**Descripción del Proyecto: Carro de Compras MasterBikes**

El **proyecto MasterBikes** es una aplicación web de comercio electrónico que permite a los usuarios navegar y comprar bicicletas. Una de las funcionalidades clave de esta aplicación es el carro de compras, que permite a los usuarios agregar, eliminar y gestionar productos antes de realizar una compra. La implementación de esta funcionalidad ilustra el funcionamiento de la tecnología cliente-servidor y la utilización de los conceptos básicos de TCP/IP, diferenciando claramente las tareas del **Front-End** y **Back-End**.

**Tecnología Cliente-Servidor**

En una arquitectura cliente-servidor, la comunicación se divide entre dos tipos de sistemas:

**Cliente:** El cliente es la aplicación que los usuarios finales interactúan con sus navegadores web. En el contexto de MasterBikes, el cliente incluye todas las interacciones del usuario con la interfaz de usuario, como agregar productos al carro de compras, ver detalles del producto y proceder al pago.

**Servidor:** El servidor es donde reside la lógica de la aplicación, el procesamiento de datos y el almacenamiento. En MasterBikes, el servidor maneja solicitudes del cliente, accede a la base de datos para obtener información sobre productos, guarda las sesiones de usuario y procesa las acciones relacionadas con el carro de compras.

**Elementos Básicos de TCP/IP**

La comunicación entre el cliente y el servidor se realiza utilizando el conjunto de protocolos TCP/IP. Aquí se destacan algunos elementos clave:

Protocolo TCP (Transmission Control Protocol): TCP asegura la entrega fiable y ordenada de los datos entre el cliente y el servidor. Por ejemplo, cuando un usuario agrega un producto al carro, esta acción se envía al servidor usando TCP, garantizando que la solicitud llegue correctamente.

Protocolo IP (Internet Protocol): IP es responsable de dirigir y enrutar los paquetes de datos entre el cliente y el servidor. En el caso de MasterBikes, la dirección IP del servidor permite que el cliente sepa dónde enviar las solicitudes y dónde recibir las respuestas.

HTTP/HTTPS (HyperText Transfer Protocol / Secure HyperText Transfer Protocol): Estos protocolos se utilizan para la comunicación entre el navegador web del cliente y el servidor web. HTTPS añade una capa de seguridad mediante la encriptación de los datos transmitidos. MasterBikes utiliza HTTPS para proteger la información sensible del usuario, como los detalles de la compra y los datos personales.

**Front-End vs Back-End**

**Front-End**:

Descripción: El front-end se refiere a todo lo que el usuario interactúa directamente en su navegador web. Incluye la interfaz de usuario y la experiencia del usuario (UI/UX).

Tecnologías Utilizadas: HTML, CSS, JavaScript y frameworks/librerías como React, Angular o Vue.js.

Responsabilidades:

Mostrar productos de manera atractiva y fácil de usar.

Permitir a los usuarios agregar y eliminar productos del carro.

Actualizar dinámicamente la vista del carro de compras.

Validar y enviar formularios de compra.

**Back-End**:

Descripción: El back-end se refiere a todo lo que sucede detrás de escena en el servidor. Incluye la lógica de la aplicación, la gestión de bases de datos y la autenticación de usuarios.

Tecnologías Utilizadas: Python, Django, bases de datos SQL (como PostgreSQL o MySQL).

Responsabilidades:

Procesar solicitudes del cliente, como agregar o eliminar productos del carro.

Manejar la lógica de negocio, como calcular el total del carro de compras.

Acceder y modificar datos en la base de datos.

Gestionar las sesiones del usuario para mantener el estado del carro de compras.

Proporcionar respuestas a las solicitudes del cliente, devolviendo vistas HTML o datos en formato JSON.

**Capas de un Proyecto Web utilizando el Framework Django**

En el desarrollo de aplicaciones web utilizando el framework Django, es común dividir la aplicación en tres capas principales: Presentación, Negocios y Datos. Cada capa tiene una responsabilidad específica y contribuye a la separación de preocupaciones, lo que facilita el mantenimiento y escalabilidad del proyecto.

**1. Capa de Presentación**

La capa de presentación es responsable de interactuar directamente con el usuario. Incluye todo lo que el usuario ve y con lo que interactúa en la aplicación web.

**Responsabilidades:**

* Mostrar datos al usuario en un formato comprensible y atractivo.
* Recibir y procesar entradas del usuario.
* Validar entradas antes de enviarlas a la capa de negocios.
* Gestionar la lógica de presentación, como la navegación entre páginas.
* Componentes en Django:

Templates: Los archivos HTML que definen la estructura y diseño de las páginas web.

Views: Las funciones o clases que controlan la lógica de presentación. Reciben solicitudes del navegador, interactúan con la capa de negocios y devuelven una respuesta HTTP (usualmente un template renderizado).

Static Files: Archivos estáticos como CSS, JavaScript e imágenes que son utilizados en las plantillas.

**2. Capa de Negocios**

La capa de negocios contiene la lógica central de la aplicación. Es donde se implementan las reglas de negocio y la lógica de procesamiento de datos.

**Responsabilidades:**

* Implementar la lógica de negocio de la aplicación.
* Validar datos y aplicar reglas de negocio.
* Coordinar entre la capa de presentación y la capa de datos.

**Componentes en Django:**

Models: Definen la estructura de los datos y proporcionan métodos para interactuar con la base de datos.

Forms: Gestionan la entrada del usuario y la validación de datos.

Custom Business Logic: Puede estar en los modelos, vistas, o en archivos de utilidades separados.

**3. Capa de Datos**

La capa de datos se encarga del almacenamiento y acceso a los datos. Esta capa interactúa directamente con la base de datos y es responsable de la persistencia de los datos.

**Responsabilidades:**

* Definir la estructura de la base de datos.
* Realizar operaciones de lectura/escritura en la base de datos.
* Proporcionar una interfaz para que la capa de negocios interactúe con los datos.
* Componentes en Django:

Database Models: Las clases del modelo Django, que definen la estructura de las tablas de la base de datos.

ORM (Object-Relational Mapping): Django ORM facilita las operaciones de base de datos mediante métodos de alto nivel en las instancias del modelo.

Database Migrations: Gestionan los cambios en la estructura de la base de datos a lo largo del desarrollo del proyecto.

**Conclusión**

**La división de una aplicación web en capas de Presentación, Negocios y Datos utilizando Django permite una clara separación de responsabilidades. Esto no solo facilita el desarrollo y mantenimiento, sino que también hace que la aplicación sea más modular y escalable. Django, con su enfoque de "baterías incluidas", proporciona herramientas y componentes para gestionar cada una de estas capas de manera eficiente y coherente.**