Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Introducción a la Programación y Computación I
Sección: C
Cat. Ing. Moisés Velásquez
Tutor Académico: Carlos Campaneros

MANUAL TECNICO
PROYECTO 2
UHOSPITAL

Nombre: Dilan Conaher Suy Miranda
Carné: 201801194

TABLA DE CONTENIDOS:

INTRODUCCION:

El presente documento describe los aspectos técnicos informáticos del sistema UHospital del BackEnd basado en una API creada en lenguaje Python haciendo uso del Framework Flask, el programa este compuesto por distintas clases para el funcionamiento correcto de los diferentes EndPoints, el documento busca familiarizar todos los aspectos técnicos del programa y dar a conocer los distintos métodos utilizados para la creación de la API, explicaremos cada detalle de manera más extensa.

INFORMACION DESTACADA:

La API fue realizada utilizando como base Python 3, importando la instalación de Flask al entorno de desarrollo, todo esto fue construido en el procesador de texto Visual Studio Code, la realización del proyecto fue de aproximadamente 30 días, cuya realización fue separada en cuatro módulos del sistema, y cada módulo aproximadamente duro una semana su realización completa, cada modulo esta construido en un manager y API hace distintos llamados a los métodos creados en dicho manager.

OBJETIVOS:

GENERAL:

- Familiariza al estudiante con el concepto de nube y programación web.
- Aplicar los conocimientos adquiridos en el curso de Introducción a la Programación y Computación 1.
- Elaborar la lógica para presentar una solución a la propuesta planteada.

ESPECIFICO:

- Utilizar el lenguaje de programación Java como herramienta de desarrollo.
- Aplicar conceptos de backend y frontend empleando una arquitectura cliente – servidor
- Acercamiento a la programación web por medio de APIs
- Construcción de programación haciendo uso del lenguaje Python.

ESPECIFICACION TECNICA:

REQUISITOS DE HARDWARE:

- **RAM:** 1GB (mínimo).
- **ROM:** 250 MB (mínimo).
- **INDISPENSABLE:** Conexión a Internet.
- **ARQUITECTURA:** de 32Bits y 64Bits

REQUISITOS DE SOFTWARE:

- **SISTEMA OPERATIVO:** Windows, Linux, MacOS (multiplataforma).
- **LEGUAJE DE PROGRAMACION:** Python
- **PLATAFORMA:** Visual Studio Code
- **TECNOLOGIA:** Desarrollo Web
- **Navegador:** Cualquiera, se recomiendo Firefox Mozilla, Google Chrome o Opera

DATOS TECNICOS UTILIZADOS PARA LA REALIZACION:

- **Sistema operativo:** Windows 10 Pro
- **RAM:** 16GB – DDR4
- **ROM:** 1TB (Utilizada aprox. 130Mb)
- **Tarjeta gráfica:** Gráficos GeForce® GTX 1660Ti (hasta 4GB)
- **Plataforma:** Visual Studio Code 1.55.2
- **Lenguaje:** Python 3.9 & Flask 1.1.2

LOGICA DEL PROGRAMA:

El programa de UHospital está constituido por una API creada en el lenguaje de programación Python haciendo uso del framework Flask a continuación las distintas clases que se usaron para su creación.

LISTADO DE CLASES:

- App (clase principal)
- Cita
- Factura
- Manager
- Medicamento
- Receta
- Usuario
- Procfile
- Requirements

Cada clase cuenta con métodos que nos ayudan a que el sistema funciona de manera eficaz logrando así el uso correcto de la aplicación poniendo en práctica los conocimientos adquiridos, a continuación, se explicara de forma mas extensa para que sirve cada una de nuestras clases.

DESCRIPCION DE CADA UNA DE LAS CLASES USADAS:

- App (clase principal): Esta clase de Python es la encargada de gestionar los endpoints para las distintas solicitudes que la página web hará para manejar los datos.

- Cita: Esta clase de Python es la que tiene los atributos de las citas y con los objetos de esta clase se almacenaran los datos de citas.

- Factura: Esta clase de Python es la que tiene los atributos de las facturas y con los objetos de esta clase se almacenaran los datos de facturas.

- Manager: Esta clase de Python es la encargada de generar todos los métodos de los endpoints para llevar una administración organizada de la API.

- Medicamento: Esta clase de Python es la que tiene los atributos de los medicamentos y con los objetos de esta clase se almacenaran los datos de medicamentos.

- Receta: Esta clase de Python es la que tiene los atributos de las recetas y con los objetos de esta clase se almacenaran los datos de recetas.

- Usuario: Esta clase de Python es la que tiene los atributos de los usuarios y con los objetos de esta clase se almacenaran los datos de usuarios.

- Procfile: Archivo necesario para subirlo al servidor de Heroku

- Requirements: Archivo necesario para subirlo al servidor de Heroku.

METODOS UTILIZADOS EN CADA CLASE PYTHON:

- App (clase principal):
  - **Home:** encargada de procesar el endpoint.
  - **Sesión:** encargada de procesar el endpoint sesión para validar datos de iniciar sesión.
  - **Registrar:** encargada de procesar el endpoint para registrar al paciente.
  - **Recuperar:** encargada de procesar el endpoint para recuperación de contraseña
  - **Registrar receta:** encargada de procesar el endpoint registrar receta
  - **Registrar factura:** encargada de procesar el endpoint para registrar factura.
  - **Citas:** encargada de procesar el endpoint para crear citas de los pacientes
  - **Masivo doctor:** encargada de procesar el endpoint para los archivos csv para agregar usuarios.
  - **Masivo paciente:** encargada de procesar el endpoint para los archivos csv para agregar usuarios
  - **Masiva enfermera:** encargada de procesar el endpoint para los archivos csv para agregar usuarios
  - **Masivo medicamento:** encargada de procesar el endpoint para los archivos csv para agregar medicamentos.
  - **Doctores:** encargada de procesar el endpoint para retornar el valor de los usuarios.
  - **Pacientes:** encargada de procesar el endpoint para retornar el valor de los usuarios.
  - **Enfermeras:** encargada de procesar el endpoint para retornar el valor de los usuarios.
  - **Admin:** encargada de procesar el endpoint para retornar el valor de los usuarios.

- **Medicamento:** encargada de procesar el endpoint para retornar el valor de los medicamentos.
- **Paciente mostrar:** encargada de procesar el endpoint para mostrar los datos solicitados.
- **Enfermera mostrar:** encargada de procesar el endpoint para mostrar los datos solicitados.
- **Dotor mostrar:** encargada de procesar el endpoint para mostrar los datos solicitados.
- **Medicamento mostrar:** encargada de procesar el endpoint para mostrar los datos solicitados.
- **Obtener receta:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener cita:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener cita asignada:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener dato nombre:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener cita completa:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener lista cita:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener lista cita aceptada:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener factura:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener Admin:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener doctor:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener enfermera:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado
- **Obtener paciente:** encargada de procesar el endpoint para obtener y enviar datos de lo solicitado

- **Eliminar usuario:** encarnada de procesar el endpoint para eliminar el dato solicitado
- **Eliminar medicamento:** encarnada de procesar el endpoint para eliminar el dato solicitado
- **Eliminar receta:** encarnada de procesar el endpoint para eliminar el dato solicitado
- **Modificar Admin:** encarnada de procesar el endpoint para modificar el dato solicitado
- **Modificar doctor:** encarnada de procesar el endpoint para modificar el dato solicitado
- **Modificar enfermera:** encarnada de procesar el endpoint para modificar el dato solicitado
- **Modificar paciente:** encarnada de procesar el endpoint para modificar el dato solicitado
- **Modificar medicamento:** encarnada de procesar el endpoint para modificar el dato solicitado
- **Modificar cita:** encarnada de procesar el endpoint para modificar el dato solicitado
- **Modificar receta:** encarnada de procesar el endpoint para modificar el dato solicitado
- **Citas aceptadas:** encarnada de procesar el endpoint para modificar el dato solicitado

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from manager import manager
from usuario import usuario
from medicamento import medicamento
from receta import receta
from factura import factura
from cita import cita
app = Flask(__name__)

CORS(app)

Manager = manager()


@app.route('/')
def home():
    return '[IPC 1] PROYECTO#2<br> -- DILAN CONAHER SUY MIRANDA - 201801194\n '
```

```python
# REGISTRO DE PACIENTES
@app.route('/registrar', methods=['POST'])
def registrar():
    dato=request.json
    user=usuario(dato['nombre'],dato['apellido'],dato['usuario'],dato['password'],dato['fecha'],d
    prueba=Manager.Registrarusuario(user)
    if prueba ==True:
        return '{"estado":"Usuario Creado Exitosamente"}'
    else:
        return '{"estado":"El Usuario Ya Existe"}'

#INICIO DE SESION
@app.route('/recuperar/<user>')
def recuperar(user):
    dato = Manager.recuperar(user)
    if dato == None:
        return '{"estado":"No"}'
    else:
        return jsonify({'estado':dato.password})


# REGISTRO DE RECETAS
@app.route('/receta', methods=['POST'])
def registrarReceta():
    dato=request.json
    recet=receta(dato['idpaciente'],dato['nombre'], dato['padecimiento'],dato['descripcion'],dato
    prueba=Manager.registrarReceta(recet)
    if prueba ==True:
        return '{"estado":"Receta Creada"}'
    else:
        return '{"estado":"ERROR"}'

# REGISTRO FACTURAS
@app.route('/registrarFactura', methods=['POST'])
def registrarFactura():
    dato=request.json
    fact=factura(dato['fecha'],dato['nombre'],dato['doctor'],dato['consulta'],dato['operacion'],d
    prueba=Manager.registrarFactura(fact)
    if prueba ==True:
        return '{"estado":"Factura Creada"}'
    else:
        return '{"estado":"ERROR"}'
```

```python
# CREAR CITASS
@app.route('/crearCita', methods=['POST'])
def Citas():
    dato=request.json
    nuevacita=cita(dato['idpaciente'],dato['fecha'],dato['hora'],dato['motivo'],dato['estado'],dato['idd
    prueba=Manager.RegistrarCita(nuevacita)
    if prueba ==True:
        return '{"estado":"Cita Creada"}'
    else:
        return '{"estado":"Ya Existe una Cita En Proceso"}'



# PARA MASIVO DOCTORES
@app.route('/masivaDoc', methods=['POST'])
def masivaDoc():
    dato=request.json
    user=usuario(dato['nombre'],dato['apellido'],dato['usuario'],dato['password'],dato['fecha'],dato['se
    prueba=Manager.Registrarusuario(user)
    if prueba ==True:
        return '{"estado":"Usuario Creado Exitosamente"}'
    else:
        return '{"estado":"El Usuario Ya Existe"}'

# PARA MASIVOS PACIENTES
@app.route('/masivaPac', methods=['POST'])
def masivaPac():
    dato=request.json
    user=usuario(dato['nombre'],dato['apellido'],dato['usuario'],dato['password'],dato['fecha'],dato['se
    prueba=Manager.Registrarusuario(user)
    if prueba ==True:
        return '{"estado":"Usuario Creado Exitosamente"}'
    else:
        return '{"estado":"El Usuario Ya Existe"}'

# PARA MASIVOS ENFERMERA
@app.route('/masivaenfer', methods=['POST'])
def masivaEnfer():
    dato=request.json
    user=usuario(dato['nombre'],dato['apellido'],dato['usuario'],dato['password'],dato['fecha'],dato['se
    prueba=Manager.Registrarusuario(user)
    if prueba ==True:
        return '{"estado":"Usuario Creado Exitosamente"}'
    else:
        return '{"estado":"El Usuario Ya Existe"}'
```

```python
# PARA MASIVOS MEDICAMENTO
@app.route('/masivaMed', methods=['POST'])
def masivaMed():
    dato=request.json
    medical=medicamento(dato['nombre'],dato['precio'],dato['descripcion'],dato['cantidad'])
    prueba=Manager.RegistrarMedicina(medical)
    if prueba ==True:
        return '{"estado":"Usuario Creado Exitosamente"}'
    else:
        return '{"estado":"El Usuario Ya Existe"}'

# PARA OBTENER DATOS DE DOCTORES
@app.route('/doctores')
def doctores():
    return Manager.ObtenerDoctor()

# PARA OBTENER DATOS DE PACIENTES
@app.route('/pacientes')
def pacientes():
    return Manager.ObtenerPacientes()

# PARA OBTENER DATOS DE ENFERMERAS
@app.route('/enfermeras')
def enfermera():
    return Manager.ObtenerEnfemeras()

# PARA OBTENER DATOS DE ADMINISTRADOR
@app.route('/admin')
def admin():
    return Manager.ObtenerAdmin()

# PARA OBTENER DATOS DEL PACIENTE (EN LA SECCION DE PACIENTES)
@app.route('/pacienteM/<pac>')
def pacienteM(pac):
    return Manager.getPaciente(pac)

# PARA OBTENER DATOS DEL PACIENTE (EN LA SECCION DE PACIENTES)
@app.route('/enfermeraM/<pac>')
def enfermeraM(pac):
    return Manager.getEnfermera(pac)

# PARA OBTENER DATOS DEL PACIENTE (EN LA SECCION DE PACIENTES)
@app.route('/doctorM/<pac>')
def doctorM(pac):
    return Manager.getDoctor(pac)
```

```python
# PARA OBTENER DATOS DE CITAS PARA EL LISTADO
@app.route('/recetasC/<doc>')
def getRecetas(doc):
    return Manager.getRecetaDoc(doc)

# PARA OBTENER DATOS DE CITAS PARA EL LISTADO
@app.route('/citas/<pac>')
def getCitas(pac):
    return Manager.getCitas(pac)

# PARA OBTENER DATOS DE CITAS PARA EL LISTADO
@app.route('/citasAsignada/<doc>')
def getCitasAsignadas(doc):
    return Manager.getCitasAsignadas(doc)

# PARA OBTENER DATOS PARA RECETA NOMBRE
@app.route('/getDatoNombre/<paciente>')
def getDatoNombre(paciente):
    dato = Manager.getDatoNombre(paciente)
    return jsonify({'name':dato.nombre,'last':dato.apellido})

# PARA OBTENER DATOS DE CITAS PARA EL LISTADO
@app.route('/citasComplete/<doc>')
def getCitasComplete(doc):
    return Manager.getCitasComplete(doc)

# PARA OBTENER DATOS DE CITAS PENDIENTES LISTADO ENFERMERA
@app.route('/citas')
def getCitasList():
    return Manager.getCitasList()

# PARA OBTENER DATOS DE CITAS ACEPTADAS LISTADO ENFERMERA
@app.route('/citasA')
def getCitasListAcept():
    return Manager.getCitasListAcept()

# PARA OBTENER DATOS DE FACTURA LISTADO
@app.route('/getFactura')
def getFactura():
    return Manager.getFactura()

# PARA MOSTRAR DATOS ADMIN EN MODIFICAR
@app.route('/mostrarAdmin')
def adminGet():
    user = Manager.RetornoAdmin()
    return jsonify({'nombre':user.nombre,'apellido':user.apellido,'password':user.password})
```

```python
# PARA MOSTRAR DATOS DE ENFERMERAS
@app.route('/mostrarEnfer/<enfermera>')
def enferGet(enfermera):
    user = Manager.RetornoEnfer(enfermera)
    return jsonify({'usuario':user.usuario,'nombre':user.nombre,'apellido':user.apellido,'fecha':

# PARA MOSTRAR DATOS DE ENFERMERAS
@app.route('/mostrarPac/<pac>')
def pacGet(pac):
    user = Manager.RetornoPac(pac)
    return jsonify({'usuario':user.usuario,'nombre':user.nombre,'apellido':user.apellido,'fecha':

# PARA MOSTRAR DATOS DE ENFERMERAS
@app.route('/mostrarMed/<med>')
def medGet(med):
    medical = Manager.RetornoMedic(med)
    return jsonify({'nombre': medical.nombre,'precio':medical.precio,'descripcion':medical.descri

# PARA OBTENER DATOS DE MEDICAMENTOS
@app.route('/medicamento')
def medicam():
    return Manager.ObtenerMedicamentos()

# PARA OBTENER DATOS DE MEDICAMENTOS
@app.route('/medicamentos')
def medical():
    return Manager.ObtenerMedicamentoList()

# RUTA PARA BORRAR USUARIOS
@app.route('/user/<usuario>', methods=['DELETE'])
def EliminarUser(usuario):
    if (Manager.eliminarUsuario(usuario)):
        return 'Usuario Eliminado'
    return 'Error'

@app.route('/medicamento/<medicina>', methods=['DELETE'])
def eliminarMedicamento(medicina):
    if (Manager.eliminarMedicamento(medicina)):
        return 'Medicamento Eliminado'
    return 'Error'

# RUTA PARA ELIMINAR CITAS EN DOCTOR
@app.route('/citaDelete/<paciente>/<padecimiento>/<fecha>/<doctor>', methods=['DELETE'])
def EliminarRecta(paciente,padecimiento,fecha,doctor):
    if (Manager.eliminarReceta(paciente, padecimiento, fecha, doctor)):
        return 'Receta Eliminada'
    return 'Error'
```

```python
# PARA MOSTRAR DATOS DE ENFERMERAS
@app.route('/mostrarRecetaC/<paci>/<fech>/<doc>/<pad>')
def recetaGet(paci,fech,doc,pad):
    user = Manager.getReceta(paci,fech,doc,pad)
    return jsonify({'idpaciente':user.idpaciente,'nombre':user.paciente,'padecimiento':user.padecimiento,'descripcion':user

@app.route('/modificarAdmin/<user>',methods=["PUT"])
def modificarAdmin(user):
    dato=request.json
    cambio = usuario(dato['nombre'],dato['apellido'],dato['usuario'],dato['password'],dato['fecha'],dato['sexo'],dato['tele
    if (Manager.modificarAdmin(user,cambio)):
        return '{"Estado":"Modificado"}'
    return '{"Estado":"Error"}'

@app.route('/modificardoc/<user>', methods=["PUT"])
def modificarDoc(user):
    dato = request.json
    cambio = usuario(dato['nombre'],dato['apellido'],dato['usuario'],dato['password'],dato['fecha'],dato['sexo'],dato['tele
    if (Manager.modificarDoc(user,cambio)):
        return '{"Estado":"Doctor Modificado"}'
    return '{"Estado":"Error"}'

@app.route('/modificarEnfer/<user>', methods=["PUT"])
def modificarEnfer(user):
    dato = request.json
    cambio = usuario(dato['nombre'],dato['apellido'],dato['usuario'],dato['password'],dato['fecha'],dato['sexo'],dato['tele
    if (Manager.modificarEnfer(user,cambio)):
        return '{"Estado":"Doctor Modificado"}'
    return '{"Estado":"Error"}'

@app.route('/modificarPac/<user>', methods=["PUT"])
def modificarPac(user):
    dato = request.json
    cambio = usuario(dato['nombre'],dato['apellido'],dato['usuario'],dato['password'],dato['fecha'],dato['sexo'],dato['tele
    if (Manager.modificarPac(user,cambio)):
        return '{"Estado":"Doctor Modificado"}'
    return '{"Estado":"Error"}'

@app.route('/modificarMed/<med>', methods=["PUT"])
def modificarMed(med):
    dato = request.json
    cambio = medicamento(dato['nombre'],dato['precio'],dato['descripcion'],dato['cantidad'])
    if (Manager.modificarMed(med,cambio)):
        return '{"Estado":"Doctor Modificado"}'
    return '{"Estado":"Error"}'
```

```python
@app.route('/modificarCita/<paciente>/<fecha>/<hora>', methods=["PUT"])
def modificarCita(paciente,fecha,hora):
    dato = request.json
    cambio = cita(dato['idpaciente'],dato['fecha'],dato['hora'],dato['motivo'],dato['estado'],dato['iddoctor'])
    if (Manager.modificarCita(paciente,fecha,hora,cambio)):
        return '{"Estado":"Cita Rechazada"}'
    return '{"Estado":"Error"}'

@app.route('/modificarCita/<paciente>/<fecha>/<hora>', methods=["PUT"])
def CitaAceptada(paciente,fecha,hora):
    dato = request.json
    cambio = cita(dato['idpaciente'],dato['fecha'],dato['hora'],dato['motivo'],dato['estado'],dato['iddoctor'])
    if (Manager.modificarCita(paciente,fecha,hora,cambio)):
        return '{"Estado":"Cita Aceptada"}'
    return '{"Estado":"Error"}'

@app.route('/modificarReceta/<paciente>/<fecha>/<doc>/<padecimiento>', methods=["PUT"])
def modificarReceta(paciente,fecha,doc,padecimiento):
    dato = request.json
    cambio = receta(dato['idpaciente'], dato['nombre'], dato['padecimiento'], dato['descripcion'], dato['iddoctor'], d
    if (Manager.modificarReceta(paciente, fecha, doc, padecimiento, cambio)):
        return '{"Estado":"Receta Modificada"}'
    return '{"Estado":"Error"}'

# EJECUTA LA API :3
if __name__ == '__main__':
    app.run(debug=True, port=8000)
```

- Manager:
    - **Constructor:** encargado de iniciar los atributos de la clase.
    - **Verificar usuario:** encargado de validar los datos para iniciar sesión.
    - **Recuperar:** encargado de retornar la contraseña del usuario.
    - **Registrar usuario:** encargado de registrar el dato ingresado por la petición.
    - **Registrar recetas:** encargado de registrar el dato ingresado por la petición.
    - **Registrar facturas:** encargado de registrar el dato ingresado por la petición.
    - **Registrar citas:** encargado de registrar el dato ingresado por la petición.
    - **Registrar medicina:** encargado de registrar el dato ingresado por la petición.
    - **Existe medicina:** encargado de registrar el dato ingresado por la petición.
    - **Existe cita:** encargado de verificar la existencia de un dato con los valores ingresados por la petición.
    - **Existe usuario:** encargado de verificar la existencia de un dato con los valores ingresados por la petición.
    - **Obtener doctor:** encargado de retornar el valor o valores solicitados por la petición de la página web.
    - **Obtener paciente:** encargado de retornar el valor o valores solicitados por la petición de la página web.
    - **Obtener enfermera:** encargado de retornar el valor o valores solicitados por la petición de la página web.
    - **Obtener Admin:** encargado de retornar el valor o valores solicitados por la petición de la página web.
    - **Obtener medicamentos:** encargado de retornar el valor o valores solicitados por la petición de la página web.
    - **Obtener medicamentos lista:** encargado de retornar el valor o valores solicitados por la petición de la página web.

- **Obtener paciente lista:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Obtener enfermera lista:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Obtener doctor listo:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Obtener dato nombre:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Obtener factura:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Obtener cita:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Obtener receta:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Obtener citas aceptadas:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Obtener citas asignadas:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Obtener citas listas:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Obtener citas listas aceptada:** encargado de retornar el valor o valores solicitados por la petición de la página web.
- **Retornar Admin:** encargado de enviar un dato solicitado por la petición de la página web.
- **Retorna doctor:** encargado de enviar un dato solicitado por la petición de la página web.
- **Retorna paciente:** encargado de enviar un dato solicitado por la petición de la página web.
- **Retornar medicamento:** encargado de enviar un dato solicitado por la petición de la página web.
- **Retornar enfermera:** encargado de enviar un dato solicitado por la petición de la página web.
- **Eliminar receta:** encargado de eliminar un dato solicitado por la petición de la página web.

- **Eliminar usuario:** encargado de eliminar un dato solicitado por la petición de la página web.
- **Eliminar medicamento:** encargado de eliminar un dato solicitado por la petición de la página web.
- **Modificar Admin:** encargado de modificar un dato solicitado por la petición de la página web.
- **Modificar doctor:** encargado de modificar un dato solicitado por la petición de la página web.
- **Modificar paciente:** encargado de modificar un dato solicitado por la petición de la página web.
- **Modificar enfermera:** encargado de modificar un dato solicitado por la petición de la página web.
- **Modificar medicamento:** encargado de modificar un dato solicitado por la petición de la página web.
- **Modificar citas:** encargado de modificar un dato solicitado por la petición de la página web.
- **Modificar recetas:** encargado de modificar un dato solicitado por la petición de la página web.

```python
# IMPORTACIONES
from usuario import usuario
from medicamento import medicamento
from cita import cita
from receta import receta
from factura import factura
import json

class manager:
    def __init__(self):
        self.usuarios =[]
        self.medicamentos =[]
        self.citas=[]
        self.recetas=[]
        self.facturas=[]
        self.usuarios.append(usuario('Carlos','Campaneros','admin',"1234",'24/07/1998','M','123456
        self.usuarios.append(usuario('Dilan','Suy','dilan',"1234",'24/07/1998','M','12345678','oft
        self.usuarios.append(usuario('Cinthia','Lopez','yesenia',"1234",'24/07/1998','F','1234567
        self.usuarios.append(usuario('Nataly','Guzman','nataly',"1234",'24/07/1998','F','12345678'
        self.medicamentos.append(medicamento('Paracetamol',"125.20","Para dolor de Cabeza","20"))
        self.citas.append(cita("nataly","05-05-2021","14:20","Dolor Cabeza","Pendiente","Ninguno")
```

```
def verificarUsuario(self,user,password):
    for x in self.usuarios:
        if x.usuario==user and x.password==password:
            return x
    return None

def recuperar(self,user):
    for x in self.usuarios:
        if x.usuario == user:
            return x
    return None

def Registrarusuario(self, user):
    validar = self.ExistenciaUser(user)
    if validar==True:
        return False
    else:
        self.usuarios.append(user)
        return True

def registrarReceta(self, receta):
    self.recetas.append(receta)
    return True

def registrarFactura(self, fact):
    self.facturas.append(fact)
    return True

def RegistrarCita(self, cita):
    validar = self.ExisteCita(cita)
    if validar==True:
        return False
    else:
        self.citas.append(cita)
        return True

def ExisteCita(self, cita):
    for i in self.citas:
        if i.idpaciente == cita.idpaciente and i.estado == 'Pendiente' or i.estado == 'Aceptada':
            return True
    return False

def ExistenciaUser(self,user):
    for i in self.usuarios:
        if i.usuario == user.usuario:
            return True
    return False
```

```
def RegistrarMedicina(self, medicina):
    validar = self.ExistenciaMedicamento(medicina)
    if validar==True:
        return False
    else:
        self.medicamentos.append(medicina)
        return True

def ExistenciaMedicamento(self,medicina):
    for i in self.medicamentos:
        if i.nombre == medicina.nombre:
            return True
    return False

def ObtenerDoctor(self):
    return json.dumps([ob.__dict__ for ob in self.usuarios if ob.tipo == 'doctor'])

def ObtenerEnfemeras(self):
    return json.dumps([ob.__dict__ for ob in self.usuarios if ob.tipo == 'enfermera'])

def ObtenerPacientes(self):
    return json.dumps([ob.__dict__ for ob in self.usuarios if ob.tipo == 'paciente'])

def ObtenerAdmin(self):
    return json.dumps([ob.__dict__ for ob in self.usuarios if ob.tipo == 'admin'])

def ObtenerMedicamentos(self):
    return json.dumps([ob.__dict__ for ob in self.medicamentos])

def ObtenerMedicamentoList(self):
    return json.dumps([ob.__dict__ for ob in self.medicamentos if ob.cantidad !='0' ])

def getPaciente(self,usuario):
    return json.dumps([ob.__dict__ for ob in self.usuarios if ob.usuario == usuario and ob.tipo == 'paciente'])

def getEnfermera(self,usuario):
    return json.dumps([ob.__dict__ for ob in self.usuarios if ob.usuario == usuario and ob.tipo == 'enfermera'])

def getDatoNombre(self,usuario):
    for i in self.usuarios:
        if i.usuario == usuario:
            return i

def getDoctor(self,usuario):
    return json.dumps([ob.__dict__ for ob in self.usuarios if ob.usuario == usuario and ob.tipo == 'doctor'])

def getCitas(self,paciente):
    return json.dumps([ob.__dict__ for ob in self.citas if ob.idpaciente == paciente ])
```

```
def getFactura(self):
    return json.dumps([ob.__dict__ for ob in self.facturas])

def getRecetaDoc(self,doc):
    return json.dumps([ob.__dict__ for ob in self.recetas if ob.iddoctor ==doc ])

def getCitasAsignadas(self,doctor):
    return json.dumps([ob.__dict__ for ob in self.citas if ob.iddoctor == doctor and ob.estado == 'Aceptada' ])

def getCitasComplete(self,doctor):
    return json.dumps([ob.__dict__ for ob in self.citas if ob.iddoctor == doctor and ob.estado == 'Completada' ])

def getCitasList(self):
    return json.dumps([ob.__dict__ for ob in self.citas if ob.estado == 'Pendiente' ])

def getCitasListAcept(self):
    return json.dumps([ob.__dict__ for ob in self.citas if ob.estado == 'Aceptada' ])

def RetornoAdmin(self):
    for i in self.usuarios:
        if i.tipo == 'admin':
            return i

def RetornoDoc(self, user):
    for i in self.usuarios:
        if i.usuario == user:
            return i

def RetornoEnfer(self, user):
    for i in self.usuarios:
        if i.usuario == user:
            return i

def RetornoPac(self, user):
    for i in self.usuarios:
        if i.usuario == user:
            return i

def RetornoMedic(self, medical):
    for i in self.medicamentos:
        if i.nombre == medical:
            return i

def eliminarReceta(self,paciente,padecimiento,fecha,doctor):
    for dato in self.recetas:
        if(dato.idpaciente==paciente and dato.padecimiento == padecimiento and dato.fecha == fecha, dato.iddoctor == doctor ):
            self.recetas.remove(dato)
            return True
```

```
def getReceta(self,paciente,fecha,doctor,padecimiento):
    for dato in self.recetas:
        if(dato.idpaciente==paciente and dato.fecha==fecha and dato.iddoctor == doctor and dato.padecimiento == padecimiento):
            return dato

def eliminarUsuario(self,user):
    for dato in self.usuarios:
        if(dato.usuario==user):
            self.usuarios.remove(dato)
            return True
    return False

def eliminarMedicamento(self,medicina):
    for dato in self.medicamentos:
        if(dato.nombre==medicina):
            self.medicamentos.remove(dato)
            return True
    return False

def modificarAdmin(self,user,Usuario):
    for data in self.usuarios:
        if (data.usuario == user):
            self.usuarios[self.usuarios.index(data)]=Usuario
            return True
    return False

def modificarDoc(self,user,Usuario):
    for data in self.usuarios:
        if (data.usuario == user and data.tipo == 'doctor'):
            self.usuarios[self.usuarios.index(data)]=Usuario
            return True
    return False

def modificarEnfer(self,user,Usuario):
    for data in self.usuarios:
        if (data.usuario == user and data.tipo == 'enfermera'):
            self.usuarios[self.usuarios.index(data)]=Usuario
            return True
    return False

def modificarPac(self,user,Usuario):
    for data in self.usuarios:
        if (data.usuario == user and data.tipo == 'paciente'):
            self.usuarios[self.usuarios.index(data)]=Usuario
            return True
    return False
```

```
def modificarMed(self,medical,medicina):
    for data in self.medicamentos:
        if (data.nombre == medical):
            self.medicamentos[self.medicamentos.index(data)]=medicina
            return True
    return False

def modificarCita(self,paciente,fecha,hora,cambio):
    for data in self.citas:
        if (data.idpaciente == paciente and data.fecha == fecha and data.hora == hora):
            self.citas[self.citas.index(data)]=cambio
            return True
    return False

def modificarReceta(self,paciente,fecha,doc,padecimiento,cambio):
    for data in self.recetas:
        if (data.idpaciente == paciente and data.fecha == fecha and data.iddoctor == doc and data.padecimiento ==padecimiento):
            self.recetas[self.recetas.index(data)]=cambio
            return True
    return False
```

- Cita:
  - **Constructor:** encargado de iniciar los atributos de la clase.

```
class cita:
    def __init__(self, idpaciente,fecha,hora,motivo,estado,iddoctor):
        self.idpaciente = idpaciente
        self.fecha = fecha
        self.hora = hora
        self.motivo = motivo
        self.estado = estado
        self.iddoctor = iddoctor
```

- Factura
  - **Constructor:** encargado de iniciar los atributos de la clase.

```
class factura:
    def __init__(self,fecha, paciente,doctor,consulta,operacion,internado,total,enfermera):
        self.paciente = paciente
        self.fecha = fecha
        self.doctor = doctor
        self.consulta = consulta
        self.operacion = operacion
        self.internado = internado
        self.total = total
        self.enfermera = enfermera
```

- Medicamento
  - **Constructor:** encargado de iniciar los atributos de la clase.

```
class medicamento:
    def __init__(self, nombre,precio,descripcion,cantidad):
        self.nombre = nombre
        self.precio = precio
        self.descripcion = descripcion
        self.cantidad = cantidad
```

- Receta
  - **Constructor:** encargado de iniciar los atributos de la clase.

```
class receta:
    def __init__(self, idpaciente,paciente,padecimiento,descripcion,iddoctor,fecha):
        self.idpaciente = idpaciente
        self.paciente = paciente
        self.padecimiento = padecimiento
        self.descripcion = descripcion
        self.iddoctor = iddoctor
        self.fecha = fecha
```

- Usuario
  - **Constructor:** encargado de iniciar los atributos de la clase.

```
class usuario:
    def __init__(self, nombre,apellido,usuario,password,fecha,sexo,telefono,especialidad,tipo):
        self.nombre = nombre
        self.apellido = apellido
        self.usuario = usuario
        self.password = password
        self.fecha = fecha
        self.sexo = sexo
        self.telefono = telefono
        self.especialidad = especialidad
        self.tipo = tipo
```

DESCRIPCION DEL FLUJO GENERAL:

- **Página web:** la pagina web es la parte visual de nuestro proyecto esta presenta todo lo grafico al usuario y le solicitan datos como nombre, apellido, usuario, contraseña datos para registro, entre otros, luego los datos ingresados son enviados a archivos JavaScript serán procesados y leídos

- **Peticiones de JavaScript:** los datos recibidos de la página web serán procesados y leídos por funciones JavaScript, luego se procese a realizar las peticiones GET, POST, PUT, DELETE correspondiendo a qué tipo de petición le haremos a la API normalmente, enviamos datos en formato JSON.

- **API procesa peticiones:** la API recibe los datos en formato JSON enviados por las peticiones realizadas por JavaScript según la petición puede ser GET, POST, PUT, DELETE, la API analiza que endpoint solicita la petición y realiza las funciones correspondientes manejadas por el manager.py

- **Respuesta de la API a JavaScript:** luego de realizar las funciones del manager.py todo método retorna una respuesta en formato JSON, estas respuestas son leídas por el JavaScript y procede a enviar una respuesta ya sea por una alerta o enviar datos directos a la página web.

- **Diagrama del flujo general de la aplicación:**

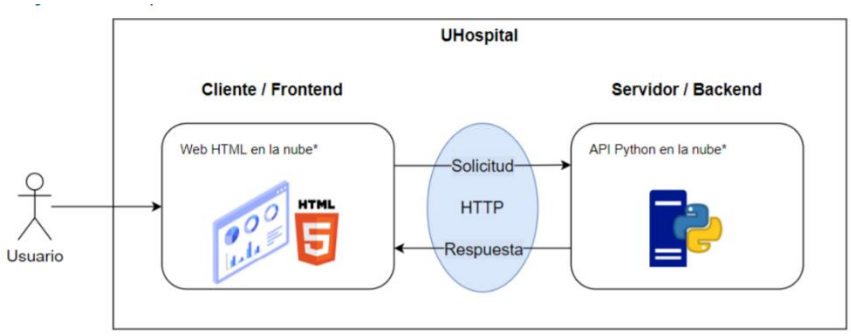

DIAGRAMA DE FLUJO GENERAL:

Diagrama general de la aplicación.