PROYECTO 3

202103718 – HARRY AARON GÓMEZ SANIC

Resumen

Software web desarrollado utilizando distintas tecnologías para las distintas partes que conforman un api, en el caso del apartado denominado "Front-End" se utilizó HTML y css, los cuales son lenguajes de metadatos, utilizados para darle una presentación visual al proyecto, en esta presentación el cliente interactuara con los distintos componentes. Además, se utilizó un framework web de Python llamado Django, el cual utiliza el patrón "Modelo-Vista-Template" (MVT), con este framework se logra capturar los datos de la parte gráfica y además le proporciona a esta una mayor versatilidad, puesto que se puede reutilizar código HTML y darle mayor fluides al flujo de vistas.

Por otro lado, en el apartado denominado "Back-End", se utilizó una librería de Python llamada flask, dentro del Back-End se realiza toda la lógica que ira sobre el Front-End, así como también se da cierta interacción entre ambas partes.

Con estas dos partes se desarrolló una página web capaz de leer archivos xml, almacenarlos en una base de datos y que la presentación visual pudiera interactuar con estos datos, siempre con la ayuda de la lógica que se realizó en el lado del Back-End, utilizando para esta comunicación entre partes los lenguajes de metadatos, JSON y XML.

Palabras clave

Front-End
Back-End
Framework

Abstract

Web software developed using many technologies for the different parts that makes up an api, in the case of the headland called "Front-End" we used HTML and css, wich are metadata languajes, used to give the project a visual presentation, in this presentation the client is going to interact with the different components in it. Also, a web framework called Django were used, this framework uses de "Model-View-Template" patron (MTV), with this framework it is posible to capture the data from the visual part and also it gives more versatility, due to the re-use of the HTML code and to a greater fluency to the Flow of views.

In other hand, in the headland called "Back-End", a library from Python called Flask were used, within the Back-End all the logic that will go over the Front-End is done, as well as some interaction between both parts.

With these two parts we developed a web page capable of reading xml files, storing them in a database and that the visual presentation could interact with this data, always with the help of the logic that was done on the Back-End side, using for this communication between parts the metadata languages, JSON and XML.

Keywords

Front-End

Back-End

Framework

Introducción

En este proyecto se requería el desarrollo de un api de atención al cliente en relación con el alquiler de distintas máquinas y componentes virtuales, en la cual se debían manejar clientes, recursos y las distintas categorías de los componentes que se están manejando.
Para la configuración de dichos componentes, de los distintos clientes registrados en la base de datos y las instancias a manejar durante el proceso de facturación de cada uno de los clientes se utilizó el manejo de archivos xml tanto para la configuración como para los datos de consumo.
Para la parte grafica con la cual el usuario interactúa se utilizaron las tecnologías html para el esqueleto de la presentación visual, css para los estilos del esqueleto, Bootstrap para manejar distintos ítems como los acordeones y botones, Django para manejar las distintas vistas y controlar los datos que se ingresaran en el front-end, además también para usar distintas características de este framework como los forms, la reutilización de código de metadata.

Para que la presentación visual trabajara de forma correcta se necesita darle una lógica que la sostenga, esta lógica se albergó en el apartado de back-end, esta lógica se manejó en la librería llamada flask, la cual se conectó al front-end para mantener la corriente de datos e una parte a otra, esta lógica incluye la creación de una base de datos, el manejo de datos desde esta base de datos hacia el front-end, la creación de nuevos datos desde programas como postman y desde el propio front-end.

Desarrollo del tema

Como ya se mencionó en la introducción al proyecto,se requeria una aplicación web que se desarrollara especialmente en Python con sus distintos frameworks y librerías, esta aplicación web tenia que ser una aplicación de procesamiento de datos de distintos clientes, recursos, categorías, etc. Dado que era especialmente una de alquiler de equipos virtuales.

Para la configuración inicial de esta aplicación era necesario leer un archivo xml, para este cometido era necesario utilizar las herramientas de html y django, en el caso de html, en su atributo input se contiene una opción llamada “file”, la cual abre una ventana emergente de selección de archivos con la cual se podía recibir en los archivos de django una copia de este archivo por medio de una función llamada object handle, la cual permite manejar un archivo en este tipo de atributos de html. Después de tener una copia del archivo que el usuario elige, se procede a leer el archivo xml por medio del element tree, ya que con este se puede acceder a los distintos atributos del xml, estos atributos se van guardando en distintas clases que se tiene que crear para después poder mandar los objetos al back-end por medio de un json. Estas clases se listan a continuación

- Cliente
- Consumo
- Recursos
- Categoría
- Configuración
- Recursoconfig
- Instancia

Cada una de estas clases almacena distintas listas de los datos que se leen en el xml, y luego de tener los objetos y listas de cada uno de estos puntos se envían a la parte del back-end donde se procesaran.
Terminando con el front-end es importante mencionar que cada una de las vistas donde el usuario interactúa esta separa en django, separada

por ruta, por función y por aplicación en algunos casos, además en cada una de estas vistas es posible el uso de formularios para recolectar datos para el back-end.

Entrando al terreno del back-end, para el procesamiento de los datos que django recogia, se utilizó la librería de Python llamada flask, flask es una librería versátil pero que fléquese en el apartado de renderización y manejo de templates, por lo que en esta ocasión se utilizó principalmente para el procesamiento de datos, las principales funciones que se desarrollaron aquí fueron los endpoints de creación de datos, los endpoints son rutas las cuales reciben cierto tipo de archivo según el programador requiera, en el caso del desarrollo del proyecto reciben archivos JSON con datos diferentes dependiente directamente del endpoint, ya que en el caso de la creación de un cliente este recibirá el nombre, identificación y demás credenciales que necesita un cliente en su respectiva cuenta, los endpoints utilizados en el desarrollo del proyecto fueron los siguientes:

- /cargaXML/configuración

La cual recibía los objetos mandados desde el front-end y los procesa para ingresarlos a un archivo JSON que genera automáticamente al leer un archivo xml de configuración.

- /cargaXML/consumo

Endpoint encargado de recibir los datos del mensaje de consumo que se carga desde el archivo xml, y como en el anterior caso, este endpoint también crea un archivo JSON como base de datos para los objetos.

- /cliente/crearCliente

Con este endpoint se reciben los datos de los formularios del front-end y se ingresan a la base de datos, también recibe json de programas como postman

- /cliente/crearInstancia

De igual forma que el anterior, este endpoint recibe datos de los forms del front y los ingresa a la base de datos, además se realizan distintas validaciones para no repetir instancias

- /cliente/cancelarInstancia

Este endpoint recibe los datos de un cliente y una instancia y cambia su estado a una instancia cancelada

- /categoría/crearCategoria

Endpoint de creación de categoría que recibe datos de los formularios y los ingresa a la base de datos, cuenta con sus propias validaciones para no repetir categorías.

- /categoría/crearConfiguracion

En este endpoint se reciben datos del front-end y se validan que las credenciales que se mandan existan en la base de datos en alguna categoría, para posteriormente añadirla.

- /recurso/crearRecurso

De igual manera que los anteriores endpoints este recibe objetos los traduce a un archivo json y los ingresa al archivo correspondiente.

- /data/consultarDatos

Este endpoint lee los archivos que funcionan como base de datos y por medio de una petición de tipo get retorna al front-end todos los atributos que se han guardado en la base de datos, una vez en el front-end estos atributos se imprimen en el html por medio de un archivo json.

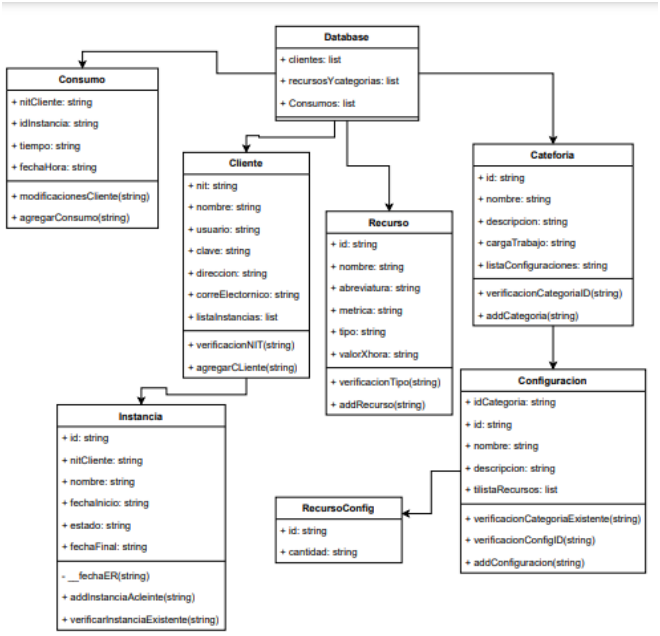

Figura 1. Diagrama de clases de proyecto 3 IPC2N

Fuente: elaboración propia.

Conclusiones

El desarrollo de aplicaciones web es un campo infinidad de herramientas que se pueden usar a nuestro beneficio, además es importante saber todos los componentes que conforman un api, ya con este conocimiento podemos elegir la mejor herramienta de trabajo.

En el caso de Python cuenta con múltiples herramientas para el desarrollo web, pero dos de sus herramientas más conocidas para este fin son django y flask, aunque flask palidece ante la versatilidad de django, no quiere decir que no sea una librería que se deba usar especialmente para el tratamiento de datos de un api, por otro lado, django puede ser utilizado tanto para la parte de front-end como para la parte de back-end.

Otro elemento importante del desarrollo web es la conexión entre ambas partes la cual se realiza con lenguajes de metadatos, entre los cuales están los que se utilizaron para la realización de este proyecto, los cuales son los archivos JSON y los XML.

Aunque el uso de los archivos XML se ha visto opacado en los últimos años por los archivos JSON aún son una buena opción para ser utilizados.

Referencias bibliográficas

Luis Joyanes Aguilar, (2008), Fundamentos de programación, algoritmos, estructuras de datos y objetos, McGraw-Hill

David Budgen, (2003), Software Design, Second Edition, Pearson Addison Wesley

Craig Larman (2001), Introduccion al analisis y diseño orientado a objetos, Prentice Hall

Extensión:

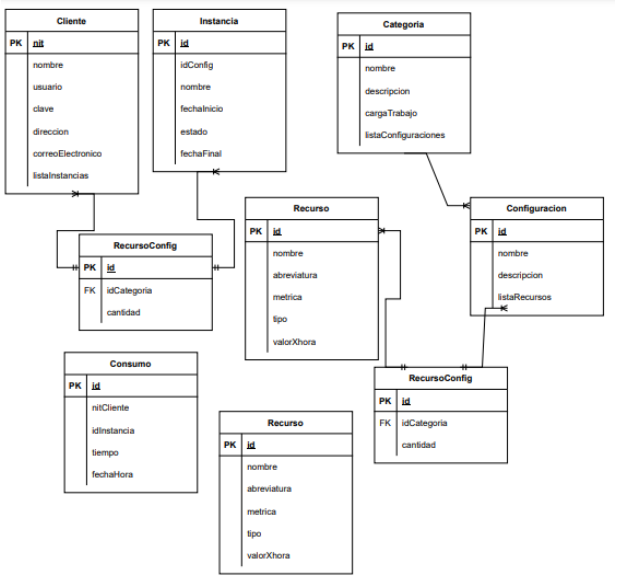

Figura 1. Modelo Relacional IPC2N