Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, Vacaciones 2021.

CONECCION CLIENTE-SERVIDOR

201801106 – Luis David Juarez Reyes

Resumen

El ensayo presenta el desarrollo de una API que brinde servicios utilizando el Protocolo HTTP como medio de comunicación para la transferencia de información entre los servicios, también el desarrollo de una aplicación web que pueda consumir esos servicios proveídos por la API. Para la creación de dicha API se hizo uso del framework Flask debido a que esta permite crear los servicios de una forma sencilla y rápida, para el desarrollo de la aplicación web que consumirá los servicios se uso el framework Django ya que este provee un patrón de diseño conocido como modelo vista-controlador y hace sencilla la construcción de la interfaz gráfica. La solución e implementación de lo antes mencionado fue a través del uso de un lenguaje de programación (Python) para el manejo de la lógica y así crear los distintos algoritmos y estructuras requeridos para el funcionamiento, también se utilizó para la construcción de la interfaz gráfica que permite interactuar de una forma más agradable e intuitiva con la aplicación

Palabras clave

Python, Django, Flask, Lenguaje de Marcado Extensible (XML), Programación Orientada a Objetos (POO).

Abstract

The essay presents the development of an API that provides services using the HTTP Protocol as a means of communication for the transfer of information between services, as well as the development of a web application that can consume those services provided by the API. For the creation of said API, the Flask framework was used because it allows to create the services in a simple and fast way, for the development of the web application that will consume the services, the Django framework was used since it provides a pattern of design known as model-view-controller and makes the construction of the graphical interface easy. The solution and implementation of the aforementioned was through the use of a programming language (Python) to handle the logic and thus create the different algorithms and structures required for operation, it was also used for the construction of the graphical interface that allows you to interact in a more pleasant and intuitive way with the application.

Keywords

Python, Django, Flask, Extensible Markup Language (XML), Objetct Oriented Programming (OOP).

Introducción

Se realizo el desarrollo de una API que brinda servicios utilizando el Protocolo HTTP como medio de comunicación para la transferencia de información entre los servicios, también el desarrollo de una aplicación web que pueda consumir esos servicios proveídos por la API. Para la creación de dicha API se hizo uso del framework Flask debido a que esta permite crear los servicios de una forma sencilla y rápida, para el desarrollo de la aplicación web que consumirá los servicios se usó el framework Django ya que este provee un patrón de diseño conocido como modelo-vista-controlador y hace sencilla la construcción de la interfaz gráfica. La solución e implementación de lo antes mencionado fue a través del uso de un lenguaje de programación (Python) para el manejo de la lógica y así crear los distintos algoritmos y estructuras requeridos para el funcionamiento, también se utilizó para la construcción de la interfaz gráfica que permite interactuar de una forma más agradable e intuitiva con la aplicación.

Desarrollo del tema

La API permite realizar el trabajo de procesamiento de datos, esta se encarga de recibir la información procesarla y de devolver un resultado que esta a su vez se muestra de forma gráfica. Se ha optado el uso del lenguaje de programación Phyton debido a que es multiplataforma y por su facilidad de uso y también porque cuenta con frameworks como Flask y Django para la realización de aplicaciones web que para este proyecto es necesario ya que nos facilitaran el trabajo para crearlo, además también porque soporta varios paradigmas de programación. Para la elaboración de la solución se provee un archivo con extensión y estructura CSV, el cual provee información que luego se procesarán y obtendrán datos para diferentes procesos, este archivo tiene la estructura de la siguiente forma:

```
nombre,apellido,edad,fechaCumpleannos,fechaPrimeraCompra
Alejandro,Herrera,28,13/05/1993,25/03/2021
Emilio,Vega,29,28/09/1991,20/07/2020
Andres,Rodriguez,31,14/11/1989,22/08/2020
Silvia,Calderon,29,12/05/1992,25/10/2020
Lisbeth,Tejeda,30,25/02/1991,14/06/2021
```

Figura 1. Estructura del archivo SCV.

Fuente: elaboración propia.

Este archivo se procesa y se obtienen los datos, debido a que los archivos pueden contener errores se vio la necesidad de hacer un autómata para verificar los posibles errores, se utilizaron expresiones regulares para verificar los datos ya que estos datos tienen una expresión común que puede ser reconocida y a través de ello verificarlos, a continuación, se muestran las expresiones regulares utilizadas:

```
Expresion regular

nombre y apellido : [A-Za-z ]

dato numerico : [0-9]

fecha: ^([0-2][0-9]|3[0-1])(\/)(0[1-9]|1[0-2])\2(\d{4})$
```

Figura I1. Expresiones regulares.

Fuente: elaboración propia.

Si el archivo CSV presenta un error, este no es transformado a un archivo XML si no que se muestra

al usuario un listado con los posibles errores encontrados e invitando al usuario para que sean corregidos.

Caso contrario, los archivos CSV no posean ningún error, crea un archivo XML con la siguiente estructura:

```xml
<Chet>
    <clientes>
        <nombre>Alejandro</nombre>
        <apellido>Herrera</apellido>
        <edad>28</edad>
        <fechaCumpleaños>13/05/1993</fechaCumpleaños>
        <fechaPrimeraCompra>25/03/2021</fechaPrimeraCompra>
    </clientes>
    <mejoresClientes>
        <nombre>Andrés López</nombre>
        <fechaUltimaCompra>28/04/2021</fechaUltimaCompra>
        <cantidadComprada>4</cantidadComprada>
        <cantidadGastada>2240</cantidadGastada>
    </mejoresClientes>
    <juegosMasVendidos>
        <nombre>The Legend of Zelda: Breath of the Wild</nombre>
        <fechaUltimaCompra>20/06/2021</fechaUltimaCompra>
        <copiasVendidas>325</copiasVendidas>
        <stock>27</stock>
    </juegosMasVendidos>
    <juegos>
        <nombre>Super Smash Bros. Ultimate</nombre>
        <plataforma>Nintendo Switch</plataforma>
        <añoLanzamiento>2018</añoLanzamiento>
        <clasificacion>E</clasificacion>
        <stock>20</stock>
    </juegos>
</Chet>
```

Figura II1. Expresiones regulares.

Fuente: elaboración propia.

Luego de ser almacenarlo, es mostrado en pantalla. El cliente es capaz de poder realizar modificaciones en él a través de un editor un text-area(editor de texto) previo a su envío al servidor y guardar las modificaciones. También es importante considerar que el cliente debe de tener la capacidad de agregar o eliminar la cantidad de etiquetas que considere necesarias.

Luego de ser mostrada la información con estructura XML se manda la información al servidor, para ser almacenada y procesada.

En el servidor se crea un método para interpretar el archivo XML mandado por Django, interpretado el archivo se obtiene la información de las etiquetas “Mejores Clientes”,” Juegos más vendidos”, “Clasificación de juegos”, “Cliente” y “Juegos”.

Luego de haber obtenido los datos el servidor los devuelve nuevamente a Django para poder visualizar una grafica de barras de los clientes que mas han comprado , una grafica en forma de pie que muestra los juegos mas vendidos y su año de lanzamiento , una grafica que muestra la cantidad de juegos que existen por clasificación, un listado de la fecha de cumpleaños de clientes ordenada por mes y se muestra un listado de los juegos que la empresa ofrece si su stock es mejor a 10 el nombre del juego ira de color rojo.

Obtenidos los datos en el servidor se necesitaba de una estructura en la cual se guardarían la información procesada, por lo cual se llevó al análisis del uso de un tipo de dato abstracto ya que permite el uso de memoria dinámica ya que el programa lo requiere debido a que no se sabe con certeza la cantidad de datos que puede contener.

Con el paradigma de programación orientada a objetos se emulo los diferentes tipos de datos para almacenarlos en la estructura de datos correspondientes, esto abrió la posibilidad de trabajar más eficientemente y optimización de memoria del programa ya que se tienen varios datos y se necesitan agruparlos entre ellos para su manipulación.

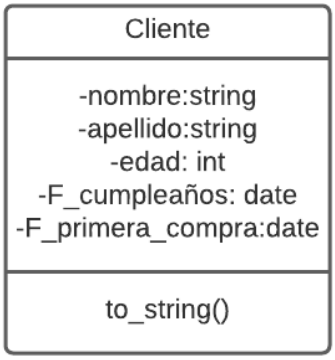

Figura IV. Modelo tipo dato Cliente.

Fuente: elaboración propia.

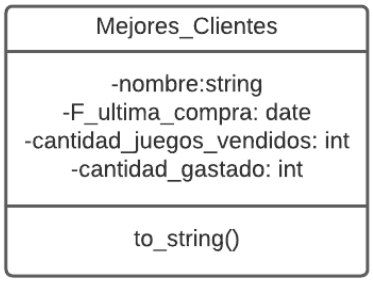

Figura V. Modelo tipo dato Mejores Cliente.

Fuente: elaboración propia.

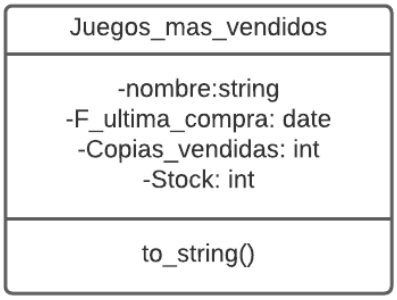

Figura VI. Modelo tipo dato Juegos más Vendidos.

Fuente: elaboración propia.

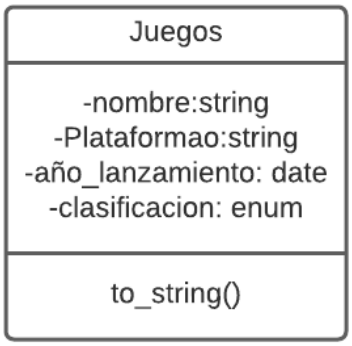

Figura VII. Modelo tipo dato Juegos más Vendidos.

Fuente: elaboración propia.

Para el funcionamiento y utilización de la API se crearon los diferentes end-points:

a. '/datos', este utiliza el método POST ya que se recibe el archivo cargado desde el front-end el cual será procesado para luego guardar un documento XML.

b. '/datos', este utiliza el método GET este cargar el archivo XML generado anterior mente para luego mandarlo al fonrt-end para que sea visualizado por el usuario.

c. '/graficas' este utiliza un método GET , abre el documento XML guardado anteriormente para luego interpretarlo y procesar la información necesaria , manda la información al front-end para la creación de graficas.

Conclusiones

Conocer e implementar bien las estructuras de datos que permiten manejar de forma muy rápida y eficiente los datos almacenados.

El desarrollo de esta aplicación da un ejemplo claro de cómo es que trabajan a nivel interno las páginas de conversión de documentos.

Se debe conocer a profundidad el lenguaje con el que se trabajara la solución ya que de estos dependen cuan accesible puede ser construir los algoritmos que determinaran la solución al problema, también es necesario conocer que paradigmas se pueden utilizar y en base a esto apegarse a una o varias para hacer lo menos tedioso posible el desarrollo de este.

Apéndice

Figura VIII. Diagrama general del flujo de la aplicación.
Fuente: Elaboración propia