Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la Programación y Computación 2

Inga. Claudia Liceth Rojas Morales
Ing. Marlon Antonio Pérez Türk
Ing. José Manuel Ruiz Juárez
Ing. Edwin Estuardo Zapeta Gómez
Ing. Fernando José Paz González

Tutores de curso:
Josue Alfredo González Caal
Andrea María Cabrera Rosito
Paula Gabriela García Reinoso
Mario Cesar Moran Porras
Denilson Florentín de León Aguilar

PRÁCTICA #3

Objetivo General

- Permitir que el estudiante desarrolle una solución al problema dado, bajo el concepto de la programación orientada a objetos y aplicar su conocimiento de frameworks para implementar un backend con Flask y frontend con Django.

Objetivos Específicos

- Que el estudiante logre conectar un frontend con el backend de manera correcta.
- Que el estudiante aprenda a realizar peticiones a una API por medio del frontend.
- En base a los datos procesados, que el estudiante logre retornar una respuesta de tipo XML.

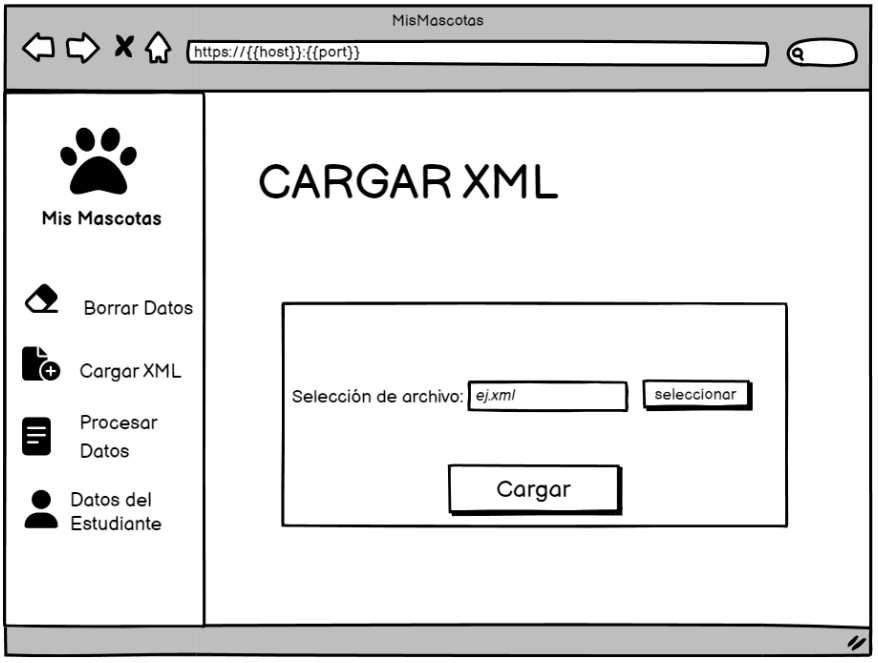
Descripción

Todas las soluciones que usted ha proporcionado a la empresa Uolmart han hecho que esta se sienta satisfecha con los resultados, es por esto mismo que usted posee ahora una gran reputación y lo han recomendado para trabajar con la empresa MisMascotitas. MisMascotitas desea que usted desarrolle su sistema de registro y control de mascotas, se le pide entonces que dicho sistema sea realizado como una aplicación web utilizando Flask y Django. Flask para backend y Django para frontend. Se le pide que en la capa de cliente, el usuario sea capaz de cargar un archivo de tipo XML que contendrá la información de las mascotas y, en base a esto, que igualmente reciba otro XML de respuesta que contendrá la información procesada para su fácil lectura. El funcionamiento es el siguiente:

Frontend - Django:
Ya que solo se le pide que implemente la parte de registros y revisión de datos, se le piden los siguientes componentes:

- **Borrado de Datos:** Esta opción funcionará para limpiar los datos, es decir que se enviará la instrucción a la API para que se tenga el estado inicial y no existan registros cargados.

- **Cargar Archivo de Registros:** En esta pantalla el usuario que utilice la aplicación web podrá cargar lo que es el archivo XML con los registros de los animales. (*Haga una revisión en la sección de archivos de entrada para visualizar el formato que estos archivos deberán tener).*

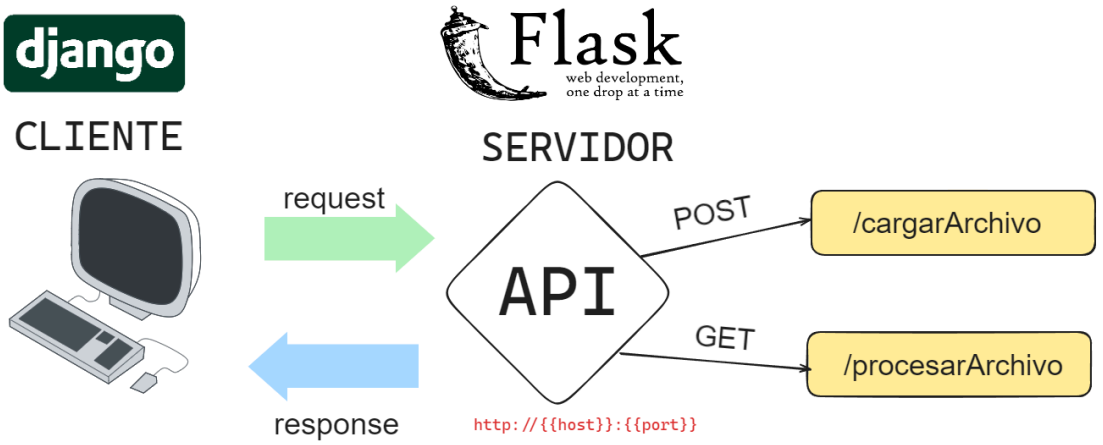


- **Revisión de datos procesados:** En esta pantalla el usuario podrá hacer revisión de los datos procesados en formato xml. (*Haga una revisión en la sección de archivos de salida para visualizar el formato que estos archivos deberán tener).*

- **Datos del estudiante:** Deberá de mostrar en frontend sus datos (nombre completo y carnet).

NOTA: Se proporcionan imágenes como sugerencia del frontend, queda siempre a discreción del estudiante.

Backend - Flask:
El backend consistirá en la API que hará uso de servicios HTTP, deberá de poder procesar los datos que reciba en formato xml.



Sugerencia para la arquitectura de la aplicación web.

NOTA: La cantidad de endpoints y nombre de estos queda a <u>discreción del estudiante</u>.

ARCHIVOS DE ENTRADA:

MisMascotas actualmente trabaja **solamente** con 3 tipos de animales:

- **Perros**
- **Gatos**
- **Conejos**

Por lo que un los archivos de entrada vendrán de esta manera:

```
<?xml version="1.0" encoding="utf-8"?>
<ingresoAnimales>
    <perro>
        <edad>2 años</edad>
        <raza>Schnauzer</raza>
    </perro>
    <gato>
        <edad>1 año</edad>
        <raza>Siamés</raza>
    </gato>
    <lagartija>
        <edad>1 año</edad>
        <raza>Indefinido</raza>
    <lagartija>
    <perro>
        <edad>6 mes</edad>
        <raza>Pug</raza>
    </perro>
    <conejo>
        <edad>9 meses</edad>
        <raza>Conejo Holandés</raza>
    <conejo>
    ...
</ingresoAnimales>
```

Puede notarse que, los animales que ingresan a MisMascotitas vendrán desordenados, y se tendrá ‘n’ cantidad en total de animales - donde ‘n’ es cualquier número entero positivo. Note que puede tenerse en el archivo animales con los que la tienda no trabaja, como en este caso “Lagartija”:

```
    <lagartija>
        <edad>1 año</edad>
        <raza>Indefinido</raza>
    <lagartija>
```

Cuando usted procese los animales debe solamente de contar perros, gatos y conejos. **Cualquier otro tipo de animal será descartado y no se tomará en cuenta para el procesamiento.**

ARCHIVO DE SALIDA:

El archivo de salida aparece cuando usted procesa los datos. En base a lo que ingrese, deberá mostrar en formato xml la cantidad total de animales con los que trabaja la tienda MisMascotas.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resultados>
    <animales>
        <perros>
            <cantidadTotal>19</cantidadTotal>
        </perros>
        <gatos>
            <cantidadTotal>26</cantidadTotal>
        </gatos>
        <conejos>
            <cantidadTotal>3</cantidadTotal>
        </conejos>
    </animales>
</resultados>
```

Consideraciones:

- Se permite el uso de listas de Python.
- **Debe de hacer uso de la POO.**
- Los archivos de entrada pueden ser **incrementales**, por lo que al ingresar un nuevo .xml deberá de verse reflejado en el procesamiento de datos con el incremento de las cantidades totales.
- Para **derecho a calificación deberá de tener Frontend y Backend.**
- Debe realizar un **diagrama de clases** sobre la solución implementada.
- El frontend deberá de ser realizado en Django.
- El backend deberá ser realizado en Flask.
- Queda siempre a discreción del estudiante el estilo del frontend y la cantidad de endpoints a utilizar, no es obligatorio implementar las sugerencias que se le muestran en el enunciado.
- La fecha de entrega debe ser **28/04/2024.**
- Se entrega el **link del repositorio** de su solución. Debe de agregar al auxiliar correspondiente para poder tener derecho a su calificación.
- No se permiten copias parciales o totales de la práctica.