PROYECTO 2 “IPCMARKET”

202210483 – Angely Lucrecia García Martínez

Resumen

En la era digital, la implementación de sistemas electrónicos para la gestión de mercados se ha convertido en una necesidad fundamental. Este ensayo presenta el desarrollo de IPCmarket, una aplicación web diseñada para administrar un mercado electrónico.

La aplicación, haciendo uso de los frameworks Flask y Django, permite a los administradores gestionar de manera efectiva, mediante archivos XML, los productos, empleados y actividades de la empresa, así como la autenticación de usuarios. A su vez, los clientes pueden navegar por los productos, agregar artículos a su carrito de compras y realizar transacciones de manera segura.

La arquitectura cliente-servidor subyace en el diseño de IPCmarket, donde el programa cliente solicita servicios o recursos del programa servidor. Esta aplicación es capaz de transformar los archivos XML actuales hacia un formato JSON, almacenando el JSON previo al envío al servidor.

Palabras clave
XML, usuarios, productos, empleados, actividades, Flask, Django, endpoints.

Abstract

In the digital era, the implementation of electronic systems for market management has become a fundamental necessity. This paper presents the development of IPCmarket, a web application designed to manage an electronic marketplace.

The application, making use of the Flask and Django frameworks, allows administrators to effectively manage, through XML files, the company's products, employees and activities, as well as user authentication. In turn, customers can browse products, add items to their shopping cart and perform transactions securely.

The client-server architecture underlies the design of IPCmarket, where the client program requests services or resources from the server program. This application is capable of transforming the actual XML files into a JSON format, storing the JSON prior to sending it to the server.

Keywords
XML, users, products, employees, activities, Flask, Django, endpoints.

Introducción

IPCmarket, una aplicación web desarrollada para administrar un mercado electrónico, responde a la necesidad de una gestión eficiente y una experiencia de cliente optimizada, ofreciendo una solución tecnológica avanzada. Se presentará un panorama general de la aplicación, explicando sus bases teóricas y la perspectiva adoptada en su diseño y desarrollo. Además, se discutirán los aspectos relevantes que sustentan la argumentación, tales como la elección de Flask y Django para la creación de una interfaz de usuario intuitiva, el uso de archivos XML para la gestión de productos, empleados y actividades de la empresa, y la transformación de estos archivos a formato JSON para estandarizar el almacenamiento y facilitar la migración a una plataforma eCommerce.

Desarrollo del tema

IPCmarket fue desarrollado utilizando Flask y Django para la creación de una interfaz web robusta e intuitiva. Estas herramientas fueron seleccionadas por su capacidad de integrar funcionalidades avanzadas con una interfaz amigable para el usuario. Para tener acceso al sistema, se tienen dos tipos de usuarios: el administrador encargado de la gestión de recursos y compras, y el usuario, quien tiene la oportunidad de ver los productos disponibles y realizar sus compras. A continuación, se detallan las funcionalidades de la aplicación.

1. Funcionalidades como Administrador
    a. Gestión de Productos: Los administradores pueden agregar productos al sistema mediante la carga de archivos XML, los cuales son transformados a formato JSON para su almacenamiento y procesamiento. Esta funcionalidad asegura que la información del inventario esté siempre actualizada.
    b. Gestión de Empleados y Actividades: La aplicación permite la administración de los empleados, incluyendo la asignación de roles y la gestión de sus actividades.
    c. Autenticación de Usuarios: Se implementa un sistema de autenticación que permite que los únicos usuarios cargados al sistema tengan acceso a la aplicación.
2. Funcionalidades como Usuario
    a. Navegación de Productos: Los clientes pueden explorar los productos disponibles en el mercado, así como los detalles de los mismos: nombre, descripción, precio, categoría y la imagen del producto.
    b. Realizar compras: El cliente puede agregar productos a su carrito de compras y visualizarlo para confirma su pedido con la información de los productos seleccionados.
3. Generación de reportes

    Una de las características distintivas de IPCmarket es la capacidad de generar reportes detallados sobre las compras que los clientes han realizado, asimismo el administrador puede obtener reportes estadísticos sobre los 3 productos con más cantidad disponible y las 3 categorías con más productos para tener una mejor referencia sobre su inventario.

Arquitectura Cliente-Servidor
La arquitectura del programa se compone de un cliente y un servidor. El software puede ser consumido desde internet como un servicio. A continuación se detalla la estructura del programa.

A. Servicio Frontend

El objetivo principal del servicio frontend es proporcionar una interfaz de usuario amigable y eficiente para interactuar con el sistema. A través de esta interfaz, los usuarios pueden realizar diversas operaciones como iniciar sesión, visualizar y cargar productos, gestionar carritos de compra, entre otros. Todo esto se logra mediante una serie de vistas y formularios que interactúan con el backend para recuperar, procesar y mostrar la información de manera coherente y presentable. Django, conocido por su arquitectura MTV (Model-Template-View), nos proporciona una estructura robusta y escalable para la creación de aplicaciones web completas considerando que se utilizaron las siguientes vistas (views):

a. Index (request)
   Muestra la página principal de la aplicación.
b. Login (request)
   Muestra el formulario de inicio de sesión para que los usuarios puedan ingresar sus credenciales.
c. Signin (request)
   Recibe las credenciales del usuario, las envía al backend para autenticación y redirige al usuario según su rol (administrador o cliente).
d. adminView (request)
   Muestra la interfaz principal del administrador después de iniciar sesión.
e. userView (request)
   Muestra la interfaz principal del usuario después de iniciar sesión.
f. adminCarga (request)
   Muestra la interfaz para que los administradores carguen archivos XML.
g. cargaXML (request)
   Permite a los administradores subir archivos XML y ver su contenido en la interfaz.
h. enviarUsuarios (request)
   Permite a los administradores cargar masivamente usuarios en el sistema.
i. enviarProductos (request)
   Permite a los administradores cargar masivamente productos en el sistema.
j. enviarEmpleados (request)
   Permite a los administradores cargar masivamente empleados en el sistema.
k. verProductos (request)
   Muestra todos los productos disponibles en el sistema a los usuarios y administradores.
l. comprar (request)
   Muestra todos los productos disponibles para que los usuarios puedan añadirlos al carrito.
m. añadirCarrito (request)
   Permite a los usuarios agregar productos específicos a su carrito de compras.
n. confirmarCompra (request)
   Permite a los usuarios finalizar la compra de los productos en su carrito.
o. estadísticas (request)
   Muestra la interfaz para ver estadísticas de productos y categorías.

B. Servicio Backend

Este servicio consiste en una API que brinda servicios utilizando el protocolo HTTP, su funcionalidad principal es procesar los datos recibidos del servicio de frontend, luego de procesar los datos es necesario que estos sean almacenados en un archivo XML, este servicio también tiene la funcionalidad de devolver los datos que fueron almacenados para que sean mostrados. Este servidor utiliza los siguientes endpoints:

a. GET /
   Descripción: Este endpoint devuelve un mensaje indicando que la API con Python y Flask está funcionando correctamente. Es utilizado para verificar el estado de la API y confirmar que el servidor está activo.

b. POST /login
   Los usuarios envían sus credenciales (ID y contraseña) mediante una solicitud POST para autenticarse como administradores o clientes. Dependiendo de las credenciales proporcionadas, se devuelve un mensaje indicando el éxito o el fracaso del inicio de sesión.
c. POST /cargaUsuarios
   Los administradores utilizan este endpoint para importar múltiples registros de usuarios al sistema simultáneamente. Cada usuario es extraído del XML proporcionado y almacenado en la base de datos del sistema.
d. GET /verUsuarios
   Los administradores pueden consultar esta lista para ver todos los usuarios registrados, incluyendo detalles como ID, nombre, edad, email y teléfono.
e. POST /cargaProductos
   Los administradores utilizan este endpoint para importar productos en masa al sistema desde un archivo XML. Cada producto, incluyendo su nombre, precio, descripción, categoría, cantidad disponible y imagen, es extraído del XML y almacenado en la base de datos.
f. GET /verProductos
   Los clientes y administradores pueden consultar esta lista para ver todos los productos disponibles para la venta, junto con detalles como nombre, precio, descripción, categoría y cantidad disponible.
g. POST /cargaEmpleados
   Los administradores utilizan este endpoint para importar múltiples registros de empleados al sistema desde un archivo XML. Cada empleado, incluyendo su código, nombre y puesto, es extraído del XML y almacenado en la base de datos.
h. GET /verEmpleados
   Los administradores pueden consultar esta lista para ver todos los empleados registrados en la empresa, incluyendo detalles como código, nombre y puesto.
i. POST /cargaActividades
   Los administradores utilizan este endpoint para importar múltiples registros de actividades al sistema desde un archivo XML. Cada actividad, incluyendo su nombre, descripción, empleado asignado, día de la semana y hora, es extraída del XML y almacenada en la base de datos.
j. GET /verActividades
   Los administradores pueden consultar esta ruta para ver todas las actividades programadas en la empresa. La lista incluye detalles como el nombre de la actividad, descripción, empleado asignado y hora de inicio.
k. POST /añadirCarrito
   Se utiliza este endpoint para seleccionar productos específicos y añadirlos a su carrito de compras. Deben proporcionar el ID del usuario, el ID del producto y la cantidad deseada. Si la operación es exitosa, se devuelve un mensaje de confirmación.
l. GET /verCarrito
   Esta ruta es utilizada para ver todos los productos que los clientes han agregado a su carrito de compras. Deben proporcionar el ID del usuario como parámetro de consulta.
m. GET /verListaCompras
   Ruta que es consultada para ver un historial de todas las compras realizadas por un usuario específico. Deben proporcionar el ID del usuario como parámetro de consulta.
n. GET /generarXMLCarrito
   Los clientes pueden visualizar un archivo XML que contiene todos los detalles de los productos en su carrito de compras

actual. Deben proporcionar el ID del usuario como parámetro de consulta.

o. POST /confirmarCompra
Los clientes utilizan este endpoint para finalizar una compra. El sistema verifica el carrito del usuario, registra la compra en el historial y vacía el carrito. Se devuelve un mensaje de confirmación o error según el resultado de la operación

p. GET /generarReporte
Los administradores utilizan este endpoint para generar un informe detallado en formato XML que contiene todas las compras realizadas en el sistema. El reporte incluye información como el ID del usuario, nombre del usuario, productos comprados y el total gastado.

q. GET /generarXMLActividadesHoy
Los administradores pueden visualizar un archivo XML que contiene todas las actividades programadas para el día actual. El archivo incluye detalles como el nombre de la actividad, descripción, empleado asignado y hora de inicio.

r. GET /top3Categorias
Los administradores utilizan este endpoint para obtener información sobre las categorías más demandadas en el sistema. La respuesta incluye las tres categorías con más productos en el inventario.

s. GET /top3Productos
Los administradores utilizan este endpoint para identificar los productos más vendidos en el sistema. La respuesta incluye los tres productos con más cantidades disponibles en el inventario.

El uso de archivos XML es fundamental para la transferencia de datos entre el frontend y el backend. El objetivo principal del uso de archivos XML en nuestro sistema es facilitar la carga masiva de datos y la generación de reportes estructurados.

Esta capacidad de estructuración es especialmente útil en aplicaciones que requieren la transferencia de datos complejos y jerárquicos, como es el caso de nuestra aplicación web.

Para el procesamiento de archivos XML, se necesita que esos tengan la siguiente estructura:

a. Archivo de usuarios

```
<?xml version="1.0" encoding="UTF-8"?>
<usuarios>
    <usuario id="UserID"
password="UserPassword">
        <nombre>Nombre del Usuario</nombre>
        <edad>Edad del Usuario</edad>
        <email>Email del Usuario</email>
        <telefono>Teléfono del
Usuario</telefono>
    </usuario>
    <!-- Más usuarios pueden ser añadidos
aquí -->
</usuarios>
```

Figura 1. Estructura de archivo XML de usuarios.

Fuente: Elaboración propia, 2024

b. Archivo de productos

```
<?xml version="1.0" encoding="UTF-8"?>
<productos>
    <producto id="ProductID">
        <nombre>Nombre del Producto</nombre>
        <precio>Precio del Producto</precio>
        <descripcion>Descripción del
Producto</descripcion>
        <categoria>Categoría del
Producto</categoria>
        <cantidad>Cantidad del
Producto</cantidad>
        <imagen>Ruta de la Imagen del
Producto</imagen>
```

Figura 2. Estructura de archivo XML de productos.

Fuente: Elaboración propia, 2024

c. Archivo de empleados

```
<?xml version="1.0" encoding="UTF-8"?>
<empleados>
    <empleado codigo="EmployeeCode">
        <nombre>Nombre del Empleado</nombre>
        <puesto>Puesto del Empleado</puesto>
    </empleado>
    <!-- Más empleados pueden ser añadidos
aquí -->
</empleados>
```

Figura 3. Estructura de archivo XML de empleados.

Fuente: Elaboración propia, 2024

d. Archivo de actividades

```
<?xml version="1.0" encoding="UTF-8"?>
<actividades>
    <actividad id="ActivityID">
        <nombre>Nombre de la
Actividad</nombre>
        <descripcion>Descripción de la
Actividad</descripcion>
        <empleado>EmpleadoID</empleado>
        <dia hora="Hora">Día</dia>
    </actividad>
    <!-- Más actividades pueden ser añadidas
aquí -->
</actividades>
```

Figura 4. Estructura de archivo XML de actividades.

Fuente: Elaboración propia, 2024

Conclusiones

La integración efectiva entre el frontend (Django) y el backend (Flask) ha permitido la creación de una aplicación web robusta y eficiente. Esta combinación ha asegurado una experiencia de usuario fluida y una gestión eficaz de los datos, logrando una coordinación que facilita el desarrollo y la escalabilidad del sistema.

El proyecto ha implementado con éxito una serie de funcionalidades clave, como la autenticación y autorización de usuarios, la carga masiva de datos mediante archivos XML, y la generación y visualización de reportes detallados. Estas características han mejorado significativamente la eficiencia operativa y han proporcionado herramientas valiosas para la administración y toma de decisiones informadas.

Además, el enfoque en la usabilidad y la mejora continua de la interfaz de usuario ha resultado en una aplicación intuitiva y accesible, facilitando la navegación y las operaciones para todos los usuarios, desde administradores hasta clientes finales.

Referencias bibliográficas

Facultad de Ingeniería, USAC (2024). *Proyecto.* Obtenido de: https://uedi.ingenieria.usac.edu.gt/campus/pluginfile.php/797532/mod_resource/content/1/%5BIPC2%5DProyecto2_VJ2024.pdf

Anexos

Diagrama de clases

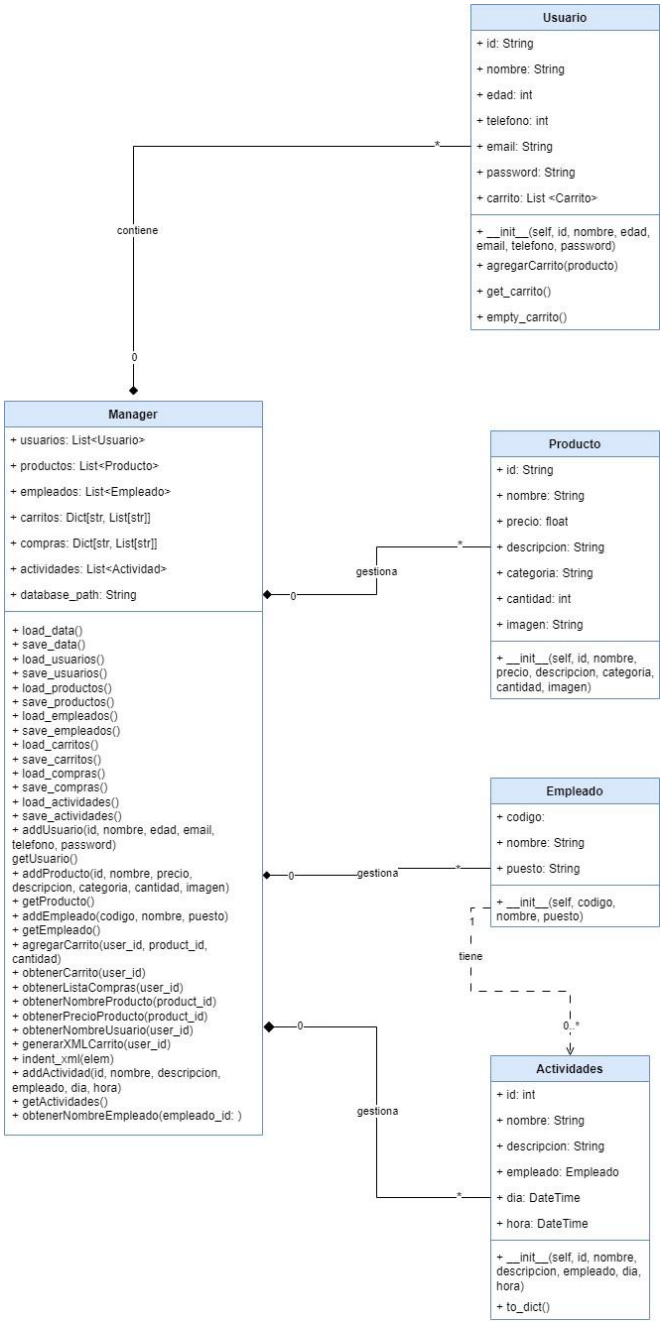

Figura 5. Diagrama de Clases

Fuente: Elaboración propia, 2024

Diagrama de casos de uso

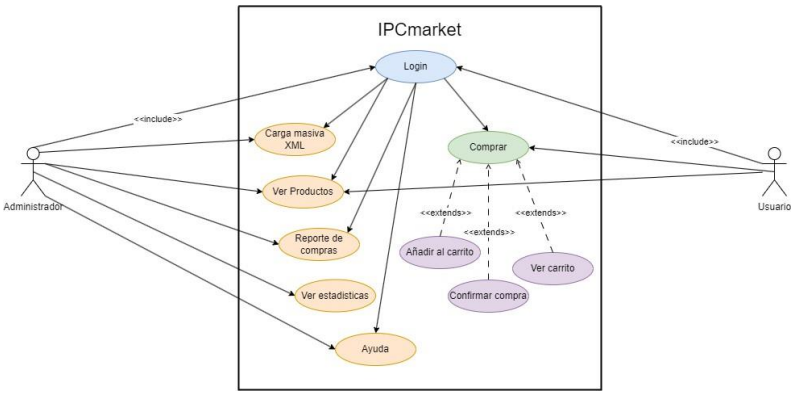

Figura 6. Diagrama de Casos de Uso

Fuente: Elaboración propia, 2024

Frontend

Figura 7. Home

Fuente: Elaboración propia, 2024

Figura 8. Inicio de sesión

Fuente: Elaboración propia, 2024

Figura 9. Carga y Vista Previa de archivos XML

Fuente: Elaboración propia, 2024

Figura 10. Ver productos

Fuente: Elaboración propia, 2024

Figura 11. Estadísticas

Fuente: Elaboración propia, 2024

Figura 12. Reporte de actividades

Fuente: Elaboración propia, 2024

Figura 13. Sección de ayuda

Fuente: Elaboración propia, 2024

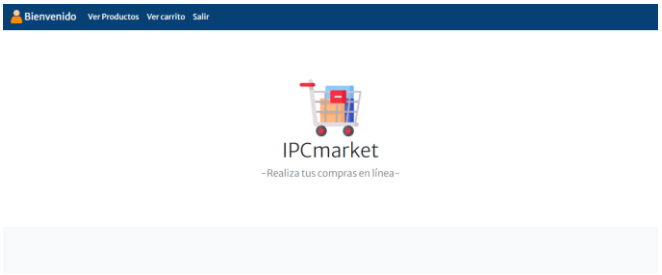

Figura 14. Home Cliente

Fuente: Elaboración propia, 2024

Figura 15. Ver y comprar productos

Fuente: Elaboración propia, 2024

Figura 16. Ver carrito

Fuente: Elaboración propia, 2024

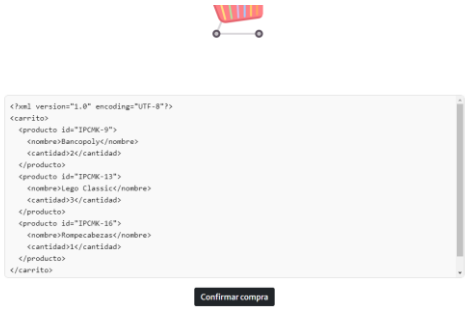

Figura 17. Confirmar compra

Fuente: Elaboración propia, 2024

Figura 18. Reporte de compras

Fuente: Elaboración propia, 2024