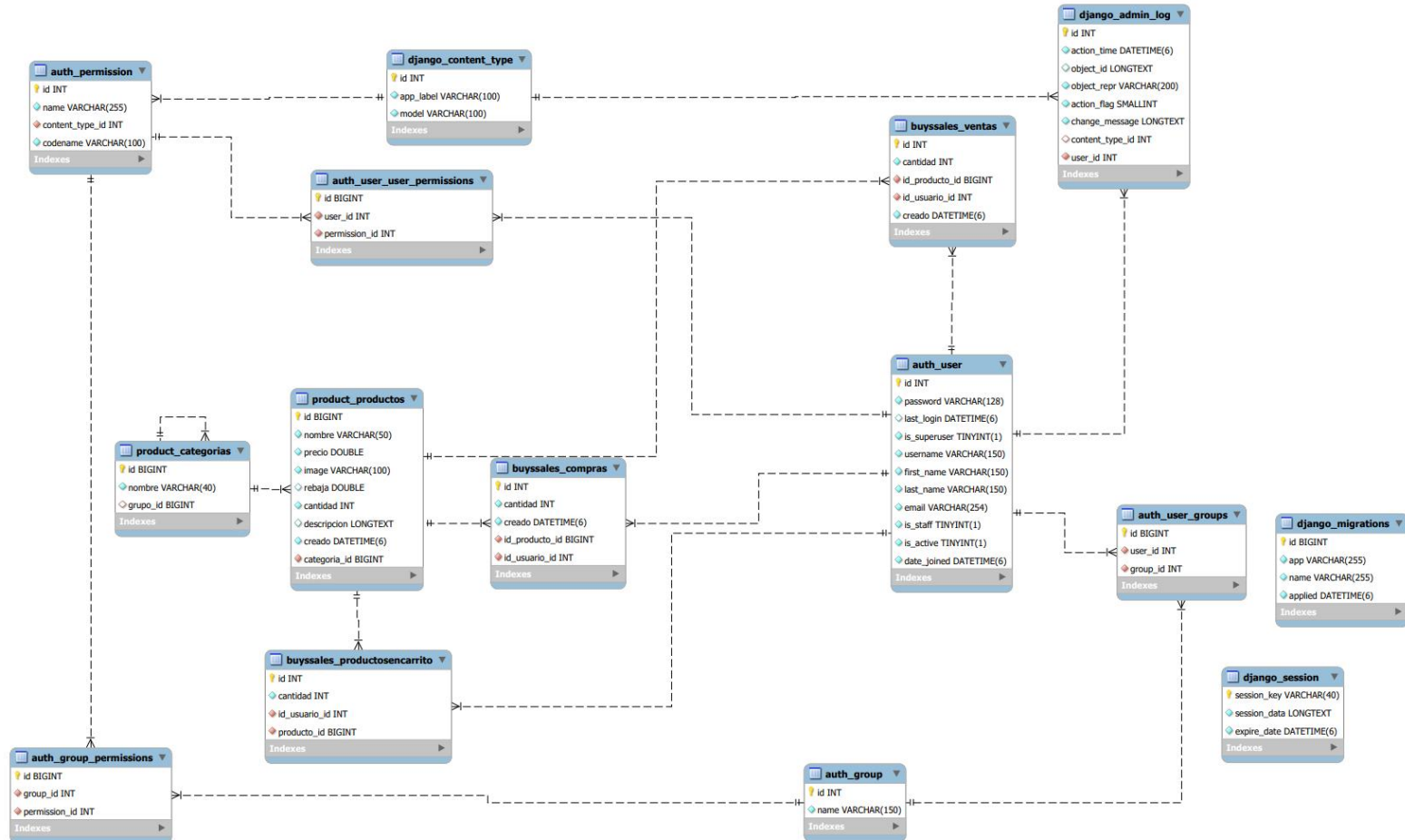
Implementación de funcionalidades al código. Adaptar apartado de administrador para las funcionalidades. E implementar interfaces diseñadas con anterioridad.

3.2.4 Pruebas

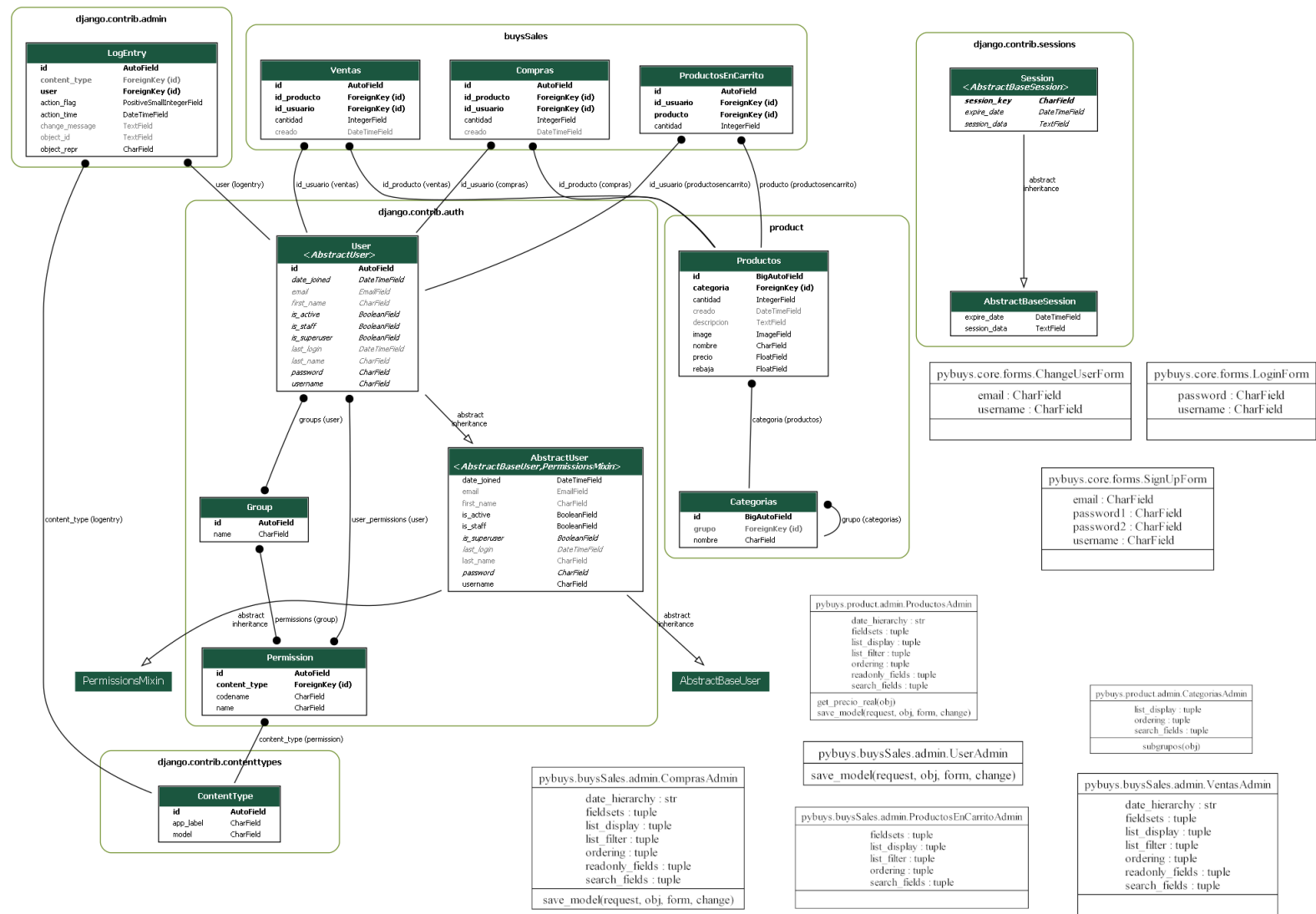
Probadas todas las funcionalidades implementadas. Hechas pruebas unitarias en código y pruebas integradas, de pico, estabilidad y estrés documentadas en Excel.

3.3 Diseño de datos

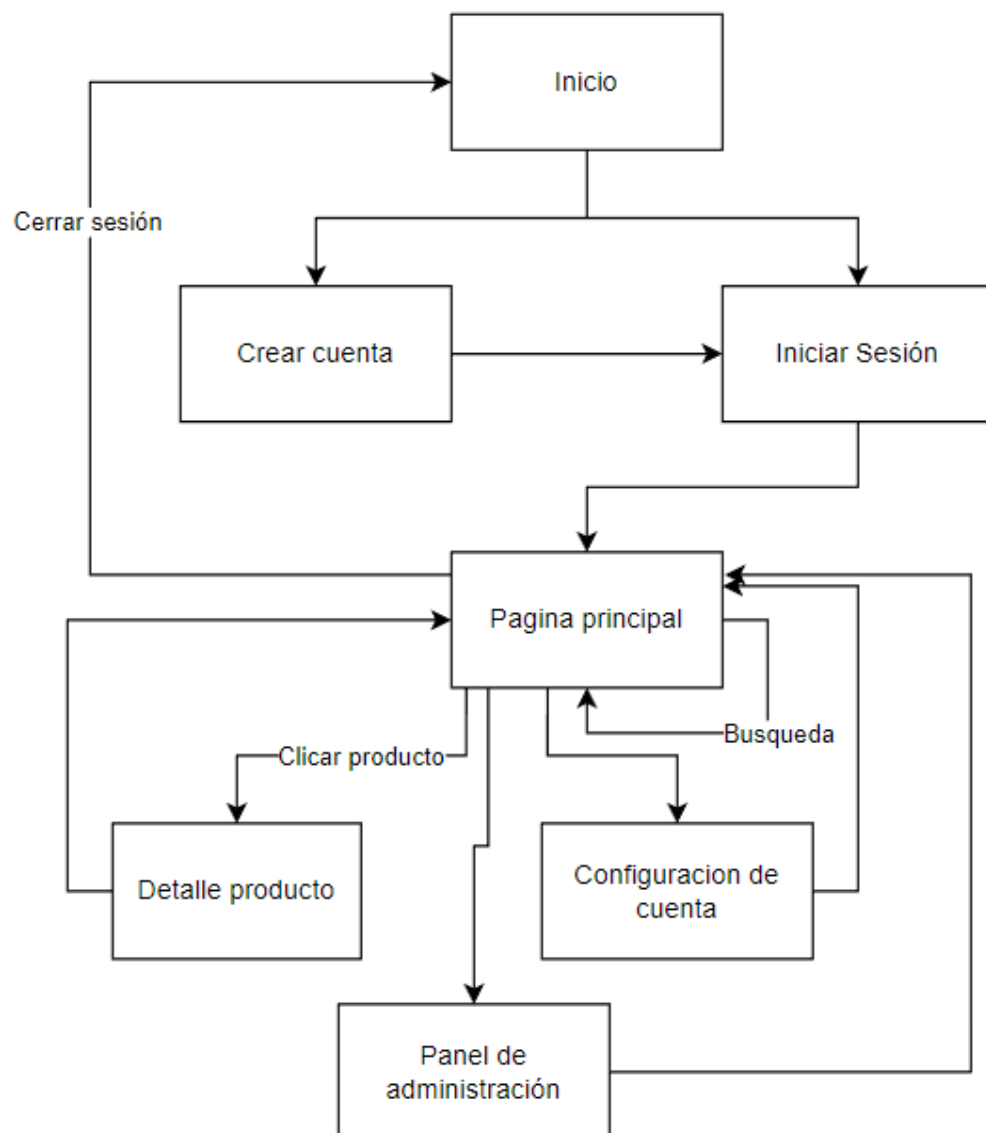
Diagrama relacional



3.4 Diagrama de clases



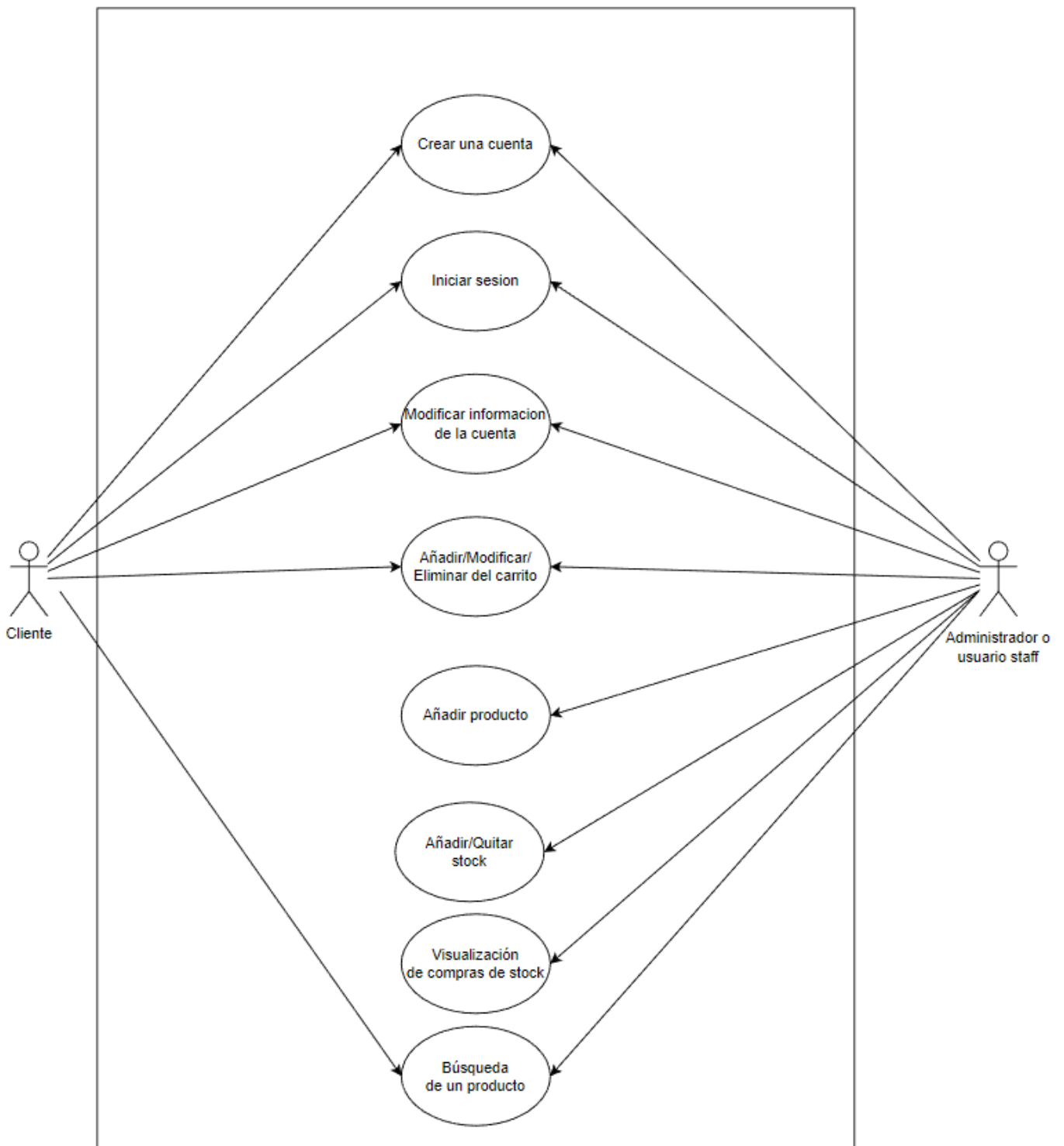
3.5 Diagrama de flujo



3.6 Interacción con el usuario

3.6.1 Casos de uso

Diagrama de casos de uso



Casos de uso detallados

	CU1 – Crear cuenta
Descripción	Un usuario nuevo podrá crear su propia cuenta. Al iniciar sesión o registrarse, el usuario es redirigido a la página de inicio.
Actores	Nuevo usuario
Precondiciones	Ninguna
Pasos del caso de uso	<ol style="list-style-type: none">1- El nuevo usuario ingresa su nombre de usuario, dirección de correo electrónico, contraseña y repite la contraseña en la pantalla de registro.2- El usuario hace clic en "Crear cuenta".3- Si algún campo obligatorio está vacío o no se cumple con los requisitos básicos (el formato del correo electrónico), se muestra un mensaje de error correspondiente y se solicita al usuario que complete correctamente los campos.4- Si todos los campos están vacíos y se cumplen requisitos básicos, se envía la información para que el servidor la procese.5- Si el correo electrónico o el nombre de usuario ya existen en el sistema, se muestra un mensaje de error indicando que ya están en uso. El usuario es redirigido a la página de crear cuenta.6- Si la contraseña no cumple con los requisitos de seguridad establecidos en Django (por ejemplo, debe contener al menos 8 caracteres, incluir letras y números), se muestra un mensaje de error y se solicita al usuario que elija una contraseña más segura. El usuario es redirigido a la página de crear cuenta.7- Si todos los campos se completan correctamente y la contraseña cumple con los requisitos, se crea la cuenta del usuario y se le redirige a la página de inicio de sesión.

	CU2– Inicio de sesión
Descripción	El usuario introduce sus datos previamente registrados para poder acceder.
Actores	Usuario registrado
Precondiciones	El usuario debe tener una cuenta registrada previamente para poder iniciar sesión con éxito.
Pasos del caso de uso	<ol style="list-style-type: none"> 1- El usuario ingresa su nombre de usuario y contraseña en la pantalla de inicio de sesión. 2- El usuario hace clic en "Iniciar sesión". 3- El servidor procesa la información. 4- Si la combinación de nombre de usuario y contraseña es incorrecta, se muestra un mensaje de error indicando que la contraseña es incorrecta. El usuario puede intentar nuevamente ingresando la contraseña correcta. 5- Si el nombre de usuario y la contraseña coinciden con los registros del sistema, el inicio de sesión tiene éxito y el usuario es redirigido a la página de inicio de la aplicación.

	CU3 – Modificar información de la cuenta
Descripción	El usuario modifica los datos de su cuenta.
Actores	Usuario registrado
Precondiciones	Tener una cuenta creada y estar con la sesión iniciada
Pasos del caso de uso	<ol style="list-style-type: none"> 1- El usuario inicia sesión en la página web. 2- En la interfaz de usuario, el usuario selecciona la opción "Configuración de cuenta". 3- Se muestra un formulario con los datos actuales del usuario, incluyendo su nombre y dirección de correo electrónico. 4- El usuario realiza las modificaciones necesarias en los campos que desea cambiar. 5- Si el correo no es correcto no permite enviar la información. 6- Al intentar guardar los cambios, se realizan las siguientes validaciones: 7- a. Si algún campo está vacío, se muestra un mensaje de error indicando que todos los campos deben ser completados. 8- b. Si el usuario intenta cambiar su dirección de correo electrónico y esa dirección ya está en uso por otro usuario, se muestra un mensaje de error indicando que la dirección de correo electrónico ya está en uso. 9- Si todas las validaciones son exitosas, se actualizan los datos de la cuenta del usuario en la base de datos. 10- Se muestra un mensaje de éxito indicando que los cambios se han guardado correctamente. 11- El usuario puede continuar utilizando la aplicación con los datos actualizados en su cuenta.

	CU4 – Añadir/Modificar/ Eliminar del carrito
Descripción	El usuario hace operaciones en el carrito.
Actores	Usuario con sesión iniciada.
Precondiciones	El usuario debe haber iniciado sesión en el sistema.
Pasos del caso de uso	<ol style="list-style-type: none"> 1- El usuario navega a la ventana de detalle de un producto y desde allí tiene la opción de añadir el producto a su carrito si hay stock disponible. 2- El usuario puede acceder a la ventana del carrito, donde puede realizar modificaciones en los elementos añadidos, como modificar la cantidad (+, -), siempre y cuando haya stock disponible. 3- Después de realizar las modificaciones deseadas, el usuario selecciona el botón de "Actualizar carrito" para confirmar los cambios. 4- Si el usuario decide eliminar un producto del carrito, puede seleccionar la opción correspondiente para eliminarlo por completo. 5- El usuario puede continuar navegando por la página y repetir los pasos anteriores para añadir, modificar o eliminar más elementos en su carrito. 6- Cuando el usuario está satisfecho con los elementos en su carrito y desea proceder con la compra, selecciona la opción de "Comprar". 7- Una vez finalizada la compra, se genera una venta con los productos seleccionados. 8- El usuario recibe una notificación de compra realizada.

	CU5 – Añadir producto
Descripción	El usuario del staff o administrador podrá añadir nuevos productos
Actores	Usuario del staff o administrador
Precondiciones	Haber iniciado sesión y entrado en el panel de administrador.
Pasos del caso de uso	<ol style="list-style-type: none"> 1- El usuario del staff o administrador selecciona la opción "Añadir producto" desde el panel de administrador. 2- Se muestra un formulario que permite al usuario ingresar los campos básicos del producto, como nombre, descripción, precio y cantidad disponible. Todos los campos esenciales deben ser completados para poder continuar. 3- El usuario completa todos los campos requeridos y selecciona la opción "Confirmar" o "Guardar". 4- Se verifica si toda la información necesaria está completa. Si falta algún campo esencial, se muestra un mensaje de error indicando qué campo debe ser completado. 5- Si toda la información requerida está completa, se guarda el producto en el sistema y se realiza una compra inicial con la cantidad especificada. Esto significa que se añade el producto al inventario y se registra una transacción de compra con la cantidad ingresada. 6- Una vez que se ha guardado el producto exitosamente, se muestra un mensaje de confirmación y se redirige al usuario a la página de gestión de productos.

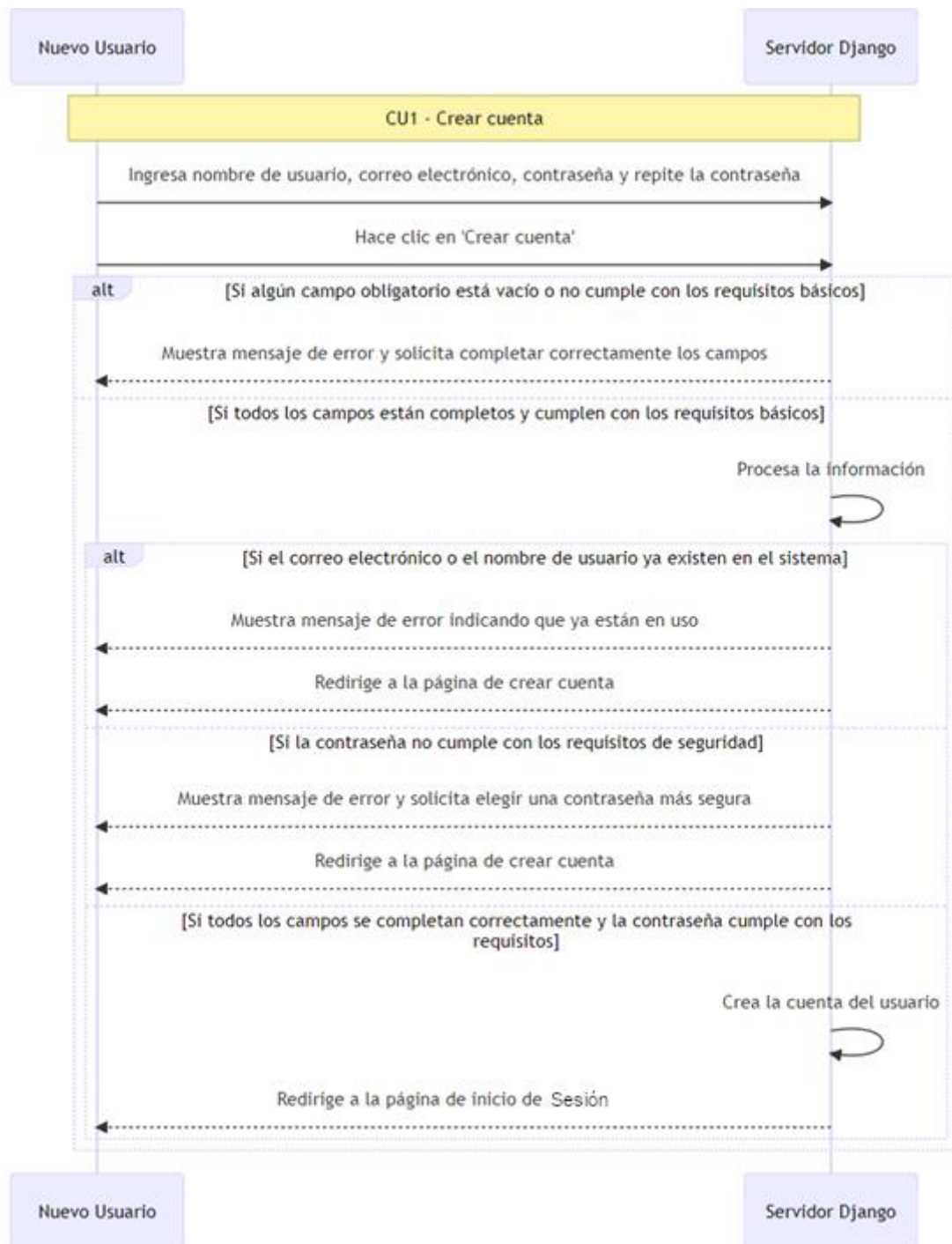
	CU6 – Añadir/Quitar stock
Descripción	El usuario del staff o administrador podrá añadir nuevos productos
Actores	Usuario del staff o administrador
Precondiciones	El usuario debe haber iniciado sesión como staff o administrador.
Pasos del caso de uso	<ol style="list-style-type: none"> 1- El usuario del staff o administrador selecciona un producto y accede a su detalle. 2- En la página de detalle del producto, se muestra un contador que refleja la cantidad de stock actual. El contador tiene un valor mínimo igual a la cantidad de stock actual en negativo, lo que permite realizar operaciones para quitar stock. 3- El usuario introduce la cantidad adecuada de stock que desea añadir o quitar en el contador. 4- El usuario hace clic en "Añadir stock" para confirmar la operación. 5- Si se añade o quita el stock correctamente, se muestra un mensaje de confirmación indicando que la operación se ha realizado exitosamente. 6- Si se intenta realizar una operación inválida, por ejemplo, si un cliente compra una cantidad mayor a la que se intenta quitar, se muestra un mensaje de error. En este caso, el usuario puede actualizar la página y repetir los pasos de nuevo.

	CU7 – Visualización de compras de stock
Descripción	El usuario del staff o administrador podrá ver las compras de stock realizadas
Actores	Usuario del staff o administrador
Precondiciones	Haber iniciado sesión y entrar en el apartado de "Compras"
Pasos del caso de uso	<ol style="list-style-type: none"> 1- El usuario del staff ingresa al apartado de "Compras" y puede ver todas sus compras realizadas en orden cronológico. Las compras se muestran en una tabla con las siguientes columnas: fecha, producto y cantidad. 2- El administrador ingresa al apartado de "Compras" y puede ver todas las compras realizadas por todos los usuarios del sistema, incluyendo el nombre de quien las realizó. Las compras se presentan en una tabla similar a la del usuario del staff, con las mismas columnas mencionadas anteriormente. 3- Tanto el usuario del staff como el administrador tienen la capacidad de ordenar la tabla por cualquier columna. Al hacer clic en el encabezado de una columna, las compras se reorganizan en función de los valores de esa columna en orden ascendente o descendente.

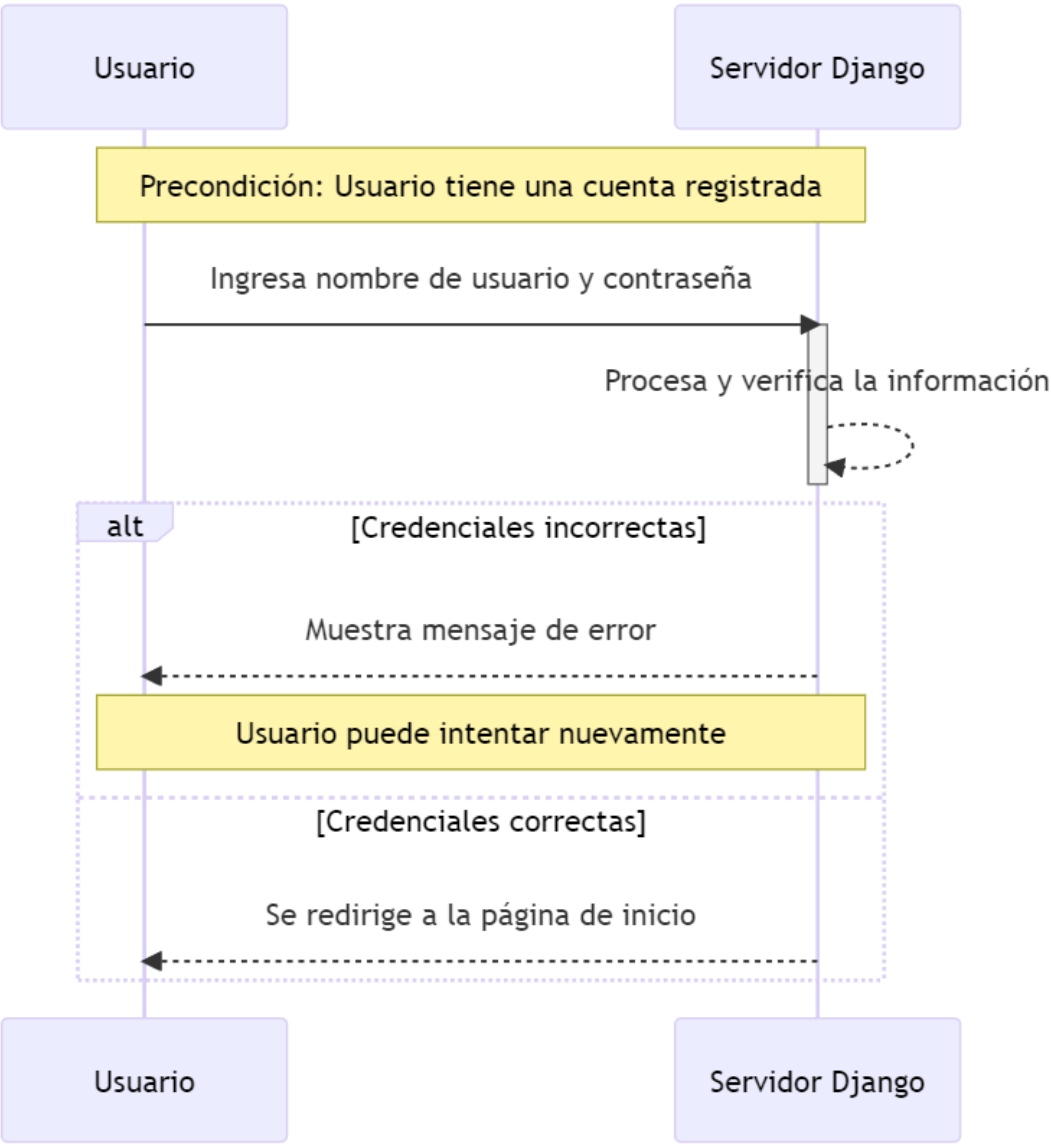
	CU8 – Búsqueda de un producto	
Descripción	Cualquier usuario con sesión iniciada podrá buscar el producto que busca.	
Actores	Usuario.	
Precondiciones	Haber iniciado sesión.	
Pasos del caso de uso	1-	El usuario ingresa en la barra de búsqueda el término o nombre del producto que desea buscar.
	2-	El sistema realiza una búsqueda en la base de datos en busca de todas las coincidencias de productos existentes con el término ingresado. Estas coincidencias pueden incluir productos que actualmente no tengan stock disponible.
	3-	El sistema muestra los resultados de la búsqueda, presentando al usuario las coincidencias encontradas. Para cada resultado, se el nombre del producto.
	4-	El usuario puede seleccionar una de las coincidencias de productos mostradas haciendo clic en ella.
	5-	Al seleccionar un producto de la lista de resultados, el sistema realizará una búsqueda con ese nombre específico, o simplemente si le da a buscar realiza una búsqueda con el patrón introducido en la barra de búsqueda.

Diagramas de secuencia

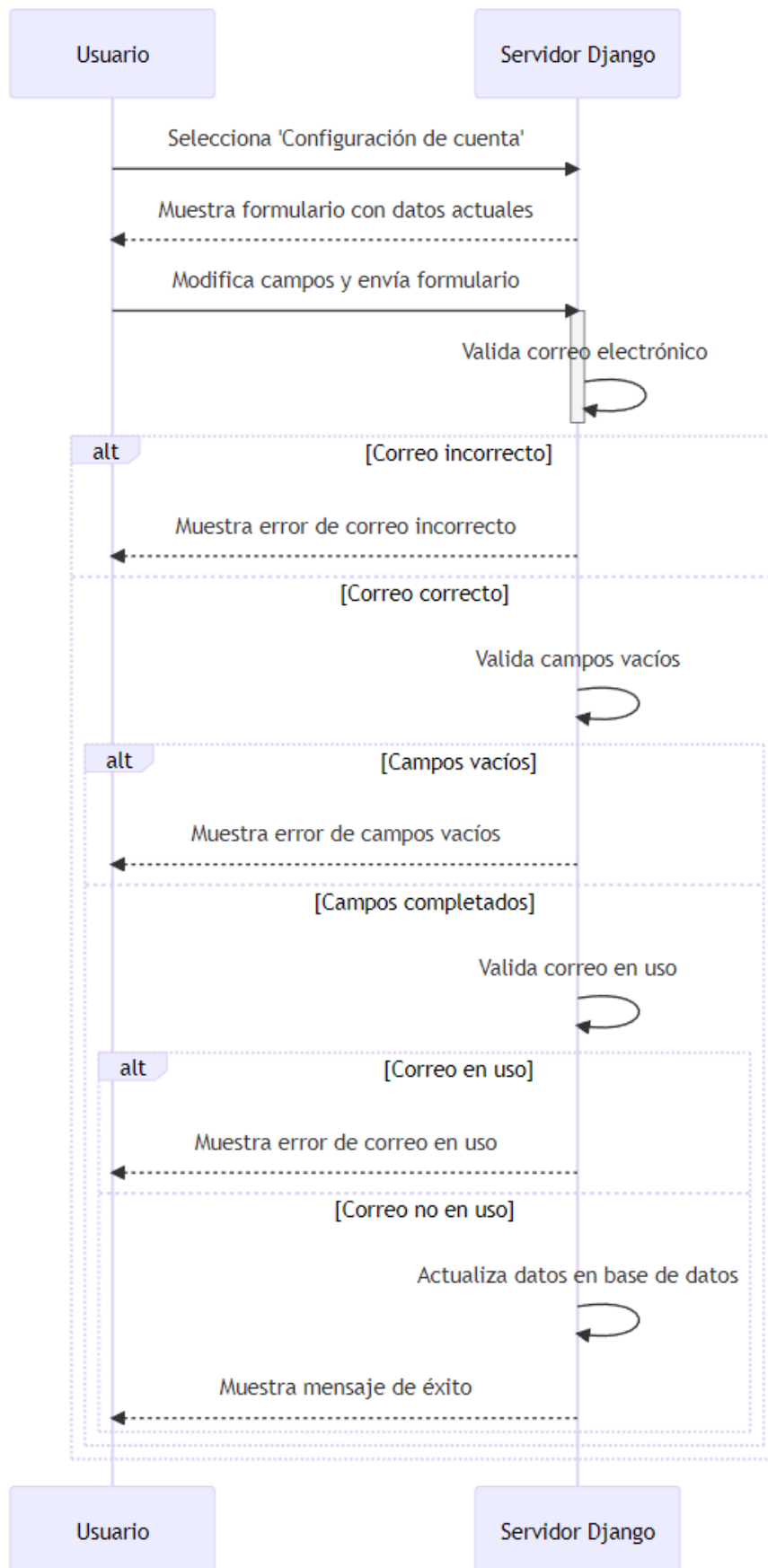
CU1- Crear cuenta



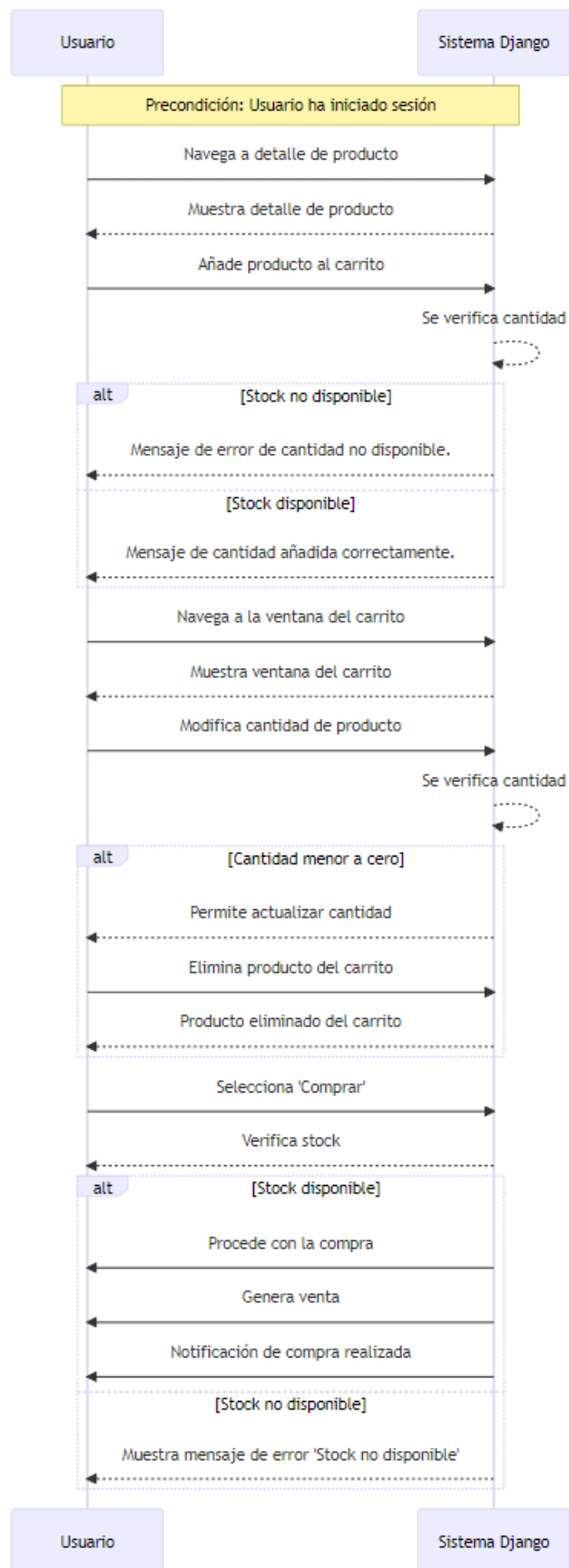
CU2- Inicio de sesión



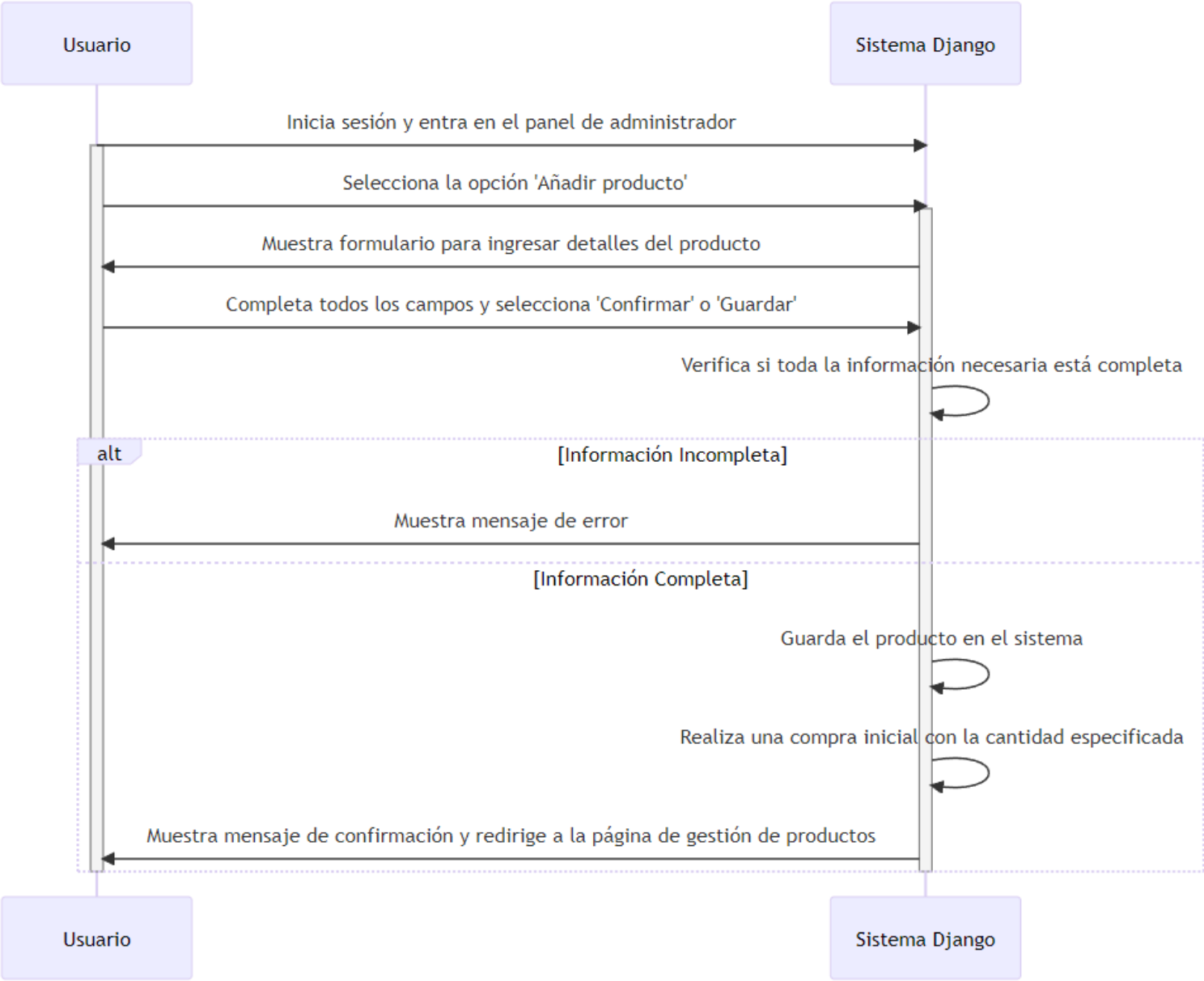
CU3- Modificar información de la cuenta



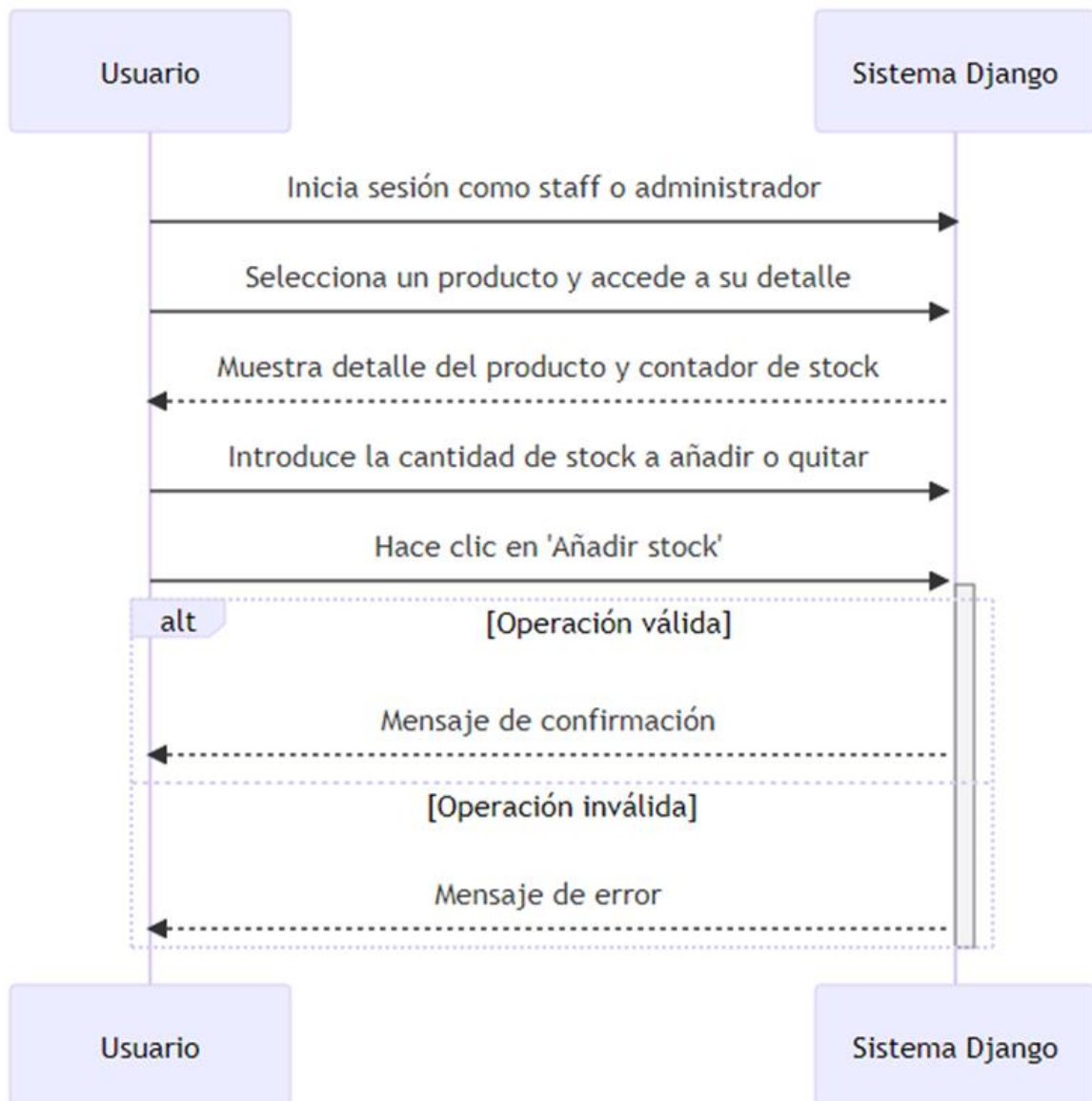
CU4 - Añadir/Modificar/ Eliminar del carrito



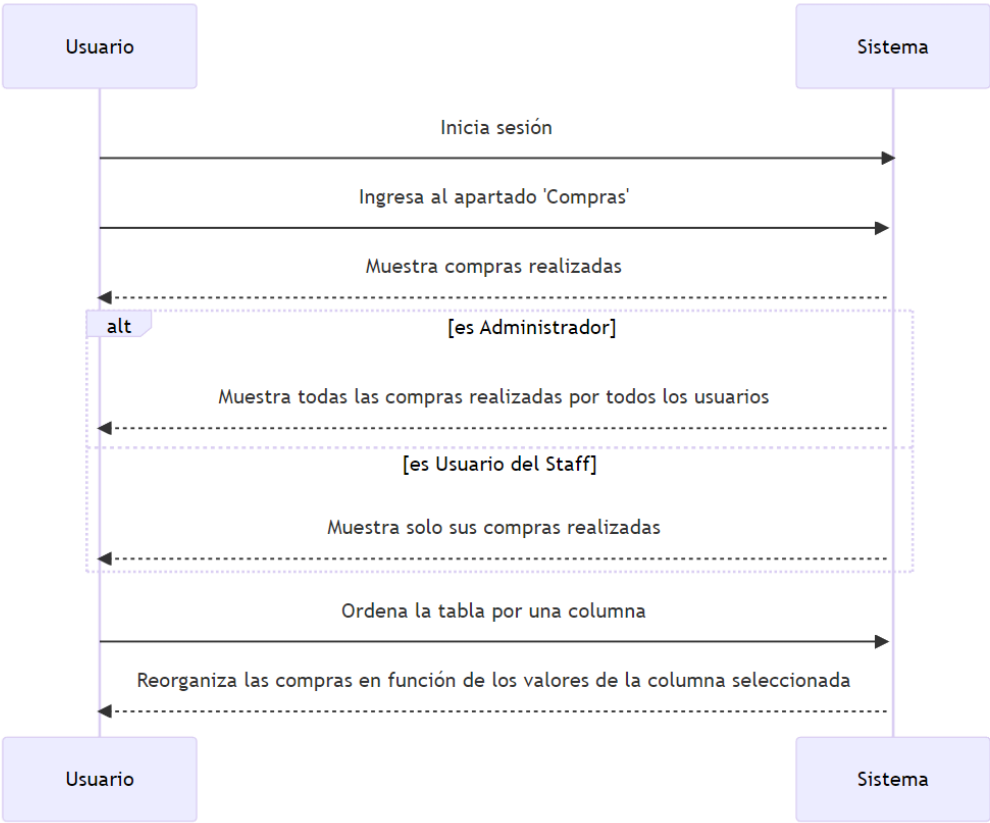
CU5 – Añadir producto



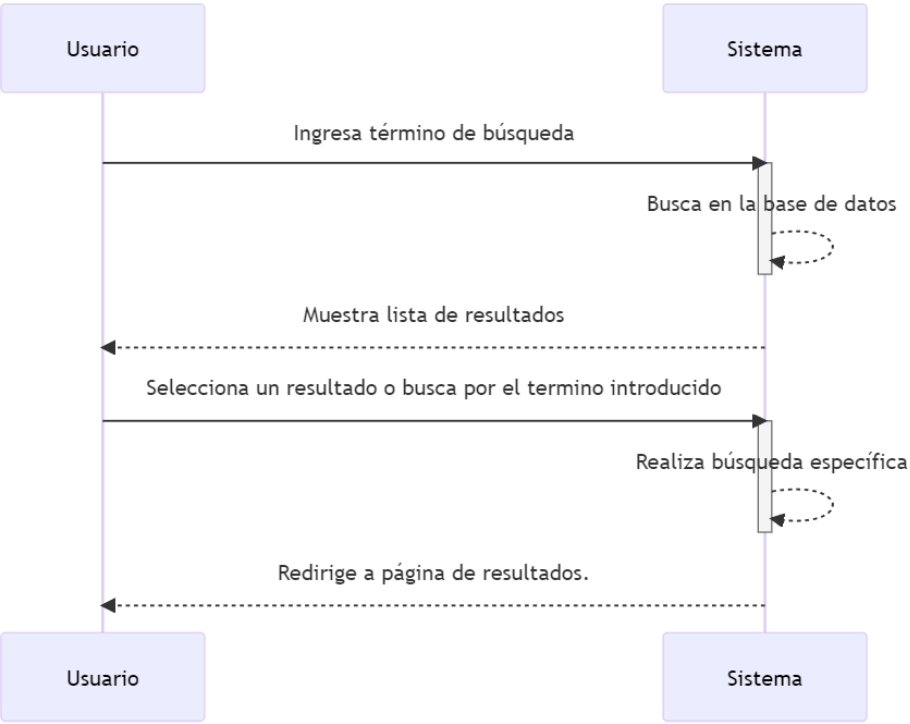
CU6 – Añadir/Quitar stock



CU7 – Visualización de compras de stock

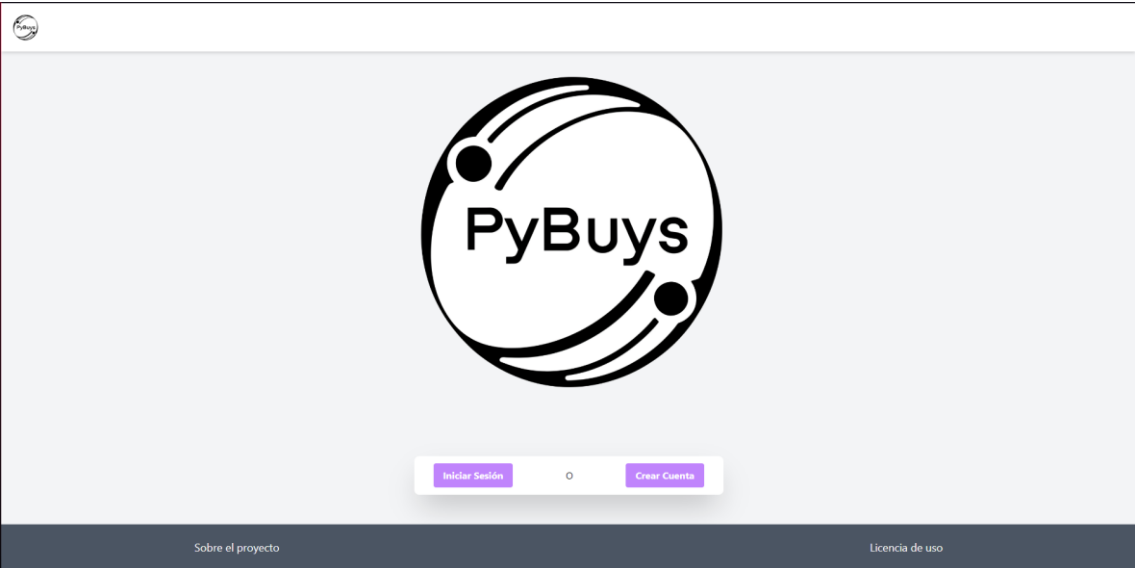


CU8 – Búsqueda de un producto

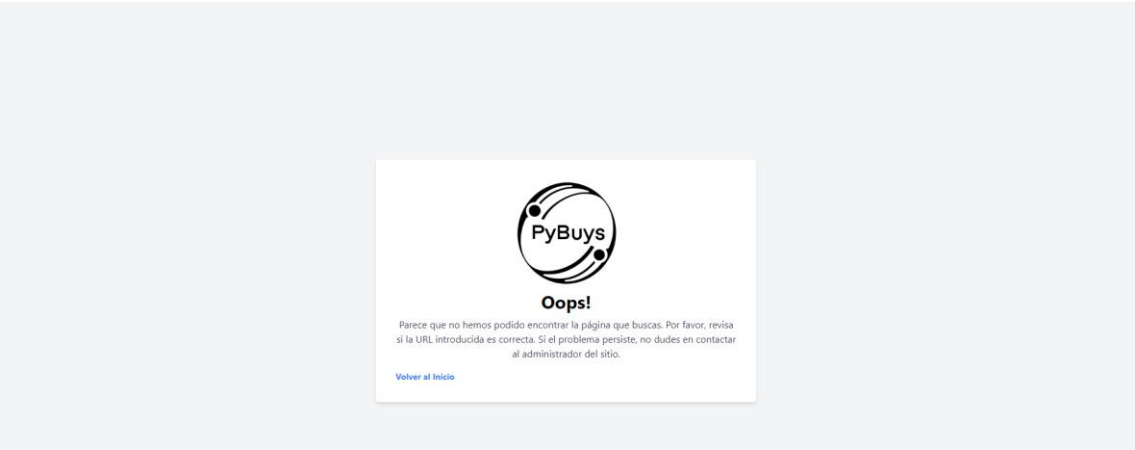


3.7 Diseño de la interfaz

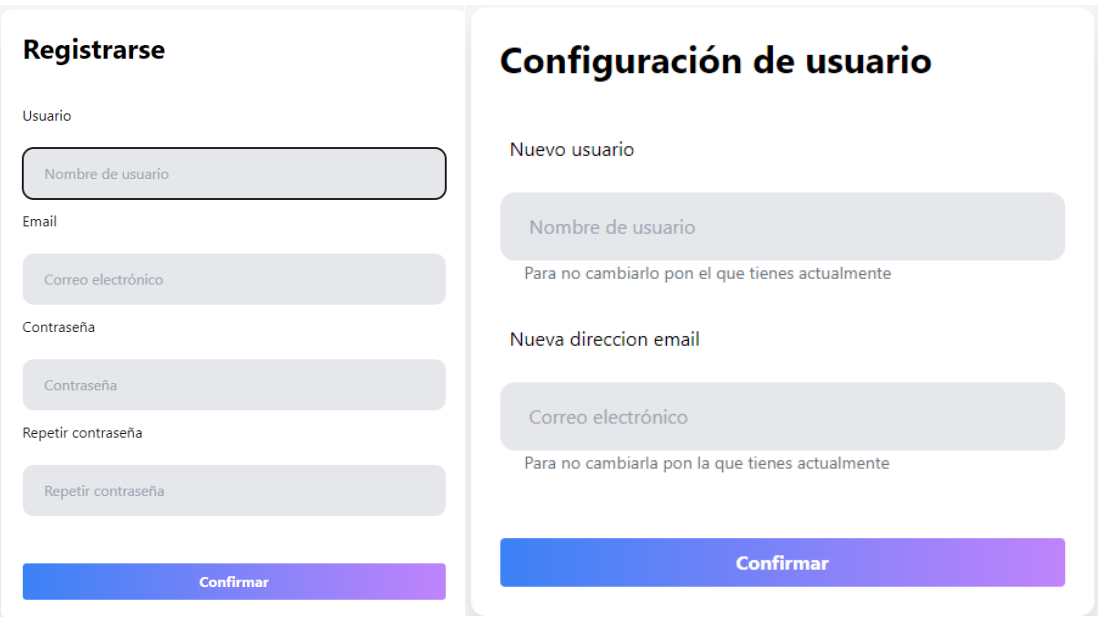
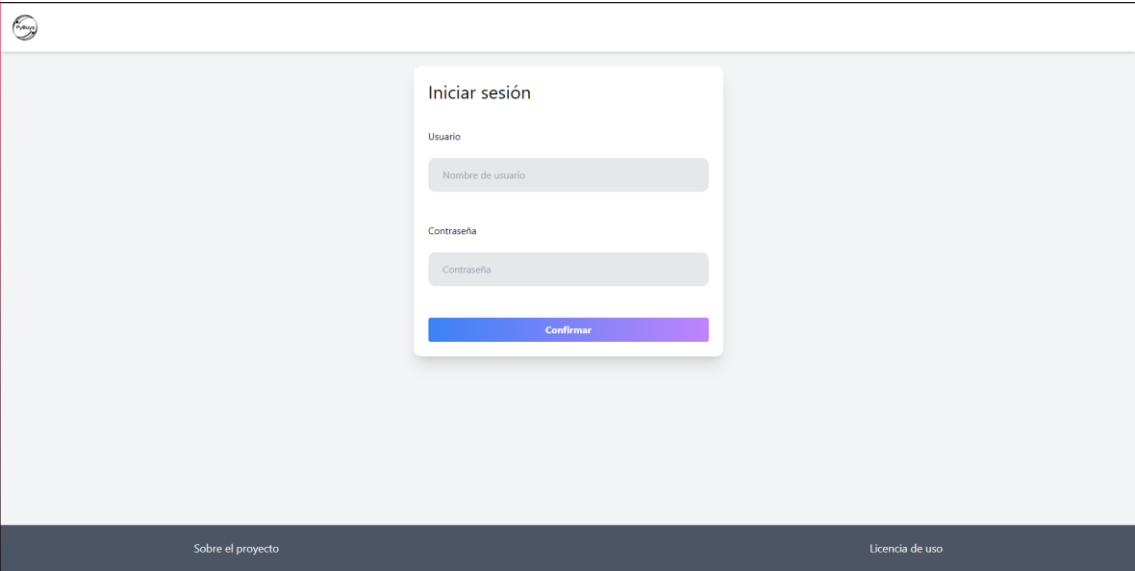
Pantalla de inicio



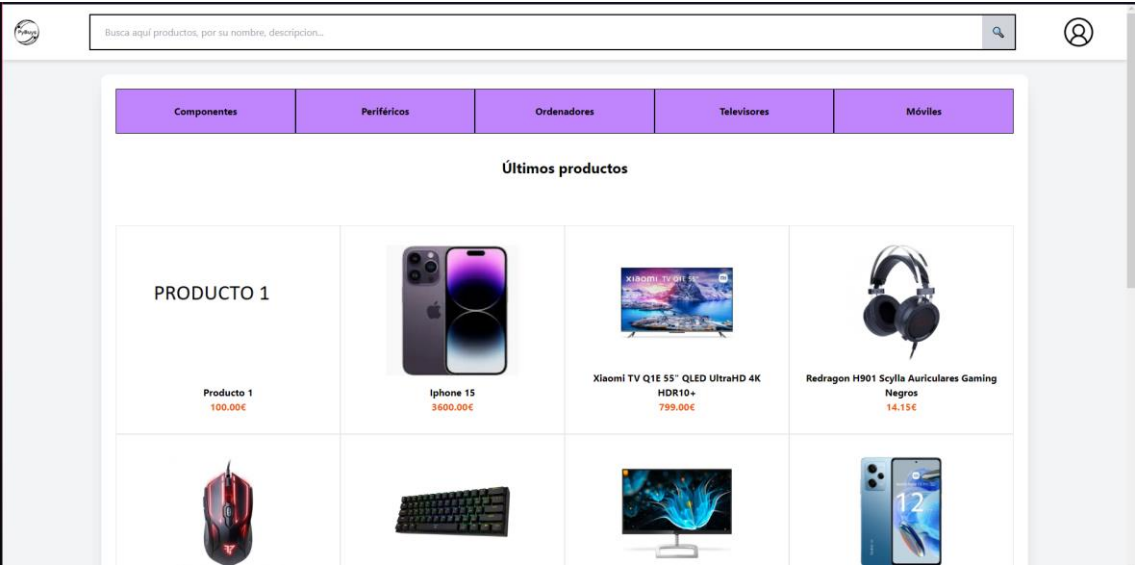
Pantalla de error 404:



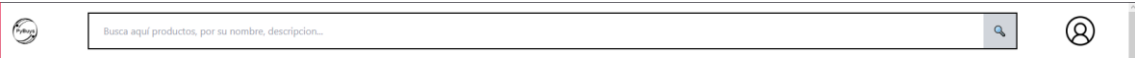
Formularios:



Pantalla principal:



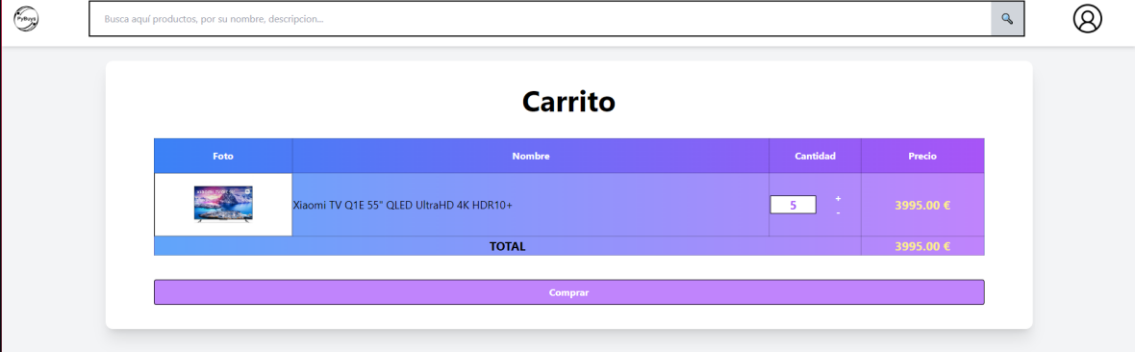
Header:



Footer:



Carrito:



Pantalla de compras:

Busca aquí productos, por su nombre, descripción...

Todas las compras

Filtrar por usuario...

Usuario	Producto	Cantidad	Fecha de compra
adminstrador	Iphone 15	200	2023-05-07 19:12:51
adminstrador	Apple Macbook Pro Apple M2/8GB/256GB SSD/GPU Deca	12	2023-05-08 16:02:58
adminstrador	Kingston A400 SSD 480GB	12	2023-05-08 16:03:12
adminstrador	Iphone 15	25	2023-05-10 17:44:22
adminstrador	Xiaomi TV Q1E 55" QLED UltraHD 4K HDR10+	45	2023-05-10 17:44:22
adminstrador	MSI GeForce RTX 4070 VENTUS 3X OC 12GB GDDR6X DLSS	2	2023-05-10 17:44:22
adminstrador	Kingston A400 SSD 480GB	48	2023-05-10 17:44:22
adminstrador	Sapphire Pulse AMD Radeon RX 6700 XT 12GB GDDR6	23	2023-05-10 17:44:22
adminstrador	Tempest MS200 Ninja Ratón Gaming 2.400 DPI Negro	6	2023-05-10 17:44:22
adminstrador	Redragon H901 Scylla Auriculares Gaming Negros	9	2023-05-10 17:44:22
adminstrador	Xiaomi Redmi Note 12 Pro 5G 8/256GB Azul Libre	15	2023-05-10 17:44:22

Detalle producto:

Busca aquí productos, por su nombre, descripción...


Componentes

Periféricos

Ordenadores

Televisores

Móviles



Redragon H901 Scylla Auriculares Gaming Negros

Disponibles: 17 unidades

23.99€

OFERTA

41.0%

TOTAL

0.00€

0

Añadir al carrito

Panel de administrador

0

Añadir stock

¡Cuidado, cada vez que añadas se generará una compra a tu nombre!

Para corregir una compra por favor añada la cantidad en negativo

Descripción:

Búsqueda:

a

Alurin Go Intel Pentium N4200/8GB/256GB SSD/nOS/14

AMD Ryzen 5 5600X 3.7GHz

Apple iPhone 14 Pro Max 128GB

Apple MacBook Pro Apple M2/8GB/256GB SSD/GPU Deca

Corsair Vengeance LPX DDR4 3200 PC4-25600 16GB 2X8

Kingston A400 SSD 480GB

Lenovo IdeaPad 3 15ITL6 Intel Core i5-1155G7/8GB/5

MSI MAG CoreLiquid C240 ARGB Kit Refrigeración LÍq

MSI MPG B550 GAMING PLUS

Philips 226E9QHAB 21.5" LED IPS FullHD FreeSync

Periféricos

Ordenadores

Televisores

Móviles

Redragon H901 Scylla Auriculares Gaming Negros

Disponibles: 17 unidades

23,99€

14.15€/ud

OFERTA

41,0%

TOTAL

0.00€

0

Añadir al carrito

+

-

Opciones:

Panel de administración

Compras

Carrito

Configuración de cuenta

Cerrar sesión

Panel de administración:

Administración de Django

BIENVENIDOS, **STAFF4** / VER EL SITIO / CAMBIAR CONTRASEÑA / CERRAR SESIÓN

Sitio administrativo

BUYSSALES

Compras

Productos en carrito

Ventas

PRODUCTOS

Categorías

Productos

Acciones recientes

Mis acciones

Ninguno disponible

Tabla de productos:

1	2	3	4	5	6
NOMBRE	PRECIO	REBAJA	PRECIO REAL	CANTIDAD	CATEGORIA
Alurin Go Intel Pentium N4200/8GB/256GB SSD/nOS/14	233,39	29,0	165.71€	8	Chromebook
AMD Ryzen 5 5600X 3.7GHz	185,9	50,0	92.95€	20	AMD
Apple iPhone 14 Pro Max 128GB	1339,0	8,0	1231.88€	0	Iphone
Apple Macbook Pro Apple M2/8GB/256GB SSD/GPU Deca	1409,01	12,0	1239.93€	20	Macbook
Corsair Vengeance LPX DDR4 3200 PC4-25600 16GB 2X8	39,99	9,0	36.39€	0	Memoria RAM
HP Victus 16-d1038ns Intel Core i7-12700H/16GB/512	879,0	23,0	676.83€	349	Portátiles gaming
Intel Core i5-12400F 2.5 GHz	173,04	0,0	173.04€	66	Intel
Iphone 15	4000,0	10,0	3600.00€	0	Iphone
Kingston A400 SSD 480GB	27,0	0,0	27.00€	249	Discos duros SSD
Lenovo IdeaPad 3 15ITL6 Intel Core i5-1155G7/8GB/5	429,99	26,0	318.19€	0	Portátiles
MSI GeForce RTX 4070 VENTUS 3X OC 12GB GDDR6X DLSS	669,9	6,0	629.71€	45	Nvidia
MSI MAG CoreLiquid C240 ARGB Kit Refrigeración Liq	139,98	0,0	139.98€	15	Ventilación
MSI MPG B550 GAMING PLUS	158,99	0,0	158.99€	12	Placas Base
Philips 226E9QHAB 21.5" LED IPS FullHD FreeSync	149,0	13,0	129.63€	0	Monitores
Producto 1	100,0	0,0	100.00€	65	Componentes
Redragon Dragonborn K630 Teclado Mecánico Gaming R	42,99	4,0	41.27€	0	Teclados
Redragon H901 Scylla Auriculares Gaming Negros	23,99	41,0	14.15€	17	Auriculares
Samsung Galaxy M13 4/128GB Verde Libre	193,99	18,0	159.07€	0	Samsung

3.8 Código fuente

Al estar diseñada la aplicación con el framework django hace falta conocer su estructura básica para poder desarrollar y entender el funcionamiento de Pybuys.

Django al crear un proyecto crea la carpeta principal donde están toda la configuración principal de la aplicación (bdd a la que conectar, urls, host en el que se aloja...). Y un manage.py que nos permitirá ejecutar el proyecto, ejecutar los test, migrar cambios a bdd entre otras cosas.

Cuando queremos implementar alguna página lo tenemos que hacer por apps, y es así como django organiza la aplicación en general.

Cada app tiene principalmente:

- Views.py: Código que se ejecuta y envía al usuario el html correspondiente
- Carpeta de templates con todos los html correspondientes.
- Models.py: Donde van los modelos que migrarán a bdd.
- Urls.py: Donde se indican el formato de las urls de la app.
- Admin.py: Si quieres modificar algo en el panel de administración.

Las urls se acceden a ellas poniendo por ejemplo si queremos acceder a detalle URL_BASE/NOMBRE_APP/detalle.

En el proyecto de Pybuys se han usado las apps, product(donde esta todo lo relacionado a productos y categorías), core(donde esta todo sobre las paginas de inicio y formularios) y buysSales(donde se encuentra todo lo relacionado a las compras, ventas y al carrito)

Partes para destacar del código:

1. La creación de templates tags. Django nos permite una sintaxis para reutilizar código y que sea más fácil su generación, algo parecido a lo que nos permite php. En este caso he creado un tag de lista de productos que nos permite siempre que invoquemos en el html ese tag y le pasemos como parámetro una lista de productos nos muestra los productos de manera correcta.

Creación del tag en Python:

```
@register.simple_tag
def show_product_list(productos):
    return {"productos": productos}

product_template = get_template("product/product_list.html")
register.inclusion_tag(product_template)(show_product_list)
```

Html:

```
{% load custom_tags %}
<div class="grid grid-cols-4 grid-flow-row mt-20 w-full">
    {% for producto in productos %}
        <a class="border text-center flex flex-col items-center py-8" href="{% url 'detail' producto.pk %}">
            
            <div class="flex flex-col justify-between w-full px-[5%] mt-4">
                <strong><p>{{ producto.nombre }}</p></strong>
                <strong><p class="text-orange-600">{{ producto.get_precio_real}}</p></strong>
            </div>
        </a>
    {% endfor %}
</div>
```

2. Funciones útiles aplicadas en javascript. En el apartado del front end, se han usado funciones y clases, bastante útiles y reutilizables para poder facilitar el desarrollo, entre ellas destacan:
 - a. Función de mostrar notificación al usuario:

```
function mostrarNotificacion(message) {
    // Verificar si el navegador soporta la API de Notificaciones
    if ("Notification" in window) {
        // Pedir permiso al usuario para mostrar notificaciones
        Notification.requestPermission().then(function (permission) {
            if (permission === "granted") {
                // Crear una nueva notificación
                var notification = new Notification(message);
                // Mostrar la notificación
                setTimeout(notification.close.bind(notification), 5000);
            }
        });
    }
}
```

Que nos permite mostrar una notificación al usuario, esto está por defecto puesto en el html base para que podamos simplemente desde cada view correspondiente enviar un mensaje sin tener que implementar en cada página que se muestre ese mensaje:

```
{% if messages %}
  {% for message in messages %}
    mostrarNotificacion('{{ message }}');
  {% endfor %}
{% endif %}
```

- b. Clase de contador, la cual sirve para poder controlar fácilmente cada vez que hay un contador (por ejemplo, cambiar el stock o cambiar cantidad en carrito).

```
class Contador {
  constructor(id_elemento, inicial, maximo, minimo = 0) {
    this.id_elemento = id_elemento;
    this.valor = inicial;
    this.maximo = maximo;
    this.minimo = minimo;
  }

  modificarCantidad(cantidadASumar) {
    cantidadASumar = parseInt(cantidadASumar);
    const elemento = document.getElementById(this.id_elemento);
    let valorActual = parseInt(elemento.textContent || elemento.value);
    let puedeSumar = (valorActual + cantidadASumar >= this.minimo && cantidadASumar < 0) || (valorActual + cantidadASumar <= this.maximo && cantidadASumar > 0);
    if (puedeSumar) {
      this.valor += cantidadASumar;
      if (elemento.tagName === 'INPUT') {
        elemento.value = this.valor;
      } else {
        elemento.textContent = this.valor;
      }
    }
  }
}
```

Simplemente pasándole la cantidad inicial, el máximo, el mínimo y el id del elemento nos permite que podamos subir o bajar cómodamente sin preocuparse mucho de que no se pasará del límite.

- c. Lista hovered, clase que nos ayuda a crear hovers complicados a los elementos de una lista.

Simplemente indicamos, que elemento es la lista, que tipo de elemento queremos que les afecte el hovered(especificación), que atributos tienen de normal y que atributos tendrán durante el hover.

```
class ListaHovered {
  constructor(id_elemento, destacado, normal, especificacion = "") {
    // destacado se agrega al elemento cuando el mouse entra
    // normal se agrega al elemento cuando el mouse sale
    // id_elemento es el id del elemento que contiene los elementos a los que se les agregará el hover
    document.addEventListener('DOMContentLoaded', function () {
      const lista = document.getElementById(id_elemento);
      const elementos = lista.querySelectorAll(especificacion);
      elementos.forEach(elemento => {
        elemento.addEventListener('mouseenter', function () {
          destacado.forEach(clase => {
            this.classList.add(clase);
          });
          normal.forEach(clase => {
            this.classList.remove(clase);
          });
        });

        elemento.addEventListener('mouseleave', function () {
          destacado.forEach(clase => {
            this.classList.remove(clase);
          });
          normal.forEach(clase => {
            this.classList.add(clase);
          });
        });
      });
    });
  }
}
```

Ejemplo:

Componentes	Periféricos	Ordenadores	Televisores	Móviles
-------------	-------------	-------------	-------------	---------

3. En Python, donde ha ido toda la lógica del servidor cabe comentar la creación de los modelos, la lógica de negocio la cual va en funciones correspondientes. Y lo mas destacable de todo eso:
- El añadir stock, el cual hace todas las comprobaciones correspondientes para que no suceda ningún error a la hora de modificar stock a un producto, incluso pasa por todos los carritos si quita stock:

```
@login_required
def add_stock(request, id_producto):
    producto = get_object_or_404(Productos, pk=id_producto)
    if request.method == "POST":
        cantidad_stock = int(request.POST.get('cantidad_stock'))

        # Asegura de que el usuario es un administrador antes de permitirle añadir stock
        if not request.user.is_staff:
            return HttpResponseForbidden()

        # Añade la cantidad de stock al producto
        if producto.cantidad + cantidad_stock < 0:
            messages.error(request, "La cantidad de stock debe ser mayor que 0.")
            return redirect('detail', pk=producto.id)
        producto.cantidad += cantidad_stock
        producto.save()

        # Registra la compra
        compra = Compras(id_usuario=request.user, id_producto=producto, cantidad=cantidad_stock)
        compra.save()

        # Comprueba si la cantidad de producto en el carrito es mayor que la cantidad de producto en stock
        # Si es así, reduce la cantidad en el carrito al nivel de stock
        carritos = ProductosEnCarrito.objects.filter(producto=producto)
        for carrito in carritos:
            if carrito.cantidad > producto.cantidad:
                carrito.cantidad = producto.cantidad
                if carrito.cantidad == 0:
                    carrito.delete()
            else:
                carrito.save()

        # Comprueba si el producto se ha agotado y, si es así, elimina el producto de todos los carritos
        if producto.cantidad <= 0:
            eliminar_de_carritos(producto.pk)

        return redirect('detail', pk=producto.id)

    return render(request, "product/detail.html", {"producto": producto})
```

Lo mismo sucede con el modificar carrito:

```
@login_required
def add_to_cart(request, id_producto):
    if request.method == 'POST':
        cantidad = request.POST.get('cantidad')
        try:
            cantidad = int(cantidad)
        except (ValueError, TypeError):
            return JsonResponse({'error': "La cantidad proporcionada no es válida."}, status=400)

        producto = get_object_or_404(Productos, id=id_producto)

        if producto.cantidad < cantidad:
            return JsonResponse({'error': f"No hay suficiente cantidad de {producto.nombre} en el inventario."}, status=400)

        carrito, created = ProductosEnCarrito.objects.get_or_create(id_usuario=request.user, producto=producto, defaults={'cantidad': cantidad})

        if not created:
            if producto.cantidad < (carrito.cantidad + cantidad):
                return JsonResponse({'error': f"No hay suficiente cantidad de {producto.nombre} en el inventario para añadir más al carrito."}, status=400)

            carrito.cantidad += cantidad
            # Si la cantidad es 0, borra el producto del carrito.
            if carrito.cantidad == 0:
                carrito.delete()
            return JsonResponse({'success': "Producto eliminado del carrito."})
        else:
            carrito.save()

    return JsonResponse({'success': "Producto añadido al carrito con éxito."})
```

b. La clase de modelo del producto:

Contiene todas las funciones disponibles para obtener el precio de manera adecuada y para formatear correctamente la descripción para hacerla más bonita. También es a destacar que al eliminar el producto también se elimina la imagen.

```
class Productos(models.Model):
    class Meta:
        verbose_name = "Producto"
        verbose_name_plural = "Productos"

    nombre = models.CharField(max_length=50, unique=True)
    precio = models.FloatField(validators=[MinValueValidator(0)])
    image = models.ImageField(upload_to="products")
    rebaja = models.FloatField(default=0, null=True, validators=[MinValueValidator(0)])
    cantidad = models.IntegerField(validators=[MinValueValidator(0)])
    categoria = models.ForeignKey(Categorias, on_delete=models.CASCADE)
    descripcion = models.TextField(blank=True, null=True)
    creado = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.nombre

    def delete(self, *args, **kwargs):
        if self.image:
            os.remove(os.path.join(settings.MEDIA_ROOT, self.image.name))
        super().delete(*args, **kwargs)

    def get_precio(self):
        precio_con_descuento = self.precio
        if self.rebaja >= 0:
            precio_con_descuento = self.precio - (self.precio * (self.rebaja / 100))
        return max(round(precio_con_descuento, 2), 0)

    def get_precio_real(self):
        return "{:.2f}€".format(self.get_precio())

    def get_descripcion_formateada(self):
        # Sustituir '*' por un punto de bala y agregar saltos de línea después de cada punto
        texto = (
            self.descripcion.replace("* ", "• ")
            .replace(".", "<br>")
            .replace("\n", "<br>")
        )
        # Identificar todo lo que esté dentro de 3 comillas y ponerlo en negrita
        texto = re.sub(r"\'{3}(.*?)\'{3}", r"<br><strong>\1</strong><br>", texto)
        return f"{texto}"
```

La descripción cambia todas las “A” por **A** y cambia por puntos de bala todas las “* ”.

- c. Otra cosa para destacar son los ficheros de admin.py, en los cuales he modificado para que se vea mejor y nos muestre información más importante:

```
class ComprasAdmin(admin.ModelAdmin):
    list_display=('id_usuario', 'id_producto', 'cantidad', 'creado')
    ordering=('id_usuario', 'id_producto', 'cantidad', 'creado')
    search_fields=('id_usuario', 'id_producto', 'cantidad', 'creado')
    list_filter=('id_usuario', 'id_producto', 'cantidad', 'creado')
    date_hierarchy='creado'
    fieldsets=(
        ('Información básica', {
            'fields':('id_usuario', 'id_producto', 'cantidad')
        }),
        ('Fecha', {
            'fields':('creado',)
        }),
    )
    readonly_fields=('creado', 'id_usuario')

    def save_model(self, request, obj, form, change):
        obj.id_usuario = request.user
        super().save_model(request, obj, form, change)

admin.site.register(Compras, ComprasAdmin)
```

```
class ProductosEnCarritoAdmin(admin.ModelAdmin):
    list_display=('id_usuario', 'producto', 'cantidad')
    ordering=('id_usuario', 'producto', 'cantidad')
    search_fields=('id_usuario', 'producto', 'cantidad')
    list_filter=('id_usuario', 'producto', 'cantidad')
    fieldsets=(
        ('Información básica', {
            'fields':('id_usuario', 'producto', 'cantidad')
        }),
    )

admin.site.register(ProductosEnCarrito, ProductosEnCarritoAdmin)
```


Y gracias a esto nos muestra, todos estos filtros y maneras de ordenar:

Seleccione Compra para ver

2023 7 de mayo 8 de mayo 10 de mayo 11 de mayo 12 de mayo 14 de mayo 17 de mayo 18 de mayo 21 de mayo 22 de mayo 23 de mayo				
ID USUARIO	ID PRODUCTO	CANTIDAD	CREADO	
adminstrador	Kingston A400 SSD 480GB	12	8 de mayo de 2023 a las 16:03	
adminstrador	Kingston A400 SSD 480GB	48	10 de mayo de 2023 a las 17:44	
adminstrador	Sapphire Pulse AMD Radeon RX 6700 XT 12GB GDDR6	23	10 de mayo de 2023 a las 17:44	
adminstrador	MSI GeForce RTX 4070 VENTUS 3X OC 12GB GDDR6X DLSS	2	10 de mayo de 2023 a las 17:44	
adminstrador	MSI MAG CoreLiquid C240 ARGB Kit Refrigeración Liq	10	10 de mayo de 2023 a las 17:44	
adminstrador	Apple Macbook Pro Apple M2/8GB/256GB SSD/GPU Deca	12	8 de mayo de 2023 a las 16:02	
adminstrador	Xiaomi Redmi Note 12 Pro 5G 8/256GB Azul Libre	15	10 de mayo de 2023 a las 17:44	
adminstrador	Philips 22E9QHAB 21.5" LED IPS FullHD FreeSync	1	10 de mayo de 2023 a las 17:50	
adminstrador	Philips 22E9QHAB 21.5" LED IPS FullHD FreeSync	1	10 de mayo de 2023 a las 17:51	
adminstrador	Tempest MS200 Ninja Ratón Gaming 2.400 DPI Negro	6	10 de mayo de 2023 a las 17:44	
adminstrador	Tempest MS200 Ninja Ratón Gaming 2.400 DPI Negro	6	10 de mayo de 2023 a las 17:45	
adminstrador	Redragon H901 Scylla Auriculares Gaming Negros	-4	10 de mayo de 2023 a las 21:57	
adminstrador	Redragon H901 Scylla Auriculares Gaming Negros	-3	10 de mayo de 2023 a las 22:03	
adminstrador	Redragon H901 Scylla Auriculares Gaming Negros	-2	10 de mayo de 2023 a las 22:02	
adminstrador	Redragon H901 Scylla Auriculares Gaming Negros	-2	11 de mayo de 2023 a las 13:23	
adminstrador	Redragon H901 Scylla Auriculares Gaming Negros	-2	11 de mayo de 2023 a las 13:23	
adminstrador	Redragon H901 Scylla Auriculares Gaming Negros	2	10 de mayo de 2023 a las 22:02	
adminstrador	Redragon H901 Scylla Auriculares Gaming Negros	2	11 de mayo de 2023 a las 13:23	
adminstrador	Redragon H901 Scylla Auriculares Gaming Negros	3	10 de mayo de 2023 a las 22:02	
adminstrador	Redragon H901 Scylla Auriculares Gaming Negros	4	10 de mayo de 2023 a las 21:56	

FILTRO

↓ Por id usuario

Todo

admin3

adminstrador

goku

staff1

staff2

staff4

Usuario_staff

UsuarioNuevo123

↓ Por id producto

Todo

Alurin Go Intel Pentium

N4200/8GB/256GB SSD/nOS/14

AMD Ryzen 5 5600X 3.7GHz

Apple iPhone 14 Pro Max 128GB

Apple Macbook Pro Apple

M2/8GB/256GB SSD/GPU Deca

Corsair Vengeance LPX DDR4 3200

PC4-25600 16GB 2X8

HP Victus 16-d1038ns Intel Core i7-

12700H/16GB/512

Intel Core i5-12400F 2.5 GHz

Iphone 15

Kingston A400 SSD 480GB

Lenovo IdeaPad 3 15ITL6 Intel Core

i5-1155G7/8GB/5

MSI GeForce RTX 4070 VENTUS 3X

OC 12GB GDDR6X DLSS

MSI MAG CoreLiquid C240 ARGB Kit

Refrigeración Liq

MSI MPG B550 GAMING PLUS

Philips 22E9QHAB 21.5" LED IPS

4. Fichero de configuración, settings.py, en el cual se establece la región, el host donde se aloja, lugar de guardado de las imágenes, la bdd a la que se tiene que conectar:

```
ALLOWED_HOSTS = ["*"]
LOGIN_URL = "/login/"
LOGIN_REDIRECT_URL = "/index"
LOGOUT_REDIRECT_URL = ""
```

```
LANGUAGE_CODE = "es"

TIME_ZONE = 'Europe/Madrid'

USE_I18N = True

USE_TZ = False
```

```
# Media files
MEDIA_URL = "/media/"
MEDIA_ROOT = BASE_DIR / "media"
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'pybuys',
        'USER': 'root',
        'PASSWORD': '1234',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

También cabe a destacar la instalación, que es algo que se desarrollará en el Manual de Instalación.

4 Fase de pruebas

Unitarias

Durante esta fase se han creado 37 pruebas unitarias distribuidas en los test.py.

En donde se han probado:

- El test.py en la app de buysSales, se ha probado si se crean correctamente las compras, ventas y los productos en carrito. Y prueba si al crear una compra se guarda quien lo ha creado.
- El test.py en la app de product, se prueba que vayan correctamente todos los métodos que contiene el producto, el funcionamiento correcto de todos los customtags y por último se prueba que al guardar un producto se guarde una compra con esa cantidad.

Integradas

Se han realizado 35 pruebas, aproximadamente 4 por requisito, las cuales se encuentran en el informe de pruebas ([I](#)), prueban todas las funcionalidades anteriormente descritas.

De Pico, estabilidad y estrés

Se han realizado unas 14 en total también se encuentran en el informe de pruebas ([I](#)), estas prueban cómo se comporta el sistema ante grandes cantidades de solicitudes variando el periodo y la cantidad según el tipo de prueba.

Pruebas fallidas

Cabe a destacar que todas las pruebas han sido correctas menos una. Esta se percibe como una futura implementación, tiene que ver con la configuración del usuario y el cambio de contraseña.

5 Conclusiones finales

Modificaciones a futuro

- **Migración de base de datos:** El sistema se migrará a otra base de datos si es necesario.
- **Cambio de contraseña:** Se implementará la funcionalidad para que los usuarios puedan cambiar su contraseña desde la configuración de su cuenta.
- **Gestión de grupos y permisos:** Se permitirá la modificación de grupos y tipos de usuario con el fin de brindar diferentes niveles de acceso al panel de administración y definir permisos específicos para cada grupo de usuarios.
- **Personalización del logo:** El sistema se adaptará para permitir la configuración del logo de acuerdo con los requerimientos del cliente.
- **Eliminación del límite de productos en la página principal:** Se eliminará la limitación de mostrar solo 16 productos en la página principal, permitiendo mostrar una cantidad mayor según las necesidades del cliente.

Ampliaciones a futuro

- **Escalabilidad y rendimiento:** Se realizará una adaptación del sistema para su uso en empresas de mayor envergadura, utilizando tecnologías como Nginx para gestionar las solicitudes de manera más eficiente y garantizar un rendimiento óptimo.
- **Sistema de pago:** Se implementará un sistema de pago utilizando las bibliotecas proporcionadas por Django, como Django-payments, para facilitar las transacciones y mejorar la experiencia de compra de los usuarios.
- **Mejora de la estética:** Se realizará una mejora en el aspecto visual del sistema, implementando cambios de diseño o temas para proporcionar una experiencia más atractiva y agradable para los usuarios.
- **Ampliación a otros tipos de negocios:** Se ampliará el enfoque del sistema para adaptarse a diferentes tipos de negocios de comercio electrónico, no limitándose únicamente a productos tecnológicos. Se implementarán etiquetas y elementos más genéricos que se ajusten a diversos tipos de productos.
- **Interfaz de administración diferenciada:** Se establecerá una interfaz de administración específica para los usuarios con roles de administrador, brindando funcionalidades adicionales y una experiencia de gestión más completa y eficiente.

6 Documentación del sistema desarrollado

6.1 Manual de instalación

Se encuentra en el fichero de README.txt junto con el código del proyecto. (II)

6.2 Manual de uso

Explicado claramente en el video correspondiente. (III)

7 Bibliografía

- **Django Documentation:** <https://docs.djangoproject.com/en/4.2/> - Documentación oficial de Django que proporciona una guía completa para el desarrollo de proyectos utilizando el framework Django.
- **Ordenar una tabla en JavaScript:** <https://www.lawebdelprogramador.com/codigo/JavaScript/5203-Ordenar-una-tabla.html> - Recurso que explica cómo implementar la funcionalidad de ordenar una tabla utilizando JavaScript.
- **Vistas genéricas de Django en Mozilla Developer Network:** https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Generic_views - Artículo de Mozilla Developer Network que explica cómo utilizar las vistas genéricas de Django de manera adecuada, proporcionando ejemplos y casos de uso
- **Tango with Django:** <http://www.tangowithdjango.com/> - Libro en línea que aborda el desarrollo de aplicaciones web utilizando Django. Proporciona una guía práctica y ejemplos paso a paso.
- **Django for Beginners:** <https://djangoforbeginners.com/> - Libro en línea que brinda una introducción al desarrollo web con Django, cubriendo desde los conceptos básicos hasta proyectos más complejos. Incluye ejemplos prácticos y explicaciones detalladas.
- **Two Scoops of Django: Best Practices for Django:** <https://www.twoscoopspress.com/products/two-scoops-of-django-3-x> - Libro que ofrece consejos y buenas prácticas para el desarrollo de aplicaciones Django. Cubre diversos aspectos de Django, desde la estructura del proyecto hasta las pruebas y despliegue.
- **Django Girls Tutorial:** <https://tutorial.djangogirls.org/> - Tutorial interactivo para principiantes que introduce el desarrollo web con Django. Proporciona instrucciones paso a paso y ejercicios prácticos.

8 Anexos

1. Anexo I: Informe de pruebas (IP.xlsx)
2. Anexo II: Manual de instalación (README.txt)
3. Anexo III: Manual de uso (MU.mp4)