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.

Backend y Frontend de una empresa

Carnet 201445840 – Luis Humberto Lémus Pérez

Resumen

La presente documentación describe las distintas funcionalidades, métodos, y validaciones que se utilizaron para la realización de la aplicación, la aplicación tienen como objetivo el análisis y la configuración que se obtienen de un archivo .xml; en el cual se lee obteniendo no solo los datos de los recursos sino también de las categorías, clientes y consumo.

Palabras clave

Paradigma
Metodos
Clases
Backend
Interfaz de usuario
Frontend
Flask
Entorno virtual
Polimorfismo
Modularidad
Herencia
Endpoint

Abstract

This documentation describes the different functionalities, methods, and validations that were used to carry out the application, the application aims at the analysis and configuration obtained from a .xml file; in which it is read obtaining not only the data of the resources but also of the categories, clients and consumption.

Keywords

Paradigm

Methods

Classes

Backend

User Interface

Frontend

Flask

Virtual Environment

Polymorphism

Modularity

Inheritance

Endpoint

Introducción

Describe las funciones y que paradigma de programación se utilizó así como la lógica utilizada para la solución del problema también se indica los diagramas de clases y se explica la utilización de backend y templates

Se Describe las validaciones utilizadas para mostrar los errores que pudiesen ocurrir así como la utilización de la aplicación web.

Desarrollo del tema

a. Explicación de Backend , Frontend, flask

El backend es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata del conjunto de acciones que pasan en una web pero que no vemos como, por ejemplo, la comunicación con el servidor.

Es la parte que ve el usuario y en la que sí se incluyen, al contrario que en Back-End, la línea de diseño y los elementos gráficos de la página. De ahí que su nombre sea Front (Parte frontal: la parte que sí se ve). Será aquí donde se incluyan **los estilos, los colores, los fondos, tamaños y las animaciones del sitio web**.

Flask es un *"micro"* Framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC.
La palabra *"micro"* no designa a que sea un proyecto pequeño o que nos permita hacer páginas web pequeñas sino que al instalar **Flask** tenemos las herramientas necesarias para crear una aplicación web funcional pero si se necesita en algún momento una nueva funcionalidad hay un conjunto muy grande **extensiones (plugins)** que se pueden instalar con **Flask** que le van dotando de funcionalidad.

b. Paradigma de programación utilizado

Programación orientada a objetos:

- Polimorfismo
- Modularidad
- Herencia

Paradigma Imperativo

- Programación modular

c. Diagrama de clases

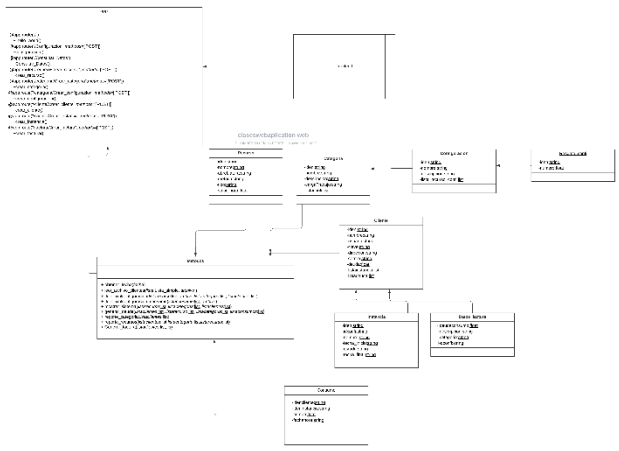


Figura 3.Diagrama de clases.

d. Métodos Principales utilizados

```python
def leer_xmlconfiguracion_consumo(listaconsumo:list, archi
 with open(archivo, encoding='utf-8') as file:
        if file.readable():
                print(True)
                xml_data= ET.fromstring(file.read())
                lst_empresas= xml_data.findall('consumo')
                for consumo in lst_empresas:
                    clase_consumo = Consumo("","",0.0,"")
                    codigo=consumo.attrib.values()
                    listas=[]
                    for x in codigo:
                        #print("la llvae es:" ,x)

                        listas.append(x)
                    clase_consumo.idencliente=listas[0]
                    #print(confi.iden)
                    clase_consumo.ideninstancia=listas[1]
                    tiempo=consumo.find('tiempo').text
                    clase_consumo.tiempo=tiempo
                    fecha=consumo.find('fechaHora').text
                    clase_consumo.fechahora=fecha
```

Figura 5. Método agrega la configuración del archivo configuración consumo

```python
def reporte_categoria(listaclientes:list):
    cliente:Cliente
    datofac:Datos_factura
    listaaux=Metodos.lista_reporte_categoria(listaclientes)
    listaaux.sort(reverse=True)
    dato=0.0
    cadena=""
    for i in range(3):
       dato=listaaux[i]
       for cliente in listaclientes:
            categoria=""
            confi=""
            for datofac in cliente.listadeuda:
                if datofac.deudaconsumo==dato:
                    categoria=datofac.categoria
                    confi=datofac.idconfi
                    cadena=cadena+"La categoria:    "+categoria+"\n"+"Con id Configuracion:    "+confi
    return cadena
def esta_ono_confi(lista:list,dato):
```

Figura 6. Método que devuelve un reporte por categoria

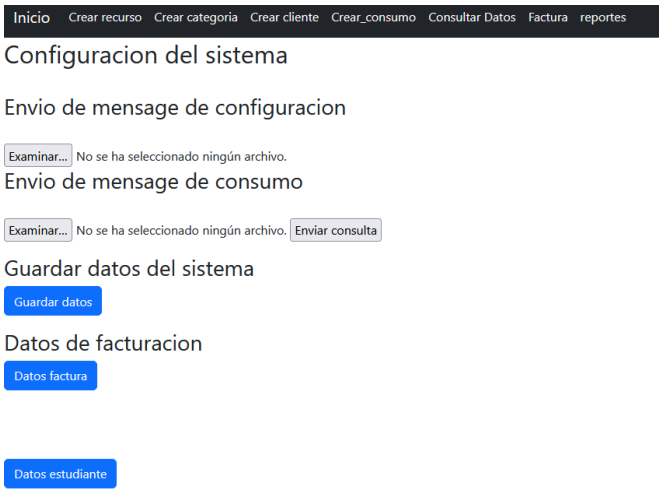
```python
def reporte_recursos(listaclientes:list,listacategoria:list,listarecurso:list):
    listaaux=Metodos.lista_aux_confi(listaclientes)
    cadenat=""
    for dato in listaaux:
                print(dato)
                listarecursoconfi,catego=Metodos.devolverlistaconfi(dato,listacategoria)
                recursoiden=""
                cadena=""
                for recursoconfi in listarecursoconfi:
                    recursoiden=recursoconfi.iden
                    recurso=Metodos.devolvervalorrecurso(recursoiden,listarecurso)

                    cadena=cadena+"Id de recurso:    "+recurso.iden+"\n"+"Nombre de recurso:  "+recurso.
                cadenat=cadenat+cadena
    return cadenat
```

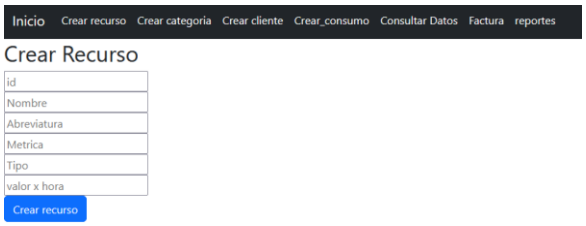
Figura 7. Método que devuelve un reporte por recurso

Figurara 7. Método que devuelve una factura por cliente

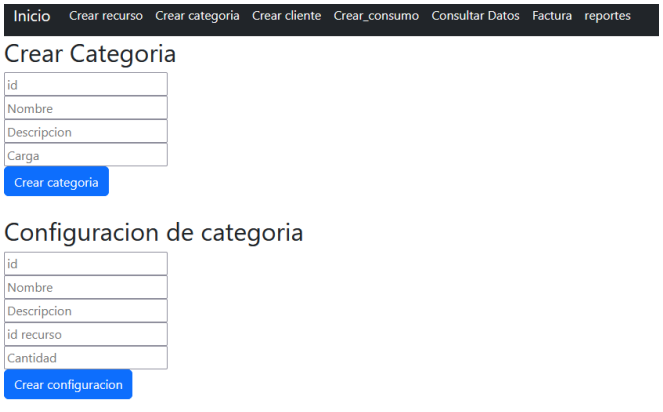
e. Interfaces de usuario



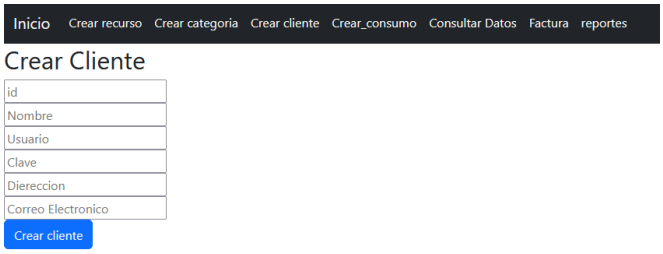
Figurara 8.pestaña principal



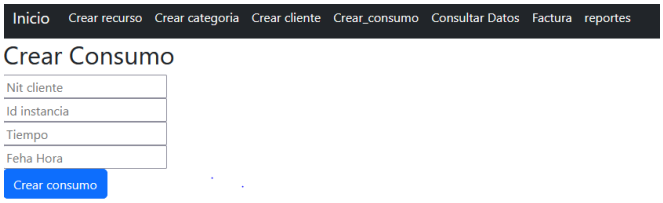
Figurara 8.crear recurso



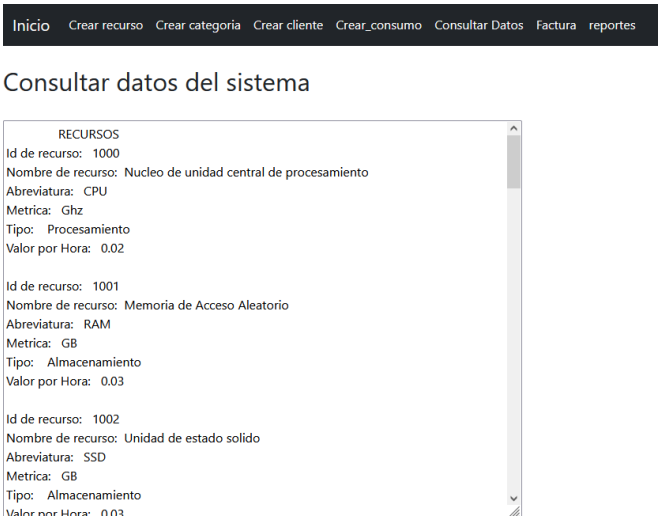
Figurara 9 Crear categoria



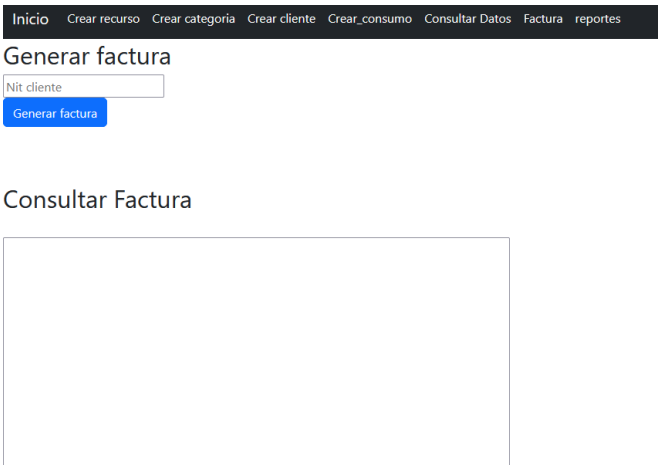
Figurara 10.crea un cliente y su instancia



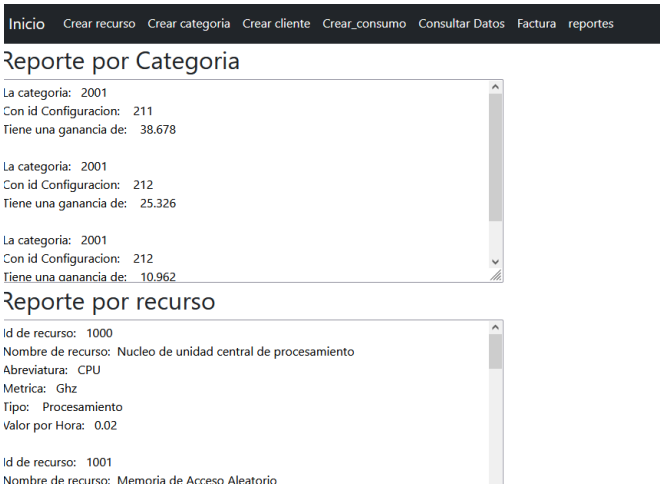
Figurara 11.crea consumo



Figurara 12.consulta datos del sistema



Figurara 12.genera factura por cliente



Figurara 12.muestra los reportes de ventas

Conclusiones

El programa es capaz de a través de dos archivos de entrada xml guardar los datos de la empresa y configuración en listas y dentro de ellas también guardar recursos, categorías así como sus configuraciones y los recursos que tiene asignado las configuraciones. También guardar clientes así como sus instancias y los consumos.
Generar un frontend para visualizar todos los endpoint que se utilizan en la aplicación.