SAT – PROYECTO 3

201602659 – María Cecilia Cotzajay López

Resumen

Se realizó un software de aplicación web basado en la arquitectura de cliente-servidor para una empresa dedicada a la compra y venta de videojuegos, utilizando al framework Django como cliente, esto incluye el uso de archivos como por ejemplo [.html] y [.css] que fueron los que se usaron para este proyecto. En el lado del cliente, se manejan vistas, url´s y archivos estáticos, plantillas definidas por mi persona para la parte gráfica del software.

En el lado del servidor se utilizó Flask para recibir las peticiones de parte del cliente y así mismo el procesamiento de datos para posteriormente enviar respuestas al mismo (API-REST).

Se utilizó el Protocolo HTTP para que fuera posible la comunicación entre ambos servidores y para la simulación de bases de datos se utilizaron archivos con extensión [.xml]

Palabras clave

i). **API-REST**: es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones y/o contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita del consumidor (la llamada) y el que requiere el productor (la respuesta).

ii). **Servidor**: es un ordenador y sus programas, que están al servicio de otros ordenadores que atiende y responde a las peticiones que le hacen los otros ordenadores

iii) **Framework:** es una plataforma de software universal y reutilizable para desarrollar aplicaciones de software, productos y soluciones. Es una especie de biblioteca, para crear aplicaciones web.

iv) **Vistas**: es una función en Python que hace una solicitud Web y devuelve una respuesta Web, esta respuesta puede ser el contenido de una página, un error, un documento [.xml], etc.

iv) **HTTP**: Hypertext Transfer Protocol, permite realizar una petición de datos y recursos, como pueden ser documentos [.html].

Abstract

A web application software based on the client-server architecture was made for a company dedicated to the purchase and sale of video games, using the Django framework as a client, this includes the use of files such as [.html] and [. css] which were the ones used for this project. On the client side, views, urls and static files are handled, templates defined by me for the graphical part of the software.

On the server side, Flask was used to receive requests from the client and also process data to later send responses to it (API-REST).

The HTTP Protocol was used to allow communication between both servers and for the simulation of databases, files with the [.xml] extension were used.

Keywords

i). API-REST*: it is a set of definitions and protocols that is used to develop and integrate the application software and / or contract between the information provider and the user, where the content that is needed from the consumer (the call) is established and the one required by the producer (the answer).*

*i**i). Server:** it is a computer and its programs, which are at the service of other computers that attend to and respond to the requests made by other computers*

***iii) Framework:** it is a universal and reusable software platform for developing software applications, products and solutions. It is a kind of library, to create web applications.*

***iv) Views:** it is a function in Python that makes a Web request and returns a Web response, this response can be the content of a page, an error, a document [.xml], etc.*

***iv) HTTP:** Hypertext Transfer Protocol, allows a request for data and resources, such as documents [.html].*

Introducción

Es muy importante aprender sobre el tema de los las arquitecturas de software como su implementación en los diferentes proyectos. Cabe mencionar que la cualidad más importante sobre las arquitecturas de software es que son la estructuración básica de un proyecto, es decir que sin esa parte tan fundamental de un proyecto el mismo fracasaría al no tener claro el tipo de sistema que se le implementará, el orden y lo más importante lo que el cliente (La persona que compra el producto o software) realmente necesita. Otra cualidad de las muchas que tiene es que el proyecto conlleva un tiempo mínimo en su realización e implementación, así como la reducción de gastos ya que una arquitectura de software es directa, exacta.

El proyecto está basado en el uso, implementación y manejo de la arquitectura Cliente-Servidor , para tener un conocimiento profundo en el funcionamiento

o manejo del mismo y de la implementación de una API-REST para la comunicación entre el cliente y el servidor.

En el siguiente apartado se describe lo que conlleva el proyecto, es decir cada parte del mismo.

Desarrollo del tema

El presente proyecto es una aplicación web basada en la arquitectura Cliente-Servidor para una empresa dedicada a la compra y venta de video juegos. Antes de empezar con la explicación del desarrollo del proyecto es necesario saber el siguiente concepto.

"La arquitectura cliente-servidor es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes".

Se implementó como cliente el framework Django, se hicieron configuraciones iniciales cuando se creó el proyecto en settings. Py tales como una nueva ruta del proyecto, importaciones de la librería os, unitpath, el Language_code, zona horaria y la dirección de la carpeta static que es donde van todos los archivos estáticos tales como plantillas HTML, CSS, imágenes y también es posible agregarles JavaScript. En este archivo py van todas las configuraciones necesarias para el funcionamiento del proyecto, también se editaron las partes de urls, es decir que en la parte de urls del proyecto, se pasó el control de valga la redundancia las urls a un archivo con el mismo nombre en la carpeta de la aplicación el cual se van agregando manualmente las url´s.

Después de todo esto se creó una aplicación con el nombre Servicio1_django en el cual van todas las vistas, urls. En las vistas se crearon una para index con html llamado index y url también con el mismo nombre que es la página inicial de la App con el título del curso y una barra de menú el cual contiene archivo con subelementos llamados cargar de archivo que es el que redirecciona a la página principal, el subelemento salirApp que redirecciona a la página de inicio. El siguiente elemento es el de Reportes, el cual tiene subelementos que muestra gráficas Resumen de IVA por fecha y NIT, Resumen por rango de fechas, Consultar Datos y Reporte en PDF.
El último elemento es el de Ayuda con subelementos documentación que es la página que muestra el presente documento del proyecto, el subelemento ayuda que muestra la información de la persona que creo la aplicación que soy yo su servidora.

La siguiente página que es la principal es la que selecciona un archivo XML, los abre y hace su respectiva lectura con la librería csv que tiene python luego de esto se hicieron validaciones de los datos con formatos establecidos mediante expresiones regulares por medio de la librería re (regex) que es la que trae python. Después se crea un documento XML si los datos vienen en orden de lo contrario deja de leer el archivo y muestra un mensaje de error en un textArea. Cuando ya se tiene el documento XML se muestra en un textArea para ser modificado por el usuario si así lo desea y cuando ya esté listo este será enviado por medio de un botón al Servidor 2 que es Flask y a través de los protocolos HTTP en este caso una petición post.

Cuando el archivo ya está disponible en el segundo servidor, este leerá el documento y creará sus respectivos objetos que se almacenaran en listas de python, se harán cálculos como sumas de compras y el resto de operaciones. Cuando finalice todo este procesamiento de datos se crea un nuevo documento con extensión XML que contendrá toda la información necesaria para hacer los reportes que más adelante se describen. Por último el servidor

Flask retornara ese documento a Django por medio de la petición get y de nuevo lo mostrara en un segundo textArea en la página principal, este textArea no puede modificarse, únicamente podrá ser visualizado el contenido del archivo.

Por último se puede visualizar una serie de reportes de la información que se obtuvo mediante el procesamiento de datos. Se muestra el reportes Resumen de IVA por fecha y NIT a través de una gráfica de barras, Resumen por rango de fechas a través de una gráfica de pie o de pastel, reporte de la cantidad de juegos que existen por clasificación por medio de una gráfica de barras, SAT es una empresa dedicada a sus clientes por lo que también tiene un reporte que muestra las fechas de cumpleaños de los mismos para poder darle bonificaciones o descuentos y por último se tiene el reporte de listas de juegos, el cual se muestran en un tabla y con dos colores según su stock. Si su stock esta en rojo significa que posee menos de 10 productos por juego.

Algunas del resto de vistas son principal, acercaDe, documentación, recibirXML, enviarXML, traerXML, segundoXML.

Es muy interesante la comparación de ambos servidores como por ejemplo en la creación, en la implementación, configuraciones. Flask es más sencillo de utilizar y reúne solo los componentes necesarios al ser un microframework. Django es un framework que es completo en cuestión de componentes y por lo tanto es más grande en espacio y en cantidad de archivos.

En este proyecto únicamente se utilizaron peticiones HTTP get (leer) y post (crear) de las cuatro más populares y que son parte del CRUD (crear, leer, actualizar y eliminar).

Conclusiones

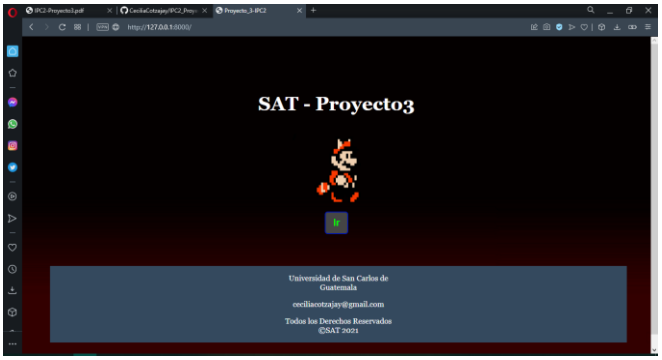
La arquitectura de software es fundamental en la elaboración de un proyecto, de lo contrario es un fracaso total, tanto en la elaboración como en lo que necesita el cliente (Persona que compra el software.)

La arquitectura Cliente-Servidor es ideal para el desarrollo de aplicaciones web y en el caso de utilizar bases de datos, entonces se usaría también una arquitectura de capas.

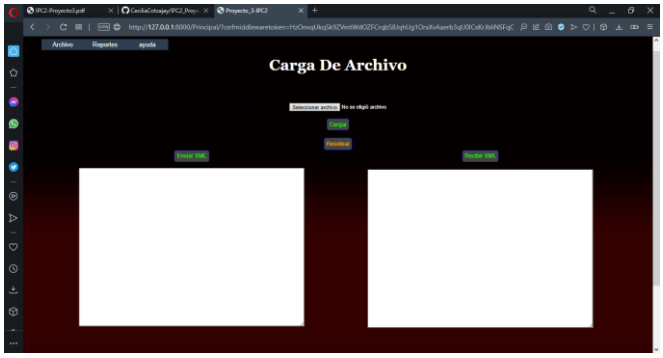
El procedimiento de la carga de Archivos y la manipulación de datos y almacenamiento de información en archivos puede ser una opción sustituyente de las bases de datos.

Es necesaria una API-REST para la comunicación del servidor del cliente y del servidor backend utilizando los protocolos HTTP y se debe tener conocimiento previo sobre ello para poder implementar este tipo de arquitectura

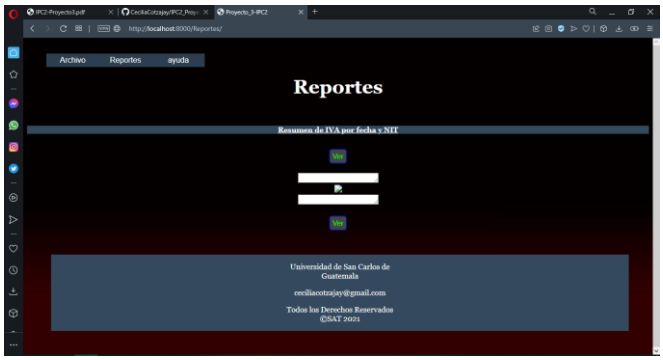
A continuación se muestran algunas ventanas de la aplicación, las más básicas



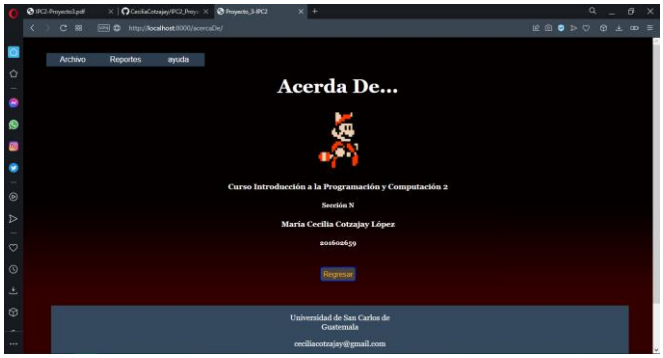
(Página de inicio, Proyecto SAT, Cecilia Cotzajay)



(Página principal, Proyecto SAT, Cecilia Cotzajay)



(Página de reportes, Proyecto SAT, Cecilia Cotzajay)



(Página Acerca de, Proyecto SAT, Cecilia Cotzajay)

Referencias bibliográficas

1) C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.

2) https://blog.hostdime.com.co/que-es-un-framework-informatica-programacion/
(Para consulta de algunas palabras clave.)

3) http://www.maestrosdelweb.com/curso-django-las-vistas/#:~:text=Un%20funci%C3%B3n%20de%20vista%20o,XML%2C%20entre%20muchas%20cosas%20m%C3%A1s.
(Para consulta de algunas palabras clave.)

4) Arquitectura de software (voigtmann.de)
(Para consulta de algunas palabras clave.)