**PLAN DE FASE**

**Versión 3.0**

**Lima, junio del 2023.**

**Historial de revisiones**

| **Fecha** | **Revisión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
| 30/04/2023 | 1.0 | Primera versión del plan de fase | Consultora HomeSkill S.A. |
| 10/05/2023 | 2.0 | Segunda versión del plan de fase | Consultora HomeSkill S.A. |
| 21/06/2023 | 3.0 | Tercera versión del plan de fase | Consultora HomeSkill S.A. |

1. **Fase de inicio.**

* **Introducción.**
  + **Objetivos de la fase.**
    - Planificar y asignar las responsabilidades de cada uno de los miembros del equipo de trabajo.
    - Identificar los requerimientos del sistema a realizar.
    - Identificar los riesgos del proyecto y establecer estrategias sobre ellos.
    - Saber qué es lo que desea el cliente y establecer objetivos para el proyecto.
  + **Descripción de la fase .**

En esta fase se llevará a cabo la planificación y definición del proyecto. Se van a establecer los objetivos del proyecto, los entregables y los requisitos para asegurar el óptimo funcionamiento. Se establecerá un equipo de trabajo a los que se les asignan tareas y responsabilidades para cada miembro del equipo. Además se va a realizar una evaluación de riesgos y se establecerán estrategias para moderarlos.

* + **Entregables de la fase.**
    - Plan de iteración 1.
    - Repositorio del proyecto.
    - Especificar la arquitectura y diseño del software.
    - Proponer un diseño inicial para la Interfaz Web (UI).
    - Establecer estilos para la página web.
    - Especificar el diseño para la base de datos.

* **Planificación**
  + **Cronograma de fase**
    - Semana 1: Definir a qué sector está dirigido el proyecto y los establecimiento de objetivos.
    - Semana 2-3:Identificar los requerimientos y especificarlos.
    - Semana 4: Diseño de la arquitectura de alto nivel del sistema.
  + **Asignación de tareas y responsabilidades**
    - **Jefe de proyecto:**

Jorge Paul Ipanaque Pazo.

* + - **Arquitecto de software:**

Jorge Paul Ipanaque Pazo.

* + - **Desarrolladores Back - End:**

Jose Carlos Cjumo Chumbes.

Gianlucas Amed Hinostroza Quispe.

* + - **Desarrolladores Front - End:**

Daniel Ames Camayo.

Jesus Angel Saenz Chang.

* + - **Analistas de Base de datos:**

Piero Jaime Ramirez Alvarado.

Daniel Ames Camayo.

* + - **Analista QA:**

Jose Carlos Cjumo Chumbes.

* + - **Analista Junior:**

Carlos Alberto Castillo Bernal.

* + - **Testers:**

Carlos Alberto Castillo Bernal.

Jesus Angel Saenz Chang.

* **Requerimientos**
  + **Análisis de requerimientos**

Los siguientes requerimientos cumplen con las expectativas del cliente:

* + - **Requerimientos:**
* Los usuarios podrán registrar un nuevo usuario o ingresar con un usuario existente al sistema.
* Los usuarios podrán buscar el producto deseado.
* El sitio web debe tener una página principal donde se pueda observar los productos destacados.
* Los usuarios podrán observar su perfil de usuario.
* Los usuarios podrán elegir o agregar métodos de pago.
* El usuario podrá ver el historial de compras a través de su perfil de usuario.
* El usuario podrá realizar compras de manera sencilla por medio de métodos de pago distintos.
  + **Especificación de requerimientos.**
    - **Requerimientos funcionales:**
* Los usuarios podrán registrar un nuevo usuario o ingresar con un usuario existente al sistema.
* Los usuarios podrán buscar el producto deseado.
* El sitio web debe tener una página principal donde se pueda observar los productos destacados.
* Los usuarios podrán observar su perfil de usuario.
* Los usuarios podrán elegir o agregar métodos de pago.
* El usuario podrá ver el historial de compras a través de su perfil de usuario.
* El usuario podrá realizar compras de manera sencilla por medio de métodos de pago distintos.
  + - **Requerimientos no funcionales:**
* El sitio web debe ser compatible con los navegadores web más populares y adaptarse a diferentes tamaños de pantalla.
* El sitio web debe tener un tiempo de carga rápido.
  + - **Restricciones:**
* El proyecto debe completarse en 2 meses.
* El equipo deberá utilizar tecnologías de software libre y de código abierto.
  + **Priorización y asignación de requerimientos** 
    - **Registro y Login del usuario:**
* **Equipo desarrollador:**
  + Daniel Ames Camayo (Front - End).
  + Jesus Angel Saenz Chang (Front - End).
  + Gianlucas Amed Hinostroza Quispe (Back - End).
  + Jose Carlos Cjumo Chumbes (Back - End).
* **Testers:**
  + Carlos Alberto Castillo Bernal.
  + Jesus Angel Saenz Chang.
    - **Vista de productos por categoría:**
* **Equipo desarrollador:**
  + Daniel Ames Camayo (Front - End).
  + Jesus Angel Saenz Chang (Front - End).
  + Gianlucas Amed Hinostroza Quispe (Back - End).
  + Jose Carlos Cjumo Chumbes (Back - End).
    - **Vista del perfil de usuario:**
* **Equipo desarrollador:**
  + Daniel Ames Camayo (Front - End).
  + Jesus Angel Saenz Chang (Front - End).
  + Gianlucas Amed Hinostroza Quispe (Back - End).
  + Jose Carlos Cjumo Chumbes (Back - End).
    - **Agregar y ver métodos de pago:**
* **Equipo desarrollador:**
  + Daniel Ames Camayo (Front - End).
  + Jesus Angel Saenz Chang (Front - End).
  + Gianlucas Amed Hinostroza Quispe (Back - End).
  + Jose Carlos Cjumo Chumbes (Back - End).
    - **Ver historial de compras:**
* **Equipo desarrollador:**
  + Daniel Ames Camayo (Front - End).
  + Jesus Angel Saenz Chang (Front - End).
  + Gianlucas Amed Hinostroza Quispe (Back - End).
  + Jose Carlos Cjumo Chumbes (Back - End).
    - **Realizar compra:**
* **Equipo desarrollador:**
  + Daniel Ames Camayo (Front - End).
  + Jesus Angel Saenz Chang (Front - End).
  + Gianlucas Amed Hinostroza Quispe (Back - End).
  + Jose Carlos Cjumo Chumbes (Back - End).
* **Diseño.**
  + **Arquitectura de alto nivel**
    - **Descripción general del sistema.**

GoShop es una tienda virtual de implementos del hogar y materiales de construcción. El propósito de este proyecto es crear una tienda online para cumplir con los requerimientos de los clientes y facilitar la venta de estos productos.

Entre los objetivos del proyecto tenemos:

* + - * Entrega de las diferentes versiones del software en los plazos acordados.
      * Garantizar una buena experiencia de usuario.
      * Realización de pruebas de calidad rigurosas para garantizar el correcto funcionamiento del software.
    - **Componentes del sistema.**
* **Capa de presentación.**

Esta capa es la responsable de la interfaz de usuario del sistema, en el presente proyecto la información se mostrará a través de una página web.

* **Capa de lógica de negocio.**

Esta capa es la responsable de la lógica y el procesamiento del sistema. En el presente proyecto esta capa se encargará de realizar las validaciones de los datos ingresados por parte del usuario, además de las reglas del negocio, es decir, reglas para asegurarse de cumplir los objetivos del sistema. También se encargará (en ciertos casos) de acceder a los datos del sistema mediante el acceso a la base de datos.

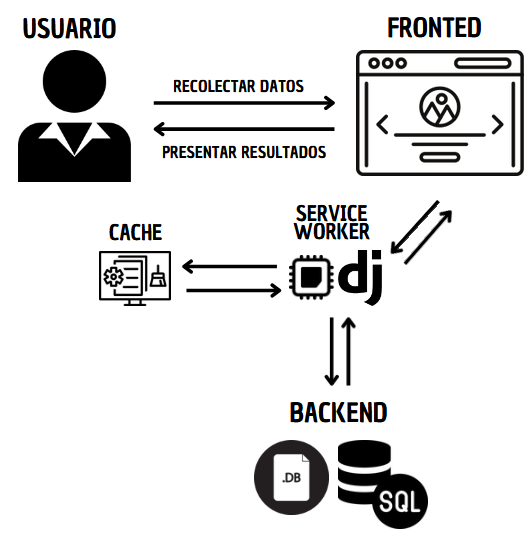
* **Capa de persistencia de datos.**

Esta capa es la responsable de almacenar y recuperar los datos del sistema. En este proyecto se utilizarán ORM (Object Relational Mapping) para crear tablas en la base de datos. Estos ORM son clases que permiten crear tablas, de este modo se logra abstraer la base de datos.

* + - **Interacciones entre los componentes.**

Las 3 capas mencionadas anteriormente se relacionan en todo momento durante el uso del software, ya que la capa de presentación se encargará de mostrar la información de la página y recoger los datos de entrada por parte del usuario, una vez hecho esto, se pasará a la capa lógica de negocio, la cual se asegurará de realizar diversas operaciones para cumplir con lo que el usuario desee en ese momento. Finalmente, la capa de persistencia de datos almacenará los datos que ya tiene y guardará los nuevos datos que el usuario le mande por medio de la capa de presentación.

* + - **Diagrama de arquitectura.**

****

* + - **Tecnologías utilizadas.**

Para el presente proyecto, GoShop, se utilizarán las siguientes tecnologías:

**Lenguajes de programación:**

* + - * Python.
      * JavaScript.

**Lenguajes para la construcción de la interfaz de usuario:**

* + - * HTML5.
      * CSS.

**Framework utilizado:**

* + - * Django (framework de Python).

**Base de datos:**

* + - * SqLite para las etapas de desarrollo y se plantea cambiar a una base de datos más grande cuando el proyecto pase a producción.
    - **Restricciones y presupuesto.**
* **Restricciones**

Para el proyecto se tuvieron en cuenta algunas restricciones que puedan retrasar o complicar el desarrollo del software:

* **El tiempo.** El tiempo que se le dedicará a cada funcionalidad podría no ser lo suficiente para su lanzamiento.
* **La tecnología**. En la actualidad, al haber bastantes herramientas y lenguajes de programación, es importante limitar o adaptarse a lo necesario para el desarrollo del proyecto.
* **Presupuesto**

Para la elaboración del presupuesto del proyecto se tuvieron en cuenta los siguientes gastos:

| **Gastos Fijos** | Electricidad | S/ 450 mensual por 3 meses |
| --- | --- | --- |
| Internet | S/ 200 mensual por 3 meses |
| Alquiler oficina | S/ 1,000 mensual por 3 meses |
| **Gastos Variables** | Compra de PC’s (7) | S/ 7000 pago único |
| Mantenimiento de PC’s | S/ 400 mensual por 3 meses |
| Compra de periféricos | S/ 700 pago único |
| **Total** | | S/ 13,850 |

* + - **Plan de implementación.**

El proyecto se implementará cumpliendo las fechas establecidas en el cronograma del proyecto, por lo tanto, se seguirán los siguientes plazos para los 3 hitos que se esperan en este proyecto:

* + - * **Hito N°1 - Fin de la primera iteración: 30/04/2023.**

Codificación de los 3 primeros requisitos, los cuales se realizarán en el plazo del 20 al 23 de abril del 2023.

* + - * **Hito N°2 - Fin de la segunda iteración: 24/05/2023.**

Codificación de requisitos 4 y 5, los cuales se realizarán en el plazo del 8 al 15 de mayo del 2022.

* + - * **Hito N°3 - Fin de la tercera iteración: 21/06/2023.**

Codificación de requisitos 6 y 7, los cuales se realizarán en el plazo del 29 de mayo al 08 de junio del 2023.

* + - **Consideraciones de calidad.**

La calidad del sistema se garantizará por la seguridad que tendrá, ya que se realizarán las respectivas pruebas unitarias tras la codificación de cada requisito. Además de esto, se seguirán buenas prácticas para el desarrollo del proyecto. A continuación, algunas de las consideraciones que se tendrán:

* **Escalabilidad.** El sistema podrá crecer sin dificultades.
* **Mantenibilidad.** El sistema podrá ser modificado, mejorado y corregido después de su implementación.
* **Fiabilidad.** Se procurará que el sistema funcione correctamente, es decir, sin errores.
* **Usabilidad.** Los usuarios tendrán facilidad de uso del sistema y la capacidad para realizar sus tareas de manera efectiva y eficiente.
* **Portabilidad.** Los usuarios podrán usar el sistema en diferentes plataformas y ambientes.
* **Evaluación.**
  + **Evaluación de riesgos.**

La evaluación de riesgos es un proceso importante en cualquier proyecto y en especial en GoShop, ya que ayudará a identificar los posibles riesgos que pueden afectar el éxito del proyecto y planificar estrategias para mitigarlos. Para llevar a cabo la evaluación de riesgos efectiva, se deberán seguir los siguientes pasos

* **Identificación de riesgos.** Identificar todos los posibles riesgos que pueden afectar al proyecto. Estos pueden incluir factores internos como falta de recursos, retrasos en la entrega, errores en la planificación, etc. o factores externos como cambios en las leyes o regulaciones, problemas económicos, eventos naturales, etc.
* **Análisis de riesgos.** Una vez que se han identificado los riesgos, es importante analizarlos para comprender su impacto en el proyecto y la probabilidad de que ocurran. Se puede utilizar una matriz de riesgos para priorizar los riesgos según su probabilidad e impacto.
* **Planificación de respuesta a riesgos.** Desarrollar estrategias para mitigar los riesgos identificados. Esto puede incluir la asignación de recursos adicionales, la reorganización de la planificación, la implementación de medidas de seguridad adicionales, etc.
* **Monitoreo y control de riesgos.** Es importante monitorear los riesgos identificados y las estrategias de mitigación en curso para asegurarse de que se están logrando los resultados deseados.
  + **Estimación de tiempo y recursos.**

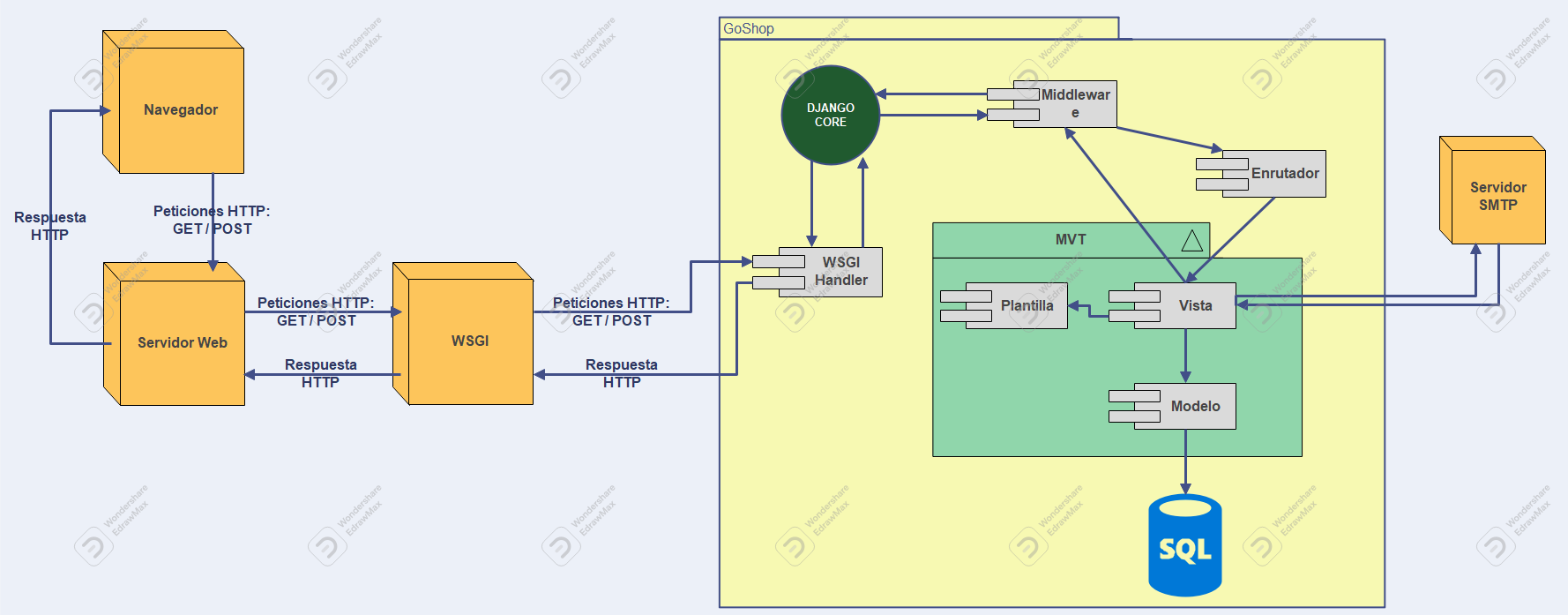
La estimación de tiempo y recursos es importante para garantizar que GoShop se complete dentro del plazo y presupuesto previstos. Para realizar una estimación precisa, es importante seguir los siguientes pasos:

* **Identificación de tareas.** Identificar todas las tareas necesarias para completar el proyecto.
* **Estimación de duración.** Estimar la duración de cada tarea. Esto puede hacerse utilizando la experiencia previa, el análisis de datos históricos o la opinión de expertos.
* **Asignación de recursos.** Asignar los recursos necesarios para completar cada tarea. Esto puede incluir recursos humanos, materiales y financieros.
* **Desarrollo del cronograma.** Desarrollar un cronograma detallado que muestre cuándo se llevarán a cabo cada tarea y cuánto tiempo se necesitará.
* **Monitoreo y control del progreso.** Es importante monitorear y controlar el progreso del proyecto para asegurarse de que se está cumpliendo con el cronograma previsto y hacer ajustes si es necesario.

1. **Elaboración.**

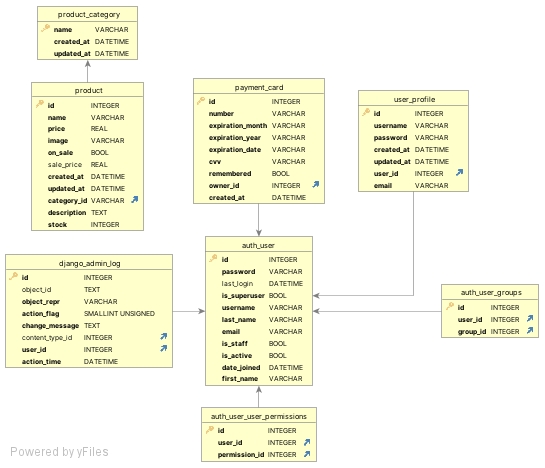
* **Objetivos y alcance de la fase.**
  + **Objetivos**

1. Crear un sitio web atractivo y funcional que brinde una experiencia de usuario satisfactoria a los visitantes.
2. Presentar productos y servicios de forma clara y atractiva para estimular las ventas y la conversión de clientes.
3. Desarrollar una estructura unificada y fácil de navegar que facilite a los clientes encontrar y comprar productos.
4. Crear contenido relevante y útil para los visitantes para aumentar el compromiso y la lealtad del cliente.
5. Integrar herramientas de pago seguras y confiables para brindar a los clientes una experiencia de compra segura y satisfactoria.
6. Desarrollar estrategias de marketing efectivas para promocionar su sitio web y atraer tráfico y clientes potenciales calificados.
   * **Alcance**
7. Definición de objetivos y público objetivo: Determinar los objetivos del sitio y las necesidades del público objetivo para desarrollar una estrategia adecuada.
8. Navegación del sitio y diseño estructural: Cree una estructura lógica y fácil de navegar para los visitantes del sitio.
9. Diseño Visual y Experiencia de Usuario: Crear un diseño visual de marca atractivo y consistente que permita a los usuarios navegar y encontrar fácilmente productos y servicios.
10. Desarrollo de contenido relevante: creación de contenido útil y relevante para los visitantes del sitio web, incluidas descripciones de productos, guías de compra, recomendaciones y otras herramientas que ayudan a los usuarios a tomar decisiones informadas.
11. Integración de herramientas y tecnología: integra herramientas de pago, gestión de inventario, análisis web, etc. para optimizar el rendimiento del sitio web.
12. Pruebas y optimización: pruebas para garantizar que el sitio web sea funcional y cumpla con los objetivos establecidos, y optimización continua para mejorar la experiencia del usuario y aumentar las conversiones.
13. Implementación de la estrategia de marketing: defina y ejecute una estrategia de marketing digital para promocionar el sitio web y generar tráfico y clientes potenciales calificados.

* **Especificación de requisitos detallados.**
  + **Registro y Login de usuario**
    1. Permitir a los usuarios registrarse en el sitio web a través de un formulario de registro.
    2. Verificar la información del usuario y validar su correo electrónico.
    3. Permitir a los usuarios iniciar sesión en el sitio web con su correo electrónico y contraseña.
  + **Vista de la página principal**
    1. Mostrar una página principal atractiva y funcional que presente los productos y servicios disponibles en el sitio web.
    2. Incluir un carrusel de imágenes de los productos destacados, promociones y ofertas especiales.
    3. Ofrecer una barra de búsqueda para que los usuarios puedan encontrar fácilmente los productos que buscan.
  + **Vista de productos por categoría**
    1. Ofrecer una vista de productos organizados por categorías (por ejemplo, "Herramientas", "Materiales de construcción", "Iluminación", etc.).
    2. Permitir a los usuarios filtrar los productos por precio, marca, disponibilidad y otros atributos relevantes.
    3. Incluir una vista detallada de cada producto que muestre su descripción, imagen, precio y especificaciones técnicas.
  + **Vista de perfil de usuario**
    1. Ofrecer una vista de perfil de usuario donde los usuarios puedan ver y editar su información personal (como nombre, dirección de correo electrónico y dirección de envío).
    2. Permitir a los usuarios ver su historial de compras, guardar productos en una lista de deseos y realizar seguimiento de sus pedidos.
  + **Agregar y ver métodos de pago**
    1. Permitir a los usuarios agregar y editar sus métodos de pago (como tarjeta de crédito o cuenta de PayPal).
    2. Garantizar la seguridad de la información de pago del usuario mediante el uso de un protocolo de seguridad seguro (HTTPS) y un procesador de pagos confiable.
  + **Ver historial de compras**
    1. Permitir a los usuarios ver un historial de sus compras anteriores, incluyendo detalles de la transacción (fecha, precio, productos comprados, dirección de envío, etc.).
  + **Realizar compras**
    1. Permitir a los usuarios agregar productos a su carrito de compras y finalizar su pedido mediante un proceso de pago seguro y confiable.
    2. Ofrecer opciones de envío y entrega para que los usuarios puedan elegir la forma más conveniente para ellos.
    3. Enviar una confirmación de pedido por correo electrónico al usuario después de realizar la compra.
* **Diagrama de arquitectura.**

En el diagrama de arquitectura se muestra como funciona el sistema de GoShop, esto bajo la forma en la que funciona Django, ya que el sitio web está funcionando bajo el patrón MVT. Se puede apreciar también las entidades externas, que, en este caso, serían el navegador, que realiza una petición HTTP al servidor web, el cual se maneja con el servidor de Django en etapa de desarrollo y en producción lo maneja el servidor de la nube. Luego de esto, la respuesta va al WSGI y luego al WSGI Handler, en este punto la petición ya se encuentra dentro del sistema GoShop. Luego de esto, el Django Core recibe las solicitudes HTTP procesadas por el WSGI Handler. Posteriormente, se continúa con el flujo que se muestra en el diagrama.Se puede ver también el funcionamiento del modelo MVT y como este se conecta con la base de datos y con el servidor SMTP.

* **Modelo de datos y diseño de base de datos.**
* Realizar un análisis detallado de los requisitos del sistema y las funcionalidades esperadas para determinar las entidades y relaciones necesarias en el modelo de datos.
* Diseñar una estructura de base de datos eficiente y escalable que cumpla con los requisitos del sistema y permite un almacenamiento adecuado de la información.
* Definir las tablas, campos, claves primarias y foráneas, así como las restricciones necesarias para mantener la integridad de los datos.
* Establecer las relaciones entre las diferentes entidades del sistema, como los usuarios, los productos, las compras y los métodos de pago.
* Considerar la seguridad de la base de datos, implementando medidas como la encriptación de datos sensibles y el control de acceso adecuado.

El siguiente diagrama Entidad - Relación está sujeto a cambios de acuerdo a las necesidades del cliente, sin embargo, es una buena referencia para el desarrollo de la base de datos:

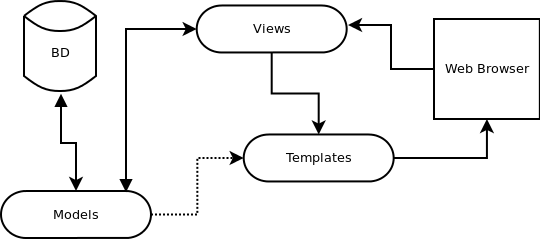
Se debe tener en cuenta, además, que el diagrama anterior no contiene todas las tablas de la base de datos del sistema, ya que Django por defecto ofrece algunas más para el correcto funcionamiento de un proyecto. Se puede apreciar algunas tablas importantes como user\_profile, el cual contendrá los datos importantes del usuario. Además, se tiene la tabla payment\_card, la cual contiene los datos de una tarjeta de crédito o de débito y será fundamental para cumplir uno de los requisitos del cliente. Finalmente, se tienen las tablas product y product\_category, las cuales serán importantes para manipular los productos que se muestran en la página web.

1. **Construcción.**

* **Objetivos y alcance de la fase.**
  + **Objetivos.**
    - El objetivo de la fase de construcción es desarrollar y construir el sistema utilizando el framework Django.
    - Utilizar buenas prácticas de desarrollo de software.
    - Implementar los requerimientos definidos en la fase de elaboración, se realizarán pruebas y se preparará el sistema para su despliegue en producción.
  + **Alcance.**
    - **Implementación de los requerimientos.** Durante esta fase, se llevará a cabo la implementación detallada de los requerimientos funcionales y no funcionales del sistema.
    - **Desarrollo de funcionalidades.** Se llevará a cabo la implementación de las funcionalidades específicas del sistema utilizando Django.
    - **Pruebas unitarias.** Se realizarán las pruebas unitarias respectivas para asegurar la calidad del sistema.
    - **Documentación y entregables.** Se realizará la documentación respectiva para las 3 iteraciones, además de un manual de usuario al finalizar el desarrollo y los respectivos entregables del código fuente.
* **Diseño detallado de componentes.**
  + **Identificación de componentes.**

Los componentes que interactúan en el proyecto GoShop son los que interactúan en un proyecto de Django, ya que este proyecto se realizará con dicho framework. Los componentes son los siguientes: modelos, vistas, plantillas, URLs, Middlewares, formularios y el panel de administración de Django.

* + **Descripción de componentes.**
    - **Modelos.** Los modelos definen la estructura de la base de datos del sistema mediante los ORM, como se mencionó anteriormente. Estos modelos son clases que se codifican en el archivo models.py.
    - **Vistas.** Las vistas son las encargadas de procesar las solicitudes HTTP y devolver las respuestas correspondientes. Estas se encuentran en el archivo views.py y permiten mostrar los templates al usuario y realizar operaciones con los modelos de la base de datos.
    - **Plantillas.** Las plantillas se utilizan para generar las páginas HTML que se envían al navegador del usuario. Las plantillas se definen en archivos HTML en la carpeta templates de la aplicación Django.
    - **URLs.** Las URLs son las rutas que el usuario puede visitar en la aplicación. Estas URLs se definen en el archivo urls.py en la carpeta del proyecto y también en la carpeta de la aplicación, ya que de esta manera se conecta el proyecto con la aplicación en la estructura de carpetas de Django.
    - **Middlewares.** Los middlewares son componentes que se ejecutan antes y después de las vistas y pueden modificar el comportamiento de la aplicación. En Django, un ejemplo de middleware es el middleware de autenticación, que verifica si el usuario que realiza una petición HTTP está autenticado en el sistema y, en caso contrario, realiza una acción que se controla mediante el archivo de las vistas.
    - **Formularios.** Los formularios se utilizan para recolectar datos del usuario y enviarlos al servidor. Los formularios pueden ser creados con módulos de Django o en la carpeta templates mediante HTML.
    - **Administración.** La aplicación Django incluye una interfaz de administración predeterminada que permite realizar tareas de administración en el sistema de forma sencilla.
  + **Interfaces de los componentes.**
    - **Modelos.** Se crearán modelos en Django para crear tablas en la base de datos, algunas de estas son los modelos Product, ProductCategory, UserProfile, entre otros.
    - **Vistas.** Las vistas a crear dependen de la funcionalidad que se está desarrollando. Se plantea crear diferentes archivos de vistas para mantener el código un poco más fragmentado y, por lo tanto, más comprensible. Algunas de los métodos de las vistas principales son shop(request), singup(request), singin(request), entre otros.
    - **Plantillas.** Las plantillas muestran los datos al usuario por medio de las vistas. Los documentos HTML a crear son shop.html, register.html, login.html, entre otros..
    - **URLs.** Algunas URLs que se utilizarán en el proyecto serán login/, register/, profile/, entre otros. Estos le servirán a Django en el archivo urls.py para identificar las URLs y conectarlo con las vistas y estas a su vez mandarán la información a los templates HTML.
    - **Middlewares.** Los middlewares que se utilizarán son los que incluye Django por defecto.
    - **Formularios.** Los formularios serán creados en HTML para un mayor control de estos, dado que una opción es crearlo mediante Django gracias a sus módulos de formularios, pero se prefiere una mejor personalización y se opta por crearlos mediante código HTML. Además, estos formularios serán importantes para mandar peticiones POST a las vistas y realizar diversas operaciones con los modelos de la base de datos.
    - **Administración.** El panel de administración de Django servirá para manipular los modelos de la base de datos mediante una interfaz intuitiva y sencilla. Por lo tanto, se podrán realizar pruebas en la base de datos sin realizar ninguna modificación de código. Esto es una facilidad que brinda Django para agilizar pequeñas pruebas que se quiera hacer en cuanto al ingreso de registros en la base de datos.
  + **Diagrama referencial de un proyecto de Django.**



* + **Detalles de implementación.**

Para realizar la implementación, se seguirá el diagrama de arquitectura y el diseño de la base de datos realizado en la fase anterior. Este último está sujeto a cambios a pedido del cliente, por lo que puede haber variaciones en la base de datos.

Además, para realizar la implementación, se seguirá el patrón de diseño usado por Django, es decir, el patrón MVT (Model - View - Template), este es un patrón de diseño que favorece la escalabilidad y la flexibilidad del proyecto. A continuación se detallará el flujo en el que trabaja este patrón de diseño:

**Modelo.** El modelo maneja todo lo relacionado con la información, esto incluye cómo acceder a esta, la validación, relación entre los datos y su comportamiento. En Django, el modelo está representado por las clases que se comportan como ORM para crear tablas en la base de datos, estas clases se crearán, por lo general, en el archivo views.py de la aplicación Django.

**Vista.** La vista sirve como enlace entre el modelo y el template (plantilla). La vista decide qué información será mostrada y por cual template. La vista está representada por el archivo views.py, en el cual se pueden realizar operaciones y consultas a la base de datos para mostrarlas en los templates.

**Plantilla (template).** Finalmente, el template decide cómo se mostrará la información. Los templates están representados en la carpeta “templates”, en la cual hay archivos HTML para la estructura de la página web.

* **Implementación y codificación de componentes.**

Etapa crítica en el desarrollo de software. Se llevan a cabo actividades clave para convertir los diseños y especificaciones en código ejecutable y funcional. A continuación, te proporcionaré más detalles sobre la implementación y codificación de componentes.

* **Análisis de diseños y especificaciones:** Antes de comenzar la codificación, es importante revisar y comprender los diseños y especificaciones detalladas del sistema o software. Esto implica analizar los diagramas de flujo, los diseños de interfaces, las descripciones de funciones y cualquier otra documentación relevante para obtener una comprensión clara de lo que se debe implementar.
* **Elección de lenguajes y tecnologías:** En esta etapa, se selecciona el lenguaje de programación y las tecnologías adecuadas para la implementación de los componentes. La elección puede basarse en requisitos técnicos, restricciones del proyecto y experiencia del equipo de desarrollo. Es importante considerar factores como el rendimiento, la escalabilidad y la compatibilidad con el entorno de implementación.
* **Codificación de componentes:** Durante esta fase, los desarrolladores se centran en la creación del código fuente de los componentes. Siguiendo los estándares de codificación y las buenas prácticas establecidas, escriben el código necesario para que los componentes funcionen según lo especificado. Es fundamental tener en cuenta la modularidad, la legibilidad, la reutilización de código y la documentación interna para facilitar el mantenimiento futuro.
* **Control de versiones:** Es importante utilizar un sistema de control de versiones, como Git, para gestionar el código fuente y rastrear los cambios realizados durante la implementación. Esto permite mantener un historial de versiones, colaborar eficientemente entre los miembros del equipo y revertir los cambios si es necesario.
* **Pruebas unitarias.**

Las pruebas unitarias se centran en la verificación de unidades individuales de código, como funciones, métodos o clases, de manera aislada. El objetivo principal es asegurar que cada unidad funcione correctamente según los requisitos y las especificaciones establecidas. Estas pruebas se realizan típicamente por los desarrolladores y siguen un enfoque de "caja blanca", lo que significa que se tiene en cuenta la estructura interna y la lógica del código.

La ejecución de dichas pruebas se seguirá los casos de prueba definidos. Esto implica ejecutar el código del componente y comparar los resultados obtenidos con los resultados esperados. Por ejemplo, el archivo "manage.py" es un script que proporciona diferentes comandos para administrar y ejecutar el proyecto. El comando "test" es utilizado para ejecutar pruebas automatizadas en el proyecto (python manage.py test).

# Transición.

# Objetivos de la fase.

* + - Implementar el software.
    - Entregar el software.
    - Brindar capacitación y soporte.
    - Asegurar la calidad.
    - Evaluar y obtener retroalimentación.

# Alcance de la fase.

* + - Desarrollar e integrar los componentes del software.
    - Realizar pruebas exhaustivas y depuración.
    - Preparar documentación necesaria, como manuales de usuario y guías de instalación.
    - Proporcionar capacitación y soporte técnico.
    - Asegurar la calidad del software mediante pruebas adicionales y control de calidad.
    - Obtener retroalimentación del cliente o usuario final y realizar ajustes según sea necesario.

# Pruebas de Aceptación.

* + - Diseñar y ejecutar pruebas que validen si el software cumple con los requisitos y expectativas del cliente o usuario final.
    - Verificar que todas las funcionalidades y características requeridas estén implementadas correctamente.
    - Realizar pruebas de usabilidad para evaluar la facilidad de uso y la experiencia del usuario.
    - Comprobar que se satisfacen los criterios de aceptación definidos previamente.

# Pruebas de Validación.

* + - Verificar que el software cumple con los estándares y normas establecidos.
    - Validar la interoperabilidad del software con otros sistemas o componentes con los que debe interactuar.
    - Realizar pruebas de integración para verificar que el software funciona correctamente con otros sistemas o módulos relacionados.

# Modelo de despliegue del sistema.

# Plan de despliegue.

Se seguirán los siguientes pasos para el despliegue del software en el entorno de producción:

* + **Preparación.**
    - Verificar el correcto funcionamiento del software para el despliegue.
    - Preparar la base de datos y cargar los datos iniciales. Además de cambiar la base de datos de SQLite a PostgreSQL en el archivo settings.py.
  + **Implementación.**
    - Descargar y configurar el código fuente del proyecto en el servidor de la nube.
    - Realizar los cambios necesarios en el servicio de la nube para asegurar el correcto funcionamiento del sitio web.
    - Configurar los archivos de configuración y secretos del proyecto. Este paso es importante para la seguridad de los datos y para el correcto funcionamiento del sitio web en el servicio seleccionado.
  + **Verificación.**
    - Realizar pruebas de rendimiento del sitio web.
    - Verificar que todas las funcionalidades estén en ejecución y sin errores.
    - Realizar pruebas de funcionalidad en el sitio web para asegurar que todo esté funcionando correctamente.

# Especificaciones de despliegue.

* + **Servidor de presentación.**

El servidor de presentación es responsable de servir el contenido estático de la aplicación web, como los archivos CSS y JavaScript. Los archivos HTML son considerados archivos estáticos o dinámicos dependiendo de si necesitan de una vista (views.py) para su correcto funcionamiento. En este caso, se usará render.com como servidor de presentación, entonces no sería necesario configurar un servidor de presentación adicional.

* + **Servidor de aplicación.**

El servidor de aplicación es el encargado de procesar las solicitudes dinámicas de la aplicación web y devolver las respuestas correspondientes.

Render.com también puede actuar como servidor de aplicación, ya que se puede conectar a un repositorio de Git para una implementación continua.

* + **Servidor de base de datos.**

El servidor de base de datos es el encargado de almacenar y recuperar los datos de la aplicación.

Render.com también ofrece soporte para la base de datos PostgreSQL, por lo que se va a configurar una instancia de base de datos en Render y se va a configurar la aplicación Django para conectarse a la base de datos. Para este proyecto se utilizará el plan gratuito del servidor de base de datos, el cual cuenta con las siguientes características:

* + CPU: 0.1
  + RAM: 256 MB
  + Espacio en disco: 1GB
  + Base de datos:: PostgreSQL 15.

# Documentación desarrollada en el proyecto.

Se debe haber cumplido con la documentación establecida en el cronograma de proyecto al término de este, los documentos son los siguientes:

* + Plan de proyecto.
  + Plan de Fase.
  + Documento de Arquitectura del software.
  + Documento de Especificación de UI.
  + Documento de Guía de Estilos.
  + Documento de Especificación de la Base de Datos.
  + Documento de especificación de requisitos 1.
  + Documento de especificación de requisitos 2.
  + Documento de especificación de requisitos 3.
  + Documento de especificación de requisitos 4.
  + Documento de especificación de requisitos 5.
  + Documento de especificación de requisitos 6.
  + Documento de especificación de requisitos 7.
  + Plan de Iteración 1.
  + Plan de Iteración 2.
  + Plan de Iteración 3.
  + Documento de Casos de uso 1.
  + Documento de Casos de uso 2.
  + Documento de Casos de uso 3.
  + Documento de Informe de Pruebas 1.
  + Documento de Informe de Pruebas 2.
  + Documento de Informe de Pruebas 3.
  + Reporte de desarrollo de software 1.
  + Reporte de desarrollo de software 2.
  + Reporte de desarrollo de software 3.
  + Manual de usuario.
  + Acta de cierre del proyecto.

# 