

---

## ENSAYO PROYECTO 3 DE INTRODUCCION A LA PROGRAMACION 2

---

202300673 – Álvaro Gabriel Ceballos Gil

### Resumen

En el presente ensayo se encuentra una explicación clara y detallada de todos los procesos y herramientas utilizados para la implementación del proyecto 3 de introducción a la programación 2. Para el desarrollo de esta aplicación se hizo uso de 2 framework, Flask para el backend y Django para el frontend y así representar correctamente la arquitectura cliente servidor, utilizando el formato xml para poder realizar el intercambio de información entre ambos frameworks para que puedan funcionar correctamente. Se desarrolló una solución donde por medio de un XML se obtenía un listado de sentimientos positivos y negativos, y un listado de empresas, que contenía los nombres de las empresas junto con un listado de servicios, que obtenía los distintos servicios con sus nombres y los alias para dicho nombre. El XML también contenía una lista de mensajes, y el mensaje contenía el lugar y fecha del comentario, junto con el nombre del usuario y un mensaje donde se da a conocer su punto de vista.

### Palabras clave

Framework, Django, Flask, html, python

### Abstract

*In this essay there is a clear and detailed explanation of all the processes and tools used for the implementation of project 3, introduction to programming 2. For the development of this application, 2 frameworks were used, Flask for the backend and Django. . for the frontend and thus correctly represent the client-server architecture, using the xml format to be able to exchange information between both frameworks so that they can function correctly. A solution was developed where, through XML, a list of positive and negative feelings was obtained, and a list of companies, which contained the names of the companies along with a list of services, which obtained the different services with their names and the alias for said name. The XML also contained a list of messages, and the message contained the location and date of the comment, along with the user's name and a message making their point of view known.*

### Keywords

Framework, Django, Flask, html, python

## Introducción

Ensayo se encuentra una explicación clara y detallada de todos los procesos y herramientas utilizadas para poder realizar una correcta implementación del proyecto número 3 de introducción a la programación 2.

Para poder darle solución al desarrollo del programa, se utilizó la implementación de listas de python para poder almacenar los datos extraídos del xml a la aplicación y así poder manejar los datos y generar los archivos de salida solicitados.

Para el desarrollo de esta aplicación se hizo uso de 2 frameworks, uno de estos sería flask utilizado para el backend, y el otro framework sería Django el cual serviría como frontera para poder obtener los servicios del backend.

## Desarrollo del tema

El presente proyecto se desarrolló debido a que nació la necesidad de poder desarrollar un software capaz de poder obtener mensajes mandados por el usuario, y que la aplicación sea capaz de poder procesar los mensajes y de esta forma poder estructurar la información de una forma mucho más ordenada, indicando los sentimientos del usuario y respecto a ya sea un servicio o empresa indicando si su sentimiento es positivo o negativo o neutro realizando un total de los mensajes que sean positivos negativos o neutros y realizando múltiples gráficas que ayudarán a la mejor

comprensión de los datos obtenidos en base al mensaje brindado por el usuario. A su vez el programa será capaz de poder exportar reportes tanto con gráficas que explicarán mejor hoy la información procesada y obtenida, y un reporte en PDF que contendrá la información del mismo para que pueda ser más transportable y agradable a la vista.

Para poder realizar todo lo solicitado previamente se hace uso de 2 frameworks, uno de estos triángulos y flask, que será utilizado para la implementación del backend y el otro será Django que será utilizado para la implementación del front end y a su vez poder consumir el backend del flask.

- Flask

Flask es un microframework web escrito en Python. Es ligero y sencillo, diseñado para facilitar el desarrollo de aplicaciones web. A diferencia de frameworks más grandes como Django, Flask no incluye muchas funcionalidades predefinidas, lo que permite a los desarrolladores agregar solo los componentes que necesiten (como manejo de bases de datos o autenticación). Es flexible y adecuado para proyectos de cualquier tamaño, ofreciendo control total sobre el flujo de la aplicación y el manejo de rutas, plantillas, y peticiones HTTP.

Flask es un framwework más orientado a frontend, pero este también puede ser perfectamente adaptable al backend.

```
@app.route('/config/postDiccionarioXML', methods=['POST'])
def postDiccionarioXML():
    try:
        xml_data = request.data
        dom = parseString(xml_data)

        sentimientos_positivos = [node.firstChild.nodeValue.strip() for node in dom.getElementsByTagName('sentimientos_positivos')]
        sentimientos_negativos = [node.firstChild.nodeValue.strip() for node in dom.getElementsByTagName('sentimientos_negativos')]

        empresas_analizar = []
        for empresa_node in dom.getElementsByTagName('empresa'):
            nombre = empresa_node.getElementsByTagName('nombre')[0].firstChild.nodeValue.strip()
            servicios = []
            for servicio_node in empresa_node.getElementsByTagName('servicio'):
                nombre_servicio = servicio_node.getAttribute('nombre')
                alias = [alias_node.firstChild.nodeValue.strip() for alias_node in servicio_node.getElementsByTagName('alias')]
                servicios.append(servicio(nombre_servicio, alias))
            empresas_analizar.append(empresa(nombre, servicios))

        listaDiccionario.append(diccionario(sentimientos_positivos, sentimientos_negativos, empresas_analizar))

        for mensaje_node in dom.getElementsByTagName('mensaje'):
            mensaje_text = mensaje_node.firstChild.nodeValue.strip()
```

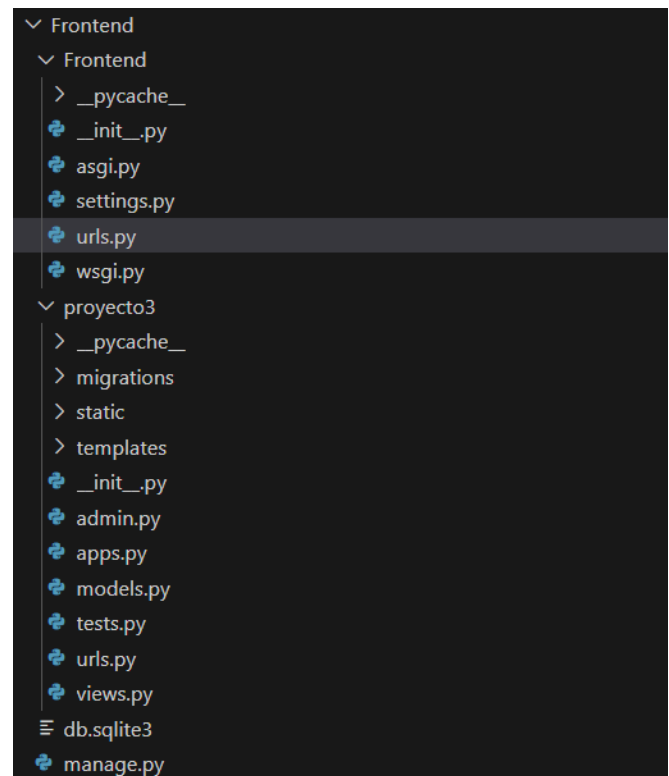
*Figura 1. Enpoons in Flask*  
*Elaboración Propia (2024)*

- Django

Django es un framework web de alto nivel escrito en Python, diseñado para simplificar el desarrollo de aplicaciones web robustas y escalables. A diferencia de Flask, Django incluye muchas funcionalidades integradas, como manejo de bases de datos, autenticación, formularios y administración, permitiendo que los desarrolladores se enfoquen en la lógica de su aplicación en lugar de en los componentes básicos. Su arquitectura basada en el patrón "MTV" (Modelo-Template-Vista) facilita la organización y mantenimiento del código. Django es ideal para proyectos de backend que requieren una estructura sólida y componentes preconfigurados.

Django es un framework web completo y de alto nivel en Python que sigue el principio de "baterías incluidas," ofreciendo herramientas listas para usar que cubren desde el manejo de bases de datos hasta la autenticación de usuarios y el enrutamiento avanzado. Su estructura se basa en el patrón Modelo-Template-Vista (MTV), similar al MVC, que facilita separar la lógica de negocio, la capa de datos, y la interfaz de usuario.

Para poder utilizar Django de necesita de todas estas carpetas y archivos



*Figura 2. Archivos de Django*  
*Elaboración Propia (2024)*

Para poder tener todas estas carpetas es necesario instalar los siguientes comandos en la terminal o consola, ya que no es necesario crearlos manualmente a continuación se dan los comandos necesarios para poder crear los archivos necesarios para poder levantar la aplicación de Django.

```
pip install django
```

```
pip install requests
```

```
django-admin startproject <<  
nombre_proyecto >>
```

```
cd << nombre_proyecto >>
```

```
py manage.py runserver
```

- HTML y Jinja 2

Una vez colocados los comandos, para poder hacer el frontend, hola necesitamos ir creando archivos html en nuestra carpeta templates, y esta utilizará de la sintaxis y uso de Jinja2 para poder conectarse con el buque. HTML con Jinja2 es la combinación de plantillas HTML y el motor de plantillas Jinja2. Jinja2 permite insertar lógica de programación dentro del código HTML, como variables, bucles, y condicionales. Esto es útil en aplicaciones web, ya que permite

generar dinámicamente el contenido de las páginas.

Por ejemplo, en Django puedes usar una plantilla HTML con Jinja2 para mostrar datos desde el backend, iterar sobre listas, o mostrar diferentes vistas basadas en condiciones, todo sin salir del archivo HTML. Jinja2 facilita la integración de lógica en las plantillas, haciéndolas más interactivas y dinámicas.

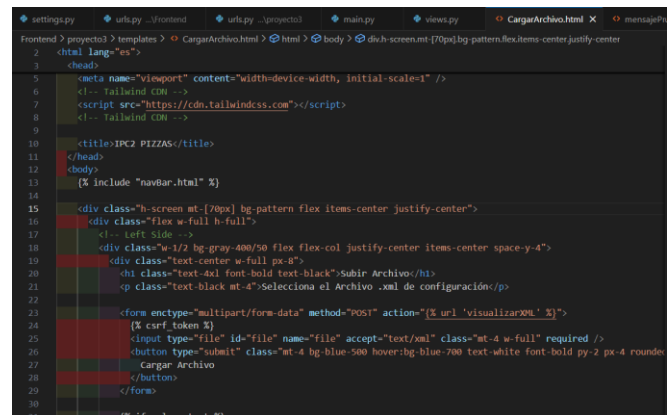


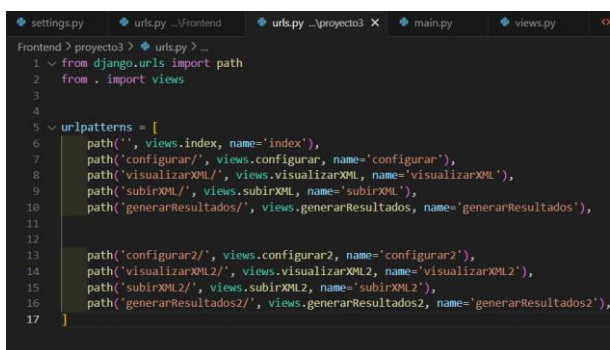
Figura 3. HTML y Jinja2  
Elaboración Propia (2024)

- Urls y Views

En Django es necesario hacer uso de nuestros archivos URL y views, ya que las url servirán para poder comunicarse entre los distintos archivos en Django, y Dios servirá para poder consumirlos endpoint del backend y a su vez renderizado los templates.

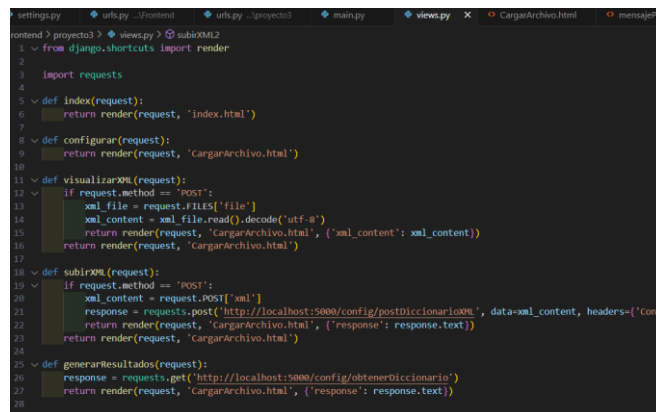
En Django, las URLs y views son fundamentales para el enrutamiento y el manejo de solicitudes. Las URLs se definen en un archivo `urls.py`, donde se mapean rutas específicas a funciones o clases en `views`. Este sistema permite que cada URL apunte a una vista particular, que es la encargada de procesar la solicitud y devolver una respuesta, en este caso un xml.

Las views son funciones o clases en el archivo `views.py` que contienen la lógica de cada página o recurso. Cuando una URL coincide con una ruta definida, se llama a la vista asociada, la cual puede realizar consultas de datos, manejar formularios o generar contenido dinámico antes de enviarlo al usuario.



```
1 from django.urls import path
2 from . import views
3
4
5 urlpatterns = [
6     path('', views.index, name='index'),
7     path('configurar/', views.configurar, name='configurar'),
8     path('visualizarXML/', views.visualizarXML, name='visualizarXML'),
9     path('subirXML/', views.subirXML, name='subirXML'),
10    path('generarResultados/', views.generarResultados, name='generarResultados'),
11
12
13    path('configurar2/', views.configurar2, name='configurar2'),
14    path('visualizarXML2/', views.visualizarXML2, name='visualizarXML2'),
15    path('subirXML2/', views.subirXML2, name='subirXML2'),
16    path('generarResultados2/', views.generarResultados2, name='generarResultados2'),
17 ]
```

Figura 4. Urls de Django  
Elaboración Propia (2024)



```
1 from django.shortcuts import render
2
3 import requests
4
5 def index(request):
6     return render(request, 'index.html')
7
8 def configurar(request):
9     return render(request, 'CargarArchivo.html')
10
11 def visualizarXML(request):
12     if request.method == 'POST':
13         xml_file = request.FILES['file']
14         xml_content = xml_file.read().decode('utf-8')
15         return render(request, 'CargarArchivo.html', {'xml_content': xml_content})
16     return render(request, 'CargarArchivo.html')
17
18 def subirXML(request):
19     if request.method == 'POST':
20         xml_content = request.POST['xml']
21         response = requests.post('http://localhost:5000/config/postDiccionarioXML', data=xml_content, headers={'Content-Type': 'application/xml'})
22         return render(request, 'CargarArchivo.html', {'response': response.text})
23     return render(request, 'CargarArchivo.html')
24
25 def generarResultados(request):
26     response = requests.get('http://localhost:5000/config/obtenerDiccionario')
27     return render(request, 'CargarArchivo.html', {'response': response.text})
```

Figura 5 views de Django  
Elaboración Propia (2024)

- d. Archivos XML

Un archivo de lenguaje de marcado extensible.

Es un documento basado en texto que se puede guardar con la extensión `.xml`. Puede escribir

XML de forma similar a otros archivos de texto.

Sirve para el manejo de información por medio de archivos estructurados.

Un archivo xml cuenta con una estructura específica, se maneja por medio se etiquetas.

Las etiquetas `<xml></xml>` se utilizan para marcar el principio y el final de un archivo XML. El contenido de estas etiquetas también

se denomina documento XML. Es la primera etiqueta que cualquier software buscará para procesar código XML.

Dependiendo de la estructura que se dese implementar, se van colocando las etiquetas.

```

ejemplo.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <solicitud clasificacion>
3 <diccionario>
4 <sentimientos_positivos>
5 <palabra> bueno </palabra>
6 <palabra> excelente </palabra>
7 <palabra> cool </palabra>
8 <palabra> satisfecho </palabra>
9 </sentimientos_positivos>
10 <sentimientos_negativos>
11 <palabra> malo </palabra>
12 <palabra> pésimo </palabra>
13 <palabra> triste </palabra>
14 <palabra> molesto </palabra>
15 <palabra> decepcionado </palabra>
16 <palabra> enojo </palabra>
17 </sentimientos_negativos>
18 <empresas_analizar>
19 <empresa>
20 <nombre> USAC </nombre>
21 <servicios>
22 <servicio nombre="inscripción">
23 <alias> Inscribi </alias>
24 <alias> inscrito </alias>
25 </servicio>
26 <servicio nombre="asignación">
27 <alias> asignado </alias>
28 </servicio>

```

Figura 6. XML

Elaboración Propia (2024)

## Conclusiones

Al utilizar 2 frameworks, tales como Flask y Django se permite un mejor desarrollo mucho más ordenado y potente que permite la creación de una mejor aplicación, ya que la combinación entre Flask y Django hoy resulta ser una combinación muy poderosa y eficaz para el desarrollo web.

## Referencias bibliográficas

Exposito López, D., García Soto, A., & Gomez Antonio Jose, M. (1999-2000). *Listas doblemente enlazadas*. Universidad de Granada.

<https://ccia.ugr.es/~jfv/ed1/tedi/cdrom/docs/ldoble.html>

## Apéndice

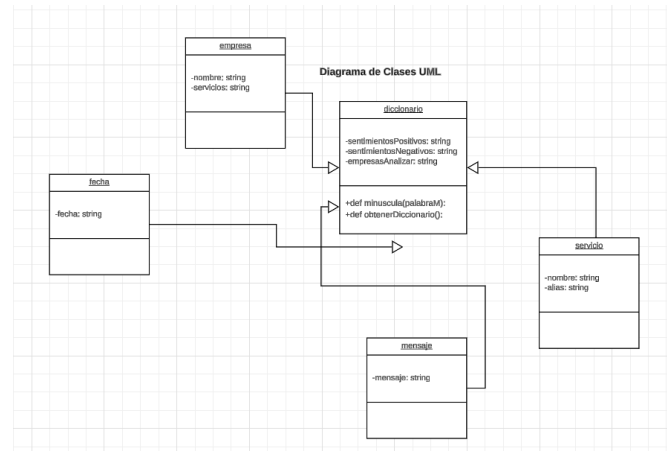


Figura 7. Diagrama de Clases Fuente: Elaboración propia

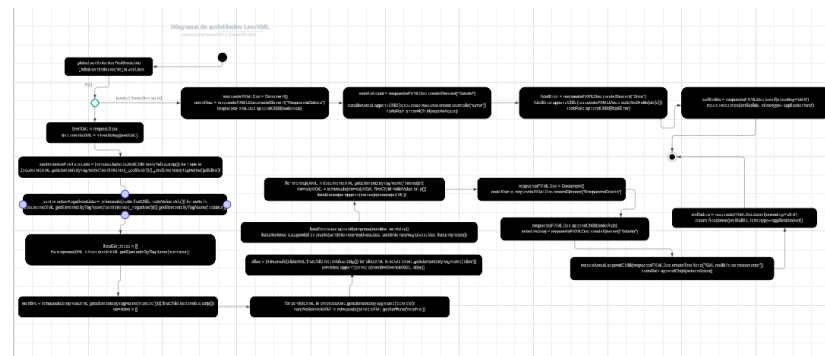


Figura 8. Diagrama Actividades: Elaboración propia