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 2022.

PROYECTO 3

202001144 – Edgar Rolando Alvarez Rodriguez

Resumen

Se presento un problema el cual consiste en la lectura de un archivo XML por medio de una página html para el manejo de entradas y salidas para mensajes, las empresas con sus propios alias y los mensajes en listas distintas para su manejo en palabras y sentimientos, dichos datos se encuentran dentro del archivo XML que deberá ser procesado con código en lenguaje Python.

Palabras clave
Código, Python, Listas

Abstract

A problem is presented which consists of reading an XML file through an html page for handling input and output for messages, companies with their own aliases and messages in different lists for handling in words and feelings, Said data is found within the XML file that must be processed with code in Python language.

Keywords
Code, Python, Lists

Introducción

En el siguiente ensayo se darán a conocer las funcionalidades creadas para la lectura y manejo de los datos dentro del archivo XML para crear las salidas y a su vez crear un archivo para la salida de los datos.

Se dan a conocer las funciones más importantes para el manejo de los datos, así como la estructura principal del código.

Desarrollo del tema

Como punto inicial se presenta el problema a resolver, el cual es el siguiente:

La empresa Tecnologías Chapinas, S.A. ha creado una estrategia para establecer si un mensaje tiene un sentimiento positivo, negativo o neutro a través de la creación de un diccionario de datos que determine palabras que puedan calificar un mensaje como positivo o negativo

El programa por desarrollar, luego de recibir el mensaje antes mencionado, deberá almacenar la información necesaria en formato XML, para posteriormente emitir reportes y realizar consultas.

Para resolver dicho problema se utilizó el lenguaje Python, junto con la librería “Minidom” para leer el archivo XML, la librería “ElementTree” para la creación de un archivo que xml como base de datos y la librería “numeralite” , “re” para las búsquedas de diferentes datos dentro del xml y la librería “json” para envió y recibimiento de datos entre el backend y el frontend.

La estructuración del código se separó en 2 carpetas una para el backend y otra para el frontend. Cada una de estas contiene diferentes archivos que ayudan a un mejor orden del proyecto.

Se utilizo el framework Django para la creación de las carpetas y el orden de los archivos dentro de estas,

Se utilizo el framework Flask para instanciar un puerto para correr la página web.

Como parte del uso de flask para el manejo de las urls y para correr la pagina web se desecharon los archivos de Django encargados de correr la pagina web y se utilizo solo para el manejo de carpetas, mientras que con Flask se conectaron las urls con archivos javascript en el frontend para enviar datos entre el backend y el frontend.

En el caso de inclusión de figuras, deben ser nítidas, legibles en blanco y negro. Se denomina figuras a gráficas, esquemas, fotografías u otros elementos gráficos.

Los archivos utilizados para correr la página web son:

a. Main.py

b. analizarxml.py

c. consultarFechas.py

d. Gestor.py

e. Retornador.py

Main.py

Este archivo es el encargado de instanciar la pagina web y correrla, así mismo como conectar las urls con los archivos javascript en el frontend.

Al mismo tiempo también se conecta con el archivo llamado Gestor.py que se encarga de llamar a otros archivos de lectura y escritura para la base de datos.

Gestor.py

Este archivo se encarga de llamar a otros archivos según su utilidad. Se conecta al main.py y corre al momento de recibir una llamada del main luego que este allá recibido una llamada del frontend.

Gestor se encarga de recibir datos y enviarlos, como los datos del xml y llamar al analizadorxml.py, retornado.py, así mismo como enviar los datos de regreso al main y que este los retorne al frontend para mostrarlos en la página html.

Analizadorxml.py

Este archivo se encarga de leer el archivo de entrada al consultar datos y escribir un archivo “request.xml” que devuelve los datos requeridos en el proyecto y mandarlos de regreso a gestor para retornarlos al main y mostrarlos en pantalla por medio de la página web.

Retornador.py

El tetornador.py recibe datos como las listas de palabras, empresas y alias dentro del archivo entrada para luego guardarlas en un archivo database.xml y luego leer este mismo cuando se borre la base de datos y devolver estas listas cuando se necesite usarlas.

Conclusiones

Es posible conectar backend con frontend por medio de librerías como Flask.

Mediante el uso correcto de librerías, el manejo de archivos y distintos datos se vuelve más fácil, por ello se recomienda no saturar un archivo de librerías y usar un numero adecuado sin exagerar.

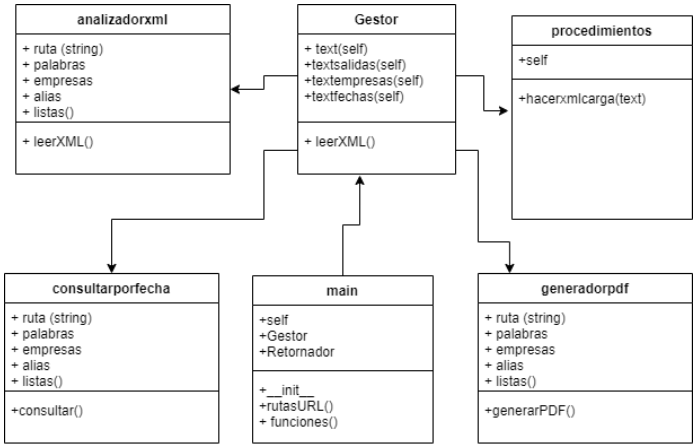
El uso de Django facilita el manejo y orden de archivos dentro de los proyectos, facilitando el trabajo del desarrollador y ayudando en el ahorro de tiempo del mismo.

Referencias bibliográficas

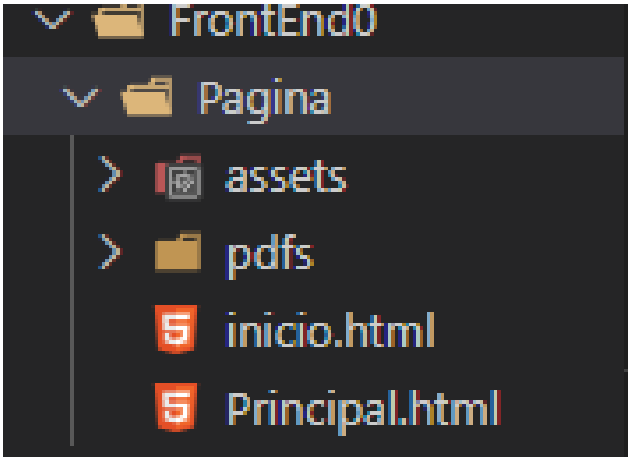
C. J. Date, (1991). An introduction to Database Systems. Addison-Wesley Publishing Company, Inc.

Node Shape (s.f.) Graphviz Recuperado 3 de marzo 2022, de https://graphviz.org/doc/info/shapes.html

Anexos:



Carpetas creadas por Django:



Instanciando la pagina web:

```python
#Crear la app
app = Flask(__name__)
app.config["DEBUG"] = True

CORS(app)

gestor =  Gestor()
retorn =  Retornador()

# EndPoints
@app.route('/',methods=['GET'])
def home():
    return 'Funciona esta Mierda UwU'
```

Gestor:

```python
class Gestor:
    def __init__(self):
        self.text = ''
        self.textSalida = ''
        self.listEmpresas = ''
        self.listFechas = ''

    #Create
    def recibirXML(self, data):
        self.text = data
        Procedimientos.hacerXML_carga(data)
        # print('Esta es la info del xml: '+self.text)


    def analizarXML(self):
        analizadorXML.lectorXML('BackEnd0\\xmlCopyEntrada.xml')

        file = 'BackEnd0\\request.xml'
```