Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2022.

TERCER PROYECTO IPC2

201902363 – CHRISTOPHER IVÁN MONTERROSO ALEGRIA

Resumen

Aplicación web creada para Tecnologías chapinas S.A para la facturación detallada de sus servicios de infraestructura de nube que aprovisiona a sus clientes, la cual consiste en la creación de cargas de sus configuraciones que agrupan los recursos necesarios. Estás configuraciones contienen los clientes y sus instancias, recursos, categorías, sus configuraciones y los recursos de la configuración. Permitiendo para la creación de estos, crearlos manualmente o realizar una carga masiva a través de un archivo xml con una estructura específica para la lectura. Para la creación de la aplicación web se utilizaron los framework: flask para el backend y django para el frontend.

Palabras clave
Django, Flask, POO, Optimización.

Abstract

Web application created for Tecnologías chapinas S.A for the detailed billing of its cloud infrastructure services that it supplies to its clients, which consists of creating loads of its configurations that group the necessary resources. These configurations contain the clients and their instances, resources, categories, their configurations, and the configuration's resources. Allowing for the creation of these, create them manually or perform a massive load through an xml file with a specific structure for reading. For the creation of the web application, the frameworks were used: flask for the backend and django for the frontend.

Keywords
Django, Flask, OOP, Optimization.

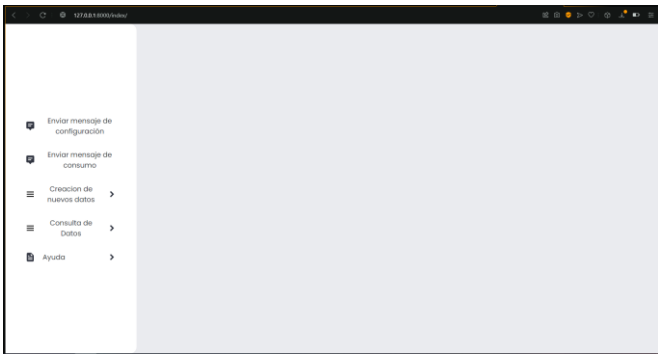
Introducción

La aplicación web inicia activando el servidor del backend para el manejo de la información. Seguido de esto se activa el servidor del frontend para usar la aplicación con la dirección. Iniciando en la pantalla principal con un menú para con opciones desplegables para el uso de la misma.

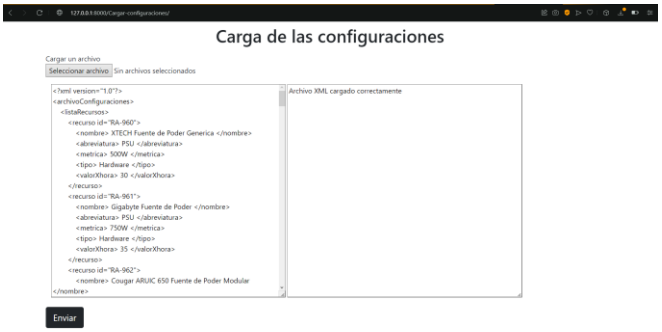
Desarrollo del tema

La aplicación web inicia en su menú principal, el cuál nos muestra las siguientes opciones:

1. Enviar mensaje de configuración.
2. Enviar mensaje de consumo.
3. Creación de nuevos datos.
4. Consulta de datos.
5. Ayuda.



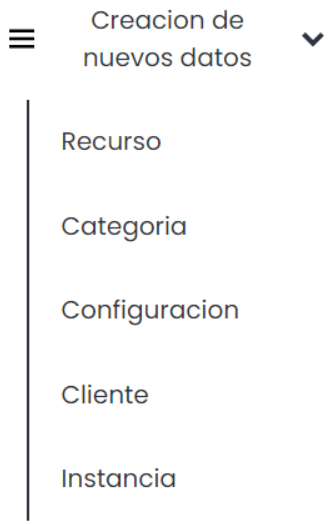
La opción 1 abre una nueva página para la carga masiva de los archivos.



Una vez cargado el archivo y presionando el botón enviar, se mostrarán en los dos cuadros mostrados en pantalla el contenido del archivo en uno y en el otro el mensaje de que se ha cargado correctamente el archivo, en caso de no cargarse de la manera correcta se mostrara un mensaje de error.

La opción 2 abre una ventana exactamente igual a la de enviar mensaje de configuración, en este se debe hacer la carga del archivo xml de consumos.

Para crear un tipo de cliente, recurso o categoría nueva debemos presionar la opción crear nuevos datos la cuál despliega unas 5 opciones más para escoger el que se desee crear



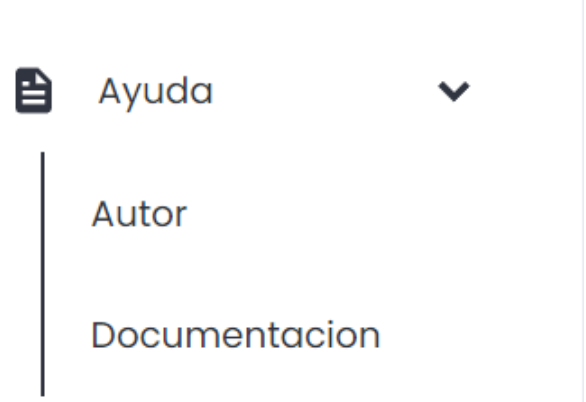
Al escoger uno de estos 5 la aplicación abrirá en una nueva ventana un apartado donde introducir los datos, para la creación del mismo

Una vez llenado los campos requeridos presionamos el botón de crear, y automáticamente se crear nuestro nuevo objeto.

Para poder visualizar cada uno de nuestros recursos, clientes etc. Debemos presionar consulta de datos en el menú principal y escoger la lista que deseamos visualizar

Una vez hecha la carga podemos generar las facturas del tiempo de atención usado por los clientes y visualizarla.

La opción de ayuda nos muestra dos opciones: Una para visualizar los datos del creador de la Aplicación, y la segunda para ver la presente documentación de la aplicación.



Conclusiones

Django es uno de los frameworks mas completos para la creación de aplicaciones web para Python.

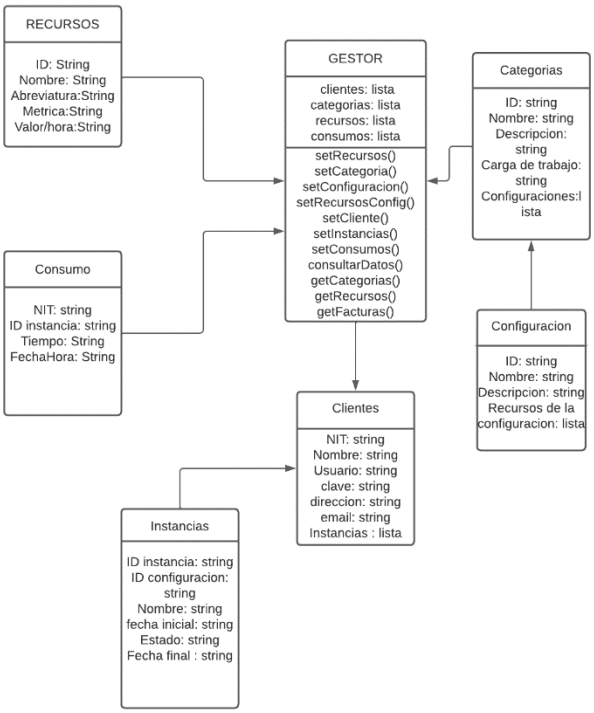
Referencias bibliográficas

MÉNDEZ, Mariano. 75.41 Algoritmos y Programación II Tda Lista y sus Derivados.

OJEDA, Luis Roberto. Tda Programacion Orientado a Objetos en Turbo. Univ. Nacional de Colombia.

Anexos:

Diagrama de clases:



Clase main.py:

Contiene todas las direcciones a utilizar para el frontend.

```
d > main.py > setRecurso
from gestor import Gestor
from xml.etree import ElementTree as ET
from DateTime import Timezones
app=Flask(__name__)
app.config['DEBUG']=True
CORS(app)
gestor=Gestor()
@app.route('/agregarConfiguraciones',methods=['post'])
def agregarConfiguraciones():···


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

@app.route('/crearRecurso',methods=['POST'])
def setRecurso():        You, 19 hours ago • Actualizacion

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

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

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

@app.route('/crearInstancia',methods=['POST'])
def setInstancia():···
```

```
def setInstancia():···

@app.route('/verRecursos',methods=['GET'])
def getRecursos():···

@app.route('/getClientes',methods=['GET'])
def getClientes():···

@app.route('/getCategorias',methods=['GET'])
def getCategorias():···

@app.route('/consultarDatos',methods=['GET'])
def getDatos():···

@app.route('/generarFacturaDetalle',methods=['GET'])
def getFacturaDetalle():
    gestor.getFacturaDetalle()
@app.route('/generarFacturaAnalisis',methods=['GET'])
def getFacturaAnalisis():
    gestor.getFacturaAnalisis()

@app.route('/abrirAutor',methods=['GET'])
def getAutor():···

if __name__=="__main__":
    app.run(host='0.0.0.0',port=3000,debug=True)
```

Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2022.

Clase gestor:

Esta clase gestiona toda la información de la aplicación web.

```
import webbrowser
You, 5 seconds ago | 1 author (You)
class Gestor:
    def __init__(self) -> None:
        self.clientes = []
        self.categorias = []
        self.recursos= []
        self.consumos = []

    def setRecursos(self,id, nombre,siglas,metrica,tipo,valorXhora): ···

    def setCategoria(self,id,nombre,descripcion,cargaTrabajo): ···

    def setConfiguracion(self,id_categoria,id_config,id_nombre,id_descripcion): ···

    def setRecurosConfig(self,id_categoria,id_config,id,cantidad): ···

    def setCliente(self,nit,nombre,usuario,clave,direccion,email): ···

    def setInstancias(self,nit,id_i,id_c,nombre,fechaInicio,estado,fechafinal): ···

    def setConsumos(self,nit,id,tiempo,fechaHora): ···

    def consultarDatos(self): ···

    def getCategorias(self):      You, 1 second ago • Uncommitted changes ···
    def getRecursos(self): ···
```

Estructuras de datos:

```
class Categoria:
    def __init__(self,id,nombre,descripcion,cargaTrabajo) -> Non
        self.id=id
        self.nombre=nombre
        self.descripcion=descripcion
        self.cargaTrabajo=cargaTrabajo
        self.configuraciones=[]
You, 2 days ago | 1 author (You)
class configuracion:
    def __init__(self,id,nombre,descripcion) -> None:
        self.id=id
        self.nombre=nombre
        self.descripcion=descripcion
        self.recursosConfiguracion=[]       You, 2 days ago • fi

You, 2 days ago | 1 author (You)
class recursosConfiguracion:
    def __init__(self,id,cantidad) -> None:
        self.id=id
        self.cantidad=cantidad
```

```
class Clientes:
    def __init__(self,nit,nombre,usuario,clave,direccion,email) -> None:
        self.nit=nit
        self.nombre=nombre
        self.usuario=usuario
        self.clave=clave
        self.direccion=direccion
        self.email=email
        self.lista_instancias=[]

You, 19 hours ago | 1 author (You)
class Instancias:
    def __init__(self,id_instancia,id_configuracion,nombre,fechaInicio,estado,fechaFinal) -> None:
        self.id_instancia=id_instancia
        self.id_configuracion=id_configuracion
        self.nombre=nombre
        self.fechaInicio=fechaInicio
        self.estado=estado
        self.fechaFinal=fechaFinal      You, 2 days ago • first commit …
```

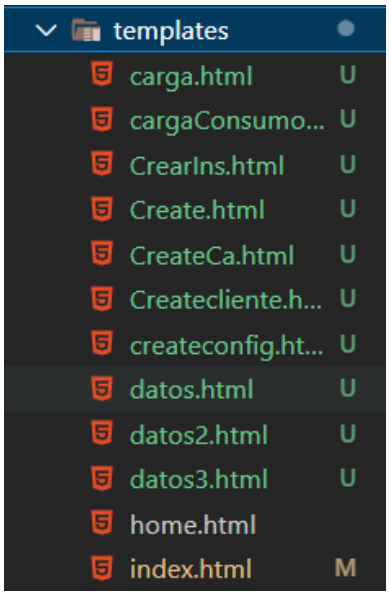
Servidor backend en ejecución:

```
{'Archivo': 'cargado exitosamente', 'Clientes creados': 3, 'Instancias creadas': 4}
127.0.0.1 - - [02/Nov/2022 14:36:37] "POST /agregarConfiguraciones HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2022 15:02:18] "GET /verRecursos HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2022 15:06:18] "POST /crearCategoria HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2022 15:06:26] "GET /getCategorias HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2022 15:08:09] "GET /verRecursos HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2022 15:17:00] "GET /abrirAutor HTTP/1.1" 200 -
```

Servidor frontend en ejecución:

```
[02/Nov/2022 15:00:32] "GET /Crear-recurso/ HTTP/1.1" 200 2414
[02/Nov/2022 15:01:20] "POST /Crear-recurso/ HTTP/1.1" 200 2527
[02/Nov/2022 15:02:14] "GET /Crear-recurso/ HTTP/1.1" 200 2414
[02/Nov/2022 15:02:18] "GET /Consulta-recursos/ HTTP/1.1" 200 8895
[02/Nov/2022 15:03:48] "GET /Crear-recurso/ HTTP/1.1" 200 2414
[02/Nov/2022 15:04:01] "GET /Crear-recurso/ HTTP/1.1" 200 2414
[02/Nov/2022 15:06:12] "GET /Crear-categoria/ HTTP/1.1" 200 2001
[02/Nov/2022 15:06:18] "POST /Crear-categoria/ HTTP/1.1" 200 2001
[02/Nov/2022 15:06:20] "GET /Crear-categoria/ HTTP/1.1" 200 2001
[02/Nov/2022 15:06:26] "GET /Consulta-Categorias/ HTTP/1.1" 200 2911
[02/Nov/2022 15:06:31] "GET /Crear-recurso/ HTTP/1.1" 200 2414
[02/Nov/2022 15:06:37] "POST /Crear-recurso/ HTTP/1.1" 200 2527
[02/Nov/2022 15:06:57] "GET /Crear-recurso/ HTTP/1.1" 200 2414
[02/Nov/2022 15:08:09] "GET /Consulta-recursos/ HTTP/1.1" 200 8895
[02/Nov/2022 15:17:00] "GET /autor/ HTTP/1.1" 200 4234
```

Templates creadas:



Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2022.

Urls para el manejo de la aplicación web:

```python
urlpatterns=[
    path('index/',views.index, name='index'),
    path('autor/',views.abrirAutor, name='Autor'),
    path('Cargar-configuraciones/',views.cargaMasiva,name='cargaConfig'),
    path('Cargar-consumos/',views.cargaMasiva2,name='cargaConsu'),
    path('Consulta-recursos/',views.verRecursos,name='consultaRecursos'),
    path('Consulta-Clientes/',views.verClientes,name='consultaClientes'),
    path('Consulta-Categorias/',views.verCategorias,name='consultaCategorias'),
    path('Crear-recurso/',views.addRecurso,name='crearRecurso'),       You, 1 second
    path('Crear-categoria/',views.addCategoria,name='crearCategoria'),
    path('Crear-configuracion/',views.addConfiguracion,name='crearConfiguracion'),
    path('Crear-cliente/',views.addCliente,name='crearCliente'),
    path('Crear-instancia/',views.addInstancia,name='crearInstancia')
]
```

Métodos para la renderización de las páginas y la transmisión de datos al backend:

```python
# You, 1 minute ago | 1 author (You)
from django.shortcuts import render
from .forms import *
# Create your views here.
import requests

endpoint="http://127.0.0.1:3000/"

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

def verRecursos(request):...

def verClientes(request):...

def verCategorias(request):...

def abrirAutor(request):...

def cargaMasiva(request):...

def cargaMasiva2(request):...
```