Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2024

PROYECTO 2

202010367 – Marcelo André Juarez Alfaro
202200236 – Kevin David Tobar Barrientos

Resumen

El ensayo presenta una solución al de una aplicación web la cual fue desarrollada en proyecto anterior la cual evoluciono a una aplicación basada en cliente-servidor. Este cambio permitirá ofrecer estadísticas visuales de productos, usuarios y compras, mediante gráficos dinámicos y entendibles, mejorando así la experiencia del usuario.
El objetivo es desarrollar un software que pueda ser consumido como un servicio desde internet. Este software transformará archivos XML a formato JSON, los almacenará y los enviará al servidor.
La aplicación se basará en una arquitectura cliente-servidor. El cliente será una aplicación web que permitirá la interacción del usuario, mientras que el servidor proporcionará los servicios necesarios para procesar y almacenar datos, para lo cual se usará Django para hacer la solución del lado del cliente, donde se mostrar las vistas al usuario y Flask para el lado del servidor donde se manejara la lógica de la aplicación como también las validaciones .

Palabras clave
JSON, XML, Cliente, Servidor

Abstract

The essay presents a solution to that of a web application which was developed in previous project which evolved to a client-server based application. This change will allow to offer visual statistics of products, users and purchases, through dynamic and understandable graphics, thus improving the user experience.
The goal is to develop a software that can be consumed as a service from the internet. This software will transform XML files to JSON format, store them and send them to the server.
The application will be based on a client-server architecture. The client will be a web application that will allow user interaction, while the server will provide the necessary services to process and store data, for which Django will be used to make the solution on the client side, where the views will be displayed to the user and Flask for the server side where the logic of the application will be handled as well as the validations.

Keywords
JSON, XML, Client, Server

Introducción

En el proyecto anterior, IPCmarket se desarrolló como una aplicación de escritorio que alcanzó un notable éxito. Ahora, con el objetivo de expandir su alcance y mejorar la experiencia del usuario, se ha decidido evolucionar esta aplicación a un entorno web basado en una arquitectura cliente-servidor. Esta transición permitirá ofrecer estadísticas visuales de productos, usuarios y compras mediante gráficos dinámicos y entendibles.

El objetivo principal es desarrollar un software accesible como un servicio desde internet. Este software se encargará de transformar archivos XML a formato JSON, almacenarlos y enviarlos al servidor. La arquitectura cliente-servidor será clave para esta implementación: el cliente será una aplicación web que facilitará la interacción del usuario, mientras que el servidor proporcionará los servicios necesarios para procesar y almacenar datos.

Para lograr una solución eficiente y escalable, se han elegido dos frameworks populares: Django para el lado del cliente y Flask para el lado del servidor.

Desarrollo del tema

Django: Lado del Cliente

Django es un framework de alto nivel para el desarrollo de aplicaciones web en Python. Su capacidad para manejar la administración de usuarios, formularios, y su sistema de plantillas robusto lo convierten en una excelente opción para desarrollar el frontend de IPCmarket. Con Django, se crearán las vistas que permitirán a los usuarios interactuar con la aplicación, realizar compras, ver estadísticas y gestionar sus datos.

Funcionalidades Implementadas con Django:

1. **Gestión de Usuarios**: Se implementarán formularios de inicio de sesión y registro de usuarios.
2. **Visualización de Productos**: Los usuarios podrán ver y buscar productos disponibles.
3. **Carrito de Compras**: Los usuarios podrán agregar productos a su carrito y confirmar compras.
4. **Estadísticas Dinámicas**: Se generarán gráficos dinámicos para mostrar estadísticas de productos y compras.

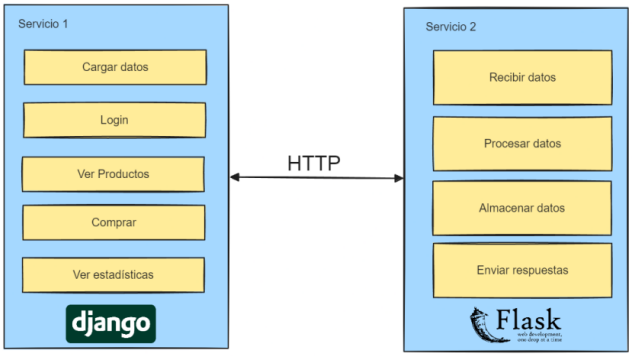

Figura 1. Arquitectura de la apliacion

Fuente:Auxiliar Rodrigo Alejandro Hernández de León

Flask: Lado del Servidor

Flask es un microframework ligero y flexible para el desarrollo de aplicaciones web en Python. Su simplicidad y modularidad permiten un control preciso sobre las funcionalidades del servidor, haciéndolo ideal para manejar la lógica de la aplicación y las validaciones necesarias en IPCmarket.

Funcionalidades Implementadas con Flask:

1. **Transformación de Datos**: Procesamiento y conversión de archivos XML a JSON.
2. **Almacenamiento de Datos**: Gestión de la persistencia de datos en formato XML.
3. **Validaciones**: Verificación de integridad y formato de los datos entrantes.
4. **API RESTful**: Implementación de endpoints para interactuar con el frontend, como carga de usuarios, productos, empleados y actividades, así como gestión del carrito de compras y procesamiento de compras.

Integración de Django y Flask

La integración de Django y Flask permitirá separar claramente las responsabilidades de la aplicación, manteniendo un frontend interactivo y un backend robusto. Django manejará la presentación y las interacciones del usuario, mientras que Flask gestionará la lógica de negocio y la validación de datos.

1. **Comunicación entre Frontend y Backend**: Utilizando API RESTful proporcionadas por Flask, Django podrá enviar y recibir datos, asegurando una sincronización eficiente entre las dos capas.
2. **Transformación y Persistencia de Datos**: Flask se encargará de transformar los datos XML a JSON y almacenarlos, mientras que Django los presentará de manera clara y comprensible para el usuario final.
3. **Seguridad y Validaciones**: Ambas capas implementarán mecanismos de seguridad y validación para garantizar la integridad y confidencialidad de la información.

Arquitectura General

La arquitectura cliente-servidor de la nueva versión de IPCmarket se estructura de la siguiente manera:

1. **Frontend (Cliente)**: Implementado con Django, donde los usuarios interactúan con la aplicación.
2. **Backend (Servidor)**: Implementado con Flask, donde se maneja la lógica de negocio, procesamiento de datos y validaciones.
3. **Persistencia de Datos**: Los datos se almacenan en archivos XML, transformados y gestionados por el backend.

Gestión de Usuarios

Django proporciona un sistema de autenticación y autorización. Se implementaron formularios para el registro y login de usuarios:

- **Registro de Usuarios**: Formularios para la creación de nuevas cuentas de usuario, con validación de campos como email y teléfono.
- **Inicio de Sesión**: Autenticación de usuarios existentes utilizando las credenciales almacenadas.

Visualización y Gestión de Productos

- **Listado de Productos**: Los usuarios pueden ver un catálogo de productos, filtrarlos por categorías y buscar productos específicos.
- **Detalle de Producto**: Vista detallada de cada producto, incluyendo descripción, precio, imagen y cantidad disponible.

Carrito de Compras

- **Agregar al Carrito**: Los usuarios pueden agregar productos al carrito de compras, especificando la cantidad deseada.
- **Ver Carrito**: Visualización del carrito de compras en formato XML.

- **Confirmar Compra**: Finalización del proceso de compra, donde se valida la disponibilidad de los productos y se registra la compra.

Estadísticas Dinámicas

- **Gráficos de Barras**: Implementación de gráficos para mostrar el top 3 de categorías con más productos y el top 3 de productos con mayor cantidad disponible.

Transformación y Almacenamiento de Datos

Flask se encarga de procesar los archivos XML recibidos, transformarlos a JSON y almacenarlos. Se implementaron las siguientes rutas:

- **/cargausuarios**: Procesa el XML de usuarios y los almacena.
- **/cargaproductos**: Procesa el XML de productos y los almacena.
- **/cargaempleados**: Procesa el XML de empleados y los almacena.
- **/cargaactividades**: Procesa el XML de actividades y los almacena.

API RESTful

Flask proporciona endpoints para la interacción entre el frontend y el backend:

- **/login**: Gestiona la autenticación de usuarios.
- **/añadircarrito**: Añade productos al carrito de compras.
- **/comprar**: Procesa la compra de productos en el carrito.
- **/obtenerproductos**: Devuelve la lista de productos almacenados.
- **/obtenerusuarios**: Devuelve la lista de usuarios almacenados.

Transformación y Persistencia de Datos

Flask transforma los datos XML a JSON y los almacena en un archivo XML general de persistencia. Django puede solicitar estos datos y presentarlos al usuario en una interfaz amigable.

Seguridad y Validaciones

Ambos frameworks implementan medidas de seguridad y validación. Django se encarga de proteger las vistas del frontend, mientras que Flask asegura la integridad de los datos en el backend.

Conclusiones

La combinación de Django para el frontend y Flask para el backend ha permitido la creación de una arquitectura cliente-servidor sólida y eficiente. Esta estructura facilita la interacción del usuario con la aplicación y asegura un procesamiento y almacenamiento efectivo de los datos.

implementación del patrón MVT en Django ha sido crucial para mantener una separación clara de responsabilidades dentro de la aplicación. Los modelos gestionan la lógica de datos, las vistas manejan la lógica de negocio y las plantillas presentan la información de manera dinámica y amigable para el usuario.

Flask ha demostrado ser una herramienta flexible y potente para manejar la lógica del servidor. Su capacidad para procesar archivos XML, transformarlos a JSON y validarlos antes de almacenarlos ha asegurado la integridad y consistencia de los datos, lo que es fundamental para el correcto funcionamiento de la aplicación.

La evolución de IPCmarket de una aplicación de escritorio a una aplicación web interactiva ha

mejorado significativamente la experiencia del usuario. La capacidad de visualizar estadísticas dinámicas, gestionar productos y compras de manera intuitiva y segura, y el soporte para múltiples tipos de usuarios (administradores y compradores) han hecho que la aplicación sea más accesible y funcional para una audiencia más amplia.

Referencias bibliográficas

José Portilla, (2020). Python for Data Science and Machine Learning Bootcamp. Udemy.

Miguel Grinberg, (2018). Flask Web Development: Developing Web Applications with Python. O'Reilly Media.

Django Software Foundation. Django Documentation. Disponible en: https://docs.djangoproject.com/en/stable/. Accedido el 22 de junio de 2024.

Pallets Projects. Flask Documentation. Disponible en: https://flask.palletsprojects.com/en/2.0.x/. Accedido el 22 de junio de 2024.

Anexos