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 2021.

IPC2-Proyecto 3

201901758 – Erick Ivan Mayorga Rodríguez

Resumen

El presente proyecto muestra un software que es capaz de ser consumido mediante internet, el cual representa una API, el software principal recibirá una bitácora que contendrá un conjunto de elementos con ciertas características las cuales se podrán procesar y ser almacenadas de manera dinámica mediante un archivo de tipo XML.

Los archivos de entrada podrán contener distintos tipos de errores los cuales deberán ser ignorados, los archivos de salida se dirigirán a un software creado en Django en el cual se presentará de manera gráfica el resultado del procesamiento del archivo inicial.

El archivo inicial es capaz de ser recibido mediante un frontend diseñado en Django, el cual mediante la librería Http request realiza una petición en la cual envia los distintos tipos de información que se almacenen en este frontend y posterior a eso son procesados por elementos pertenecientes a la Api, dicha Api esta diseñada en Flask.

Palabras clave

- HTTP: protocolo de transferencia de hipertextos, se utiliza para realizar peticiones en páginas de internet.
- Django: Framework de desarrollo web de código abierto.
- Flask: framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código
- Api: interfaz de programación de aplicaciones. Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero.

.

Abstract

This project shows a software that is capable of being consumed through the internet, which represents an API, the main software will receive a blog that will contain a set of elements with certain characteristics which can be processed and stored dynamically through a file of type XML.

The input files may contain different types of errors which should be ignored, the output files will be directed to a software created in Django in which the result of the initial file processing will be presented graphically.

The initial file is capable of being received through a frontend designed in Django, which through the Http request library makes a request in which it sends the different types of information that are stored in this frontend and after that they are processed by elements belonging to the Api, said Api is designed in Flask.

Introducción

El siguiente software muestra la solución a lo presentado anteriormente, cabe resaltar que se esta teniendo el uso de dos softwares completamente distintos, los cuales tienen distintas direcciones, y se comunican mediante el protocolo Http, el frontend esta diseñado mediante el uso de Django y el Backend tiene como base Flask para su dieseño.

Desarrollo del tema

La manera en la cual funciona este proyecto es mediante la implementación de una Api, la cual es encargada del almacenamiento mediante memoria dinámica, el uso de django se resume en una implementación de distintas vistas en Html, además de forms y librerías para poder realizar de manera correcta una petición http.

La lógica del proyecto se puede dividir en los siguientes elementos.

a. Frameworks

b. Clases

c. Funciones

d. Almacenamiento

Los Frameworks utilizados fueron Django y Flask.

Django es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Es gratuito y de cógido abierto.

Es un sistema muy versátil, puede ejecutarse con cualquier framework por parte del cliente, además de poder mandar información en casi cualquier formato.

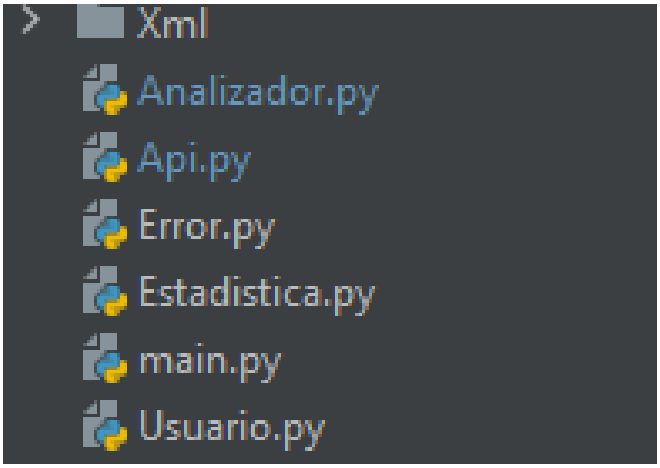
Es un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con

un mínimo número de líneas de código. Está basado en la especificación WSGI de Werkzeug y el motor de templates Jinja2

Algunas de las clases más importantes para el desarrollo de la lógica en el proyecto son:

```python
from django.shortcuts import render
from .forms import infoForm, recibir_form, codigo_form
import requests
# Create your views here.

endpoint = 'http://127.0.0.1:5000/'

def main(request):
    contexto = {
    }
    if request.method == "POST":
        form = infoForm(request.POST, request.FILES)

        if form.is_valid():
            doc = request.FILES['file'].read()
            doc = doc.decode('utf-8')

            #files = {'file': doc}

            r = requests.post(endpoint + '/enviar', data = doc)
            r1 = r.status_code
```

El siguiente extracto de código pertenece a la clase views.py, esta es una clase por defecto de Django que presenta un espacio en el cual se podrán implementar los request a al backend del programa, en este caso a la Api, muestra la implementación de un tipo de clase de la misma forma característico de Django llamado Forms.py, el cual es una representación de los elementos almacenados por medio del Html del lado del cliente y enviados al Framework de django mediante la implementación de una etiqueta de html con el mismo nombre.



Las siguientes clases muestran distintos elementos implementados den la Api de flask, el más importante a resaltar es Analizador.py el cual utiliza las demás clases para poder almacenar los datos extraídos del archivo recibido por medio del frontend.

Conclusiones

- La utilización de frameworks permite un trabajo más ordenado y limpio para poder desarrollar códigos sostenibles y escalables a largo plazo, además de soluciones complejas.
- La utilización de Http, permite llamar distintos elementos de tipo request desde el cliente, para ser consultados en la Api.
- Existen diferentes usos de Http, como “Get” o “Post” los cuales tienen distintas implementaciones y funciones intuitivas dependiendo de la manera en la cual se denominen.

Anexos:

Se utilizaron como referencia distintos framentos de código implementado y explicado en el laboratorio del Curso IPC2.

```python
from gestor import Gestor
from videojuego import Videojuego
app = Flask(__name__)
app.config["DEBUG"] = True

CORS(app)

gestor = Gestor()

#Generacion de los endpoints
@app.route('/')
def home():
    return "SERVER funciona correctamente."

@app.route('/cargarArchivo')
def cargar():
    return "Cargando archivo"

#Obtener juegos
@app.route('/games')
def getGames():
    return gestor.obtener_games()

@app.route('/archivo',methods=['POST'])
def getfile():
    datos = request.data
    json = gestor.procesar_archivo(datos)
    return json
```

```python
from django.shortcuts import render
from .forms import LoginForm, RegisterForm, EntradaForm
import requests
# Create your views here.
endpoint = 'http://127.0.0.1:5000/'
def home(request):
    response= requests.get(endpoint+'games'); #http://127.0.0.1:5000/games
    games = response.json()
    contexto = {
        'games' : games
    }
    return render(request, 'store.html', contexto)

def about(request):
    context = {
        'title':'About'
    }
    return render(request, 'about.html',context)

def login(request):
    context = {
        'title':'Login'
    }
    return render(request, 'login.html', context)
```

.