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.

PROYECTO3: LA EMPRESA TECNOLOGÍAS CHAPINAS, S.A.

202012039 – Angel Francisco Sique Santos

Resumen

La empresa Tecnologías Chapinas, S.A. está desarrollando una herramienta que sea capaz de analizar contenido de redes sociales y establecer el sentimiento de los usuarios respecto a una empresa y los servicios que provee.

Para eso se nos solicita hacer uso de las herramientas Python, Flask y Django para lograrlo.

Palabras clave

Flask

Django

HTML

Programación orientada a objetos

Python

Abstract

The company Tecnologias Chapinas, S.A. is developing a tool that is capable of analyzing content from social networks and establishing the feeling of users regarding a company and the services it provides.

For that we are asked to use the Python, Flask and Django tools to achieve it.

Keywords

Flask

Django

Matrix

Object-oriented programming

Python

Introducción

El tema principal de este proyecto es el desarrollo web. Para el proyecto se nos solicitó usar Flask y Django, que son herramientas para desarrollo web, haciendo uso de Flask para el Backend y Django para el Frontend.

El tipo de archivos permitido para la lectura es XML, para leerlo se implementó ElementTree.

Desarrollo del tema

1. Backend

Las clases usadas fueron:

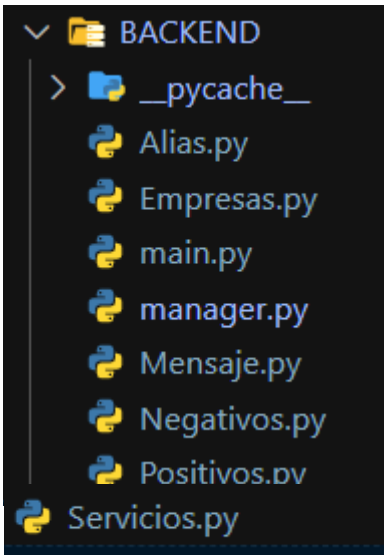

Figura 1. Clases.

Fuente: Elaboración propia

1.a. main.py:

Es donde definimos y llamamos las funciones que necesitamos, donde se une todo el Backend.

```python
from manager import Manager
from flask import Flask, jsonify, request
from flask.json import jsonify
from xml.etree import ElementTree as ET
import re

app = Flask(__name__)
manage = Manager()

@app.route('/')
def index():...

@app.route('/obtenerSalida', methods = ['GET'])
def obetenerSalida():      You, anteayer • Archivo de s

@app.route('/add', methods=['POST'])
def add():...

@app.route('/getXML')
def get_XML():

    return jsonify({'xml' : manage.xml}) , 200

@app.route('/getMensajes')
def get_mensajes():
    return jsonify(manage.obtener_Mensajes()), 200

@app.route('/getEmpresa')
def get_empresas():
    return jsonify(manage.obtener_Empresas()), 200
```

Figura 2. Parte de main.py.

Fuente: Elaboración propia

1.b. manager.py:

Es donde se hace la mayor parte de la lógica del Backend, aquí está el código de la mayoría de funciones que son llamadas en el main.py

```python
class Manager():
    def __init__(self):...

    def eliminarXML(self):...

      You, hace 6 días • Lectura de archivos …
    def agregar_Mensaje(self, l, f, h, u, r, m, d):

    def agregar_Positivo(self, p):...

    def agregar_Negativo(self, p):...

    def agregar_Empresa(self, p):...

    def obtener_Mensajes(self):...

    def obtener_Positivos(self):...

    def obtener_Negativos(self):...

    def obtener_Empresas(self):...



    def contarPalabras(self):...

    def buscarEnMensaje(self,mensaje):...

    def crearArchivoAlmacenamiento(self):...
```

Figura 3. Funciones del manager.py.

Fuente: Elaboración propia

1.c. Mensajes.py:

Este es un objeto que guarda todos los datos relacionados con el mensaje, la fecha de publicación, el mensaje, las empresas y servicios mencionados, lugar, hora y entre otras cosas.

```python
class Mensaje():
    def __init__(self, lugar, fecha, hora, usuario,red,mensaje,datos):
        self.lugar = lugar
        self.fecha = fecha
        self.hora = hora
        self.usuario = usuario
        self.red = red
        self.mensaje = mensaje
        self.datos = datos
```

Figura 4. Mensajes.py.

Fuente: Elaboración propia

1.d.Negativos.py y Positivos.py:

Estas clases son prácticamente iguales, son objetos con el único valor de “Palabra” este representa la palabra positiva o negativa obtenida en del XML.

```python
class Positivo():
    def __init__(self, palabra):
        self.palabra = palabra
```

Figura 5. Positivo.py.

Fuente: Elaboración propia

1.e. Empresas.py:

Este objeto almacena el nombre y una lista de Servicios atribuidos a cada empresa.

```python
from Servicios import Servicio
class Empresa():
    def __init__(self, nombre):
        self.nombre = nombre
        self.servicios = []

    def agregar_Servicio(self, n):
        nuevo = Servicio(n)
        self.servicios.append(nuevo)

    def obtener_Servicio(self):
        json = []

        for servicio in self.servicios:

            servicio = {
                'nombre' : servicio.nombre,
                'alias' : servicio.obtener_Alias()
            }
            json.append(servicio)

        return json
```

Figura6. Empresas.py

Fuente: Elaboración propia

1.f. Servicios.py:

Este objeto almacena el nombre del servicio y una lista de Alias atribuidos a cada servicio.

```python
from Alias import Alias
class Servicio():
    def __init__(self, nombre):
        self.nombre = nombre
        self.alias = []

    def agregar_Alias(self, n):
        nuevo = Alias(n)
        self.alias.append(nuevo)



    def obtener_Alias(self):
        json = []
        for tmpalias in self.alias:
            tmpalias = {
                'nombre' : tmpalias.nombre,
            }
            json.append(tmpalias)
        return json
```

Figura 7. Servicios.py

Fuente: Elaboración propia

1.g. Alias.py:

Este objeto solamente almacena el alias obtenido del XML.

```python
class Alias():
    def __init__(self, nombre):
        self.nombre = nombre
```

Figura 8. Alias.py

Fuente: Elaboración propia

2. Frontend

Las clases usadas fueron:

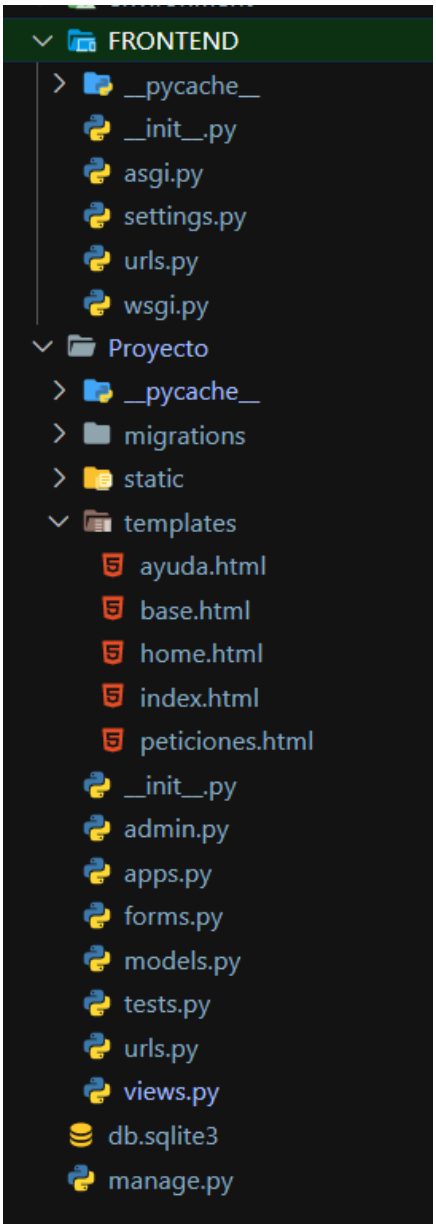

Figura 9. Clases.

Fuente: Elaboración propia

En el Frontend tenemos las clases donde más se trabaja que son view.py, que es donde tenemos nuestras request para conectar el HTML con el Backend, urls.py es donde definimos las urls de las páginas que usaremos, templates es una carpeta donde creamos todo el HTML de nuestro proyecto, fomrs.py es donde creamos formularios que sirven para la lectura de archivos y la carpeta static que es donde iría todo el JavaScript y CSS de nuestro proyecto.

Conclusiones

La utilización de Django facilita el desarrollo del Frontend a la hora de querer hacer y construir diferentes vistas las cuales pueden ser repetitivas, pero con Django se aceleran esos procesos, así como su fácil utilización al estar basado en Python.

Las aplicaciones web son el ahora de la gran mayoría del software que se desarrolla debido a la facilidad de uso remoto y acceso a datos. Si uno quiere estar en la jugada estos días, deberá estar siempre actualizado respecto a estos temas.

Referencias bibliográficas

djangoproject. (2022). *djangoproject*. Obtenido de djangoproject: https://docs.djangoproject.com/en/4.0/

getbootstrap. (2022). *getbootstrap*. Obtenido de getbootstrap: https://getbootstrap.com/

python.org. (6 de Marzo de 2022). *ElementTree*. Obtenido de python.org: https://docs.python.org/es/3/library/xml.etree.elementtree.html

w3schools. (s.f.). *w3schools*. Obtenido de w3schools: https://www.w3schools.com/css/default.asp

Apéndices

a. Diagrama de clases: