PYTHON

PROYECTO FINAL

Óscar Rodríguez Jiménez

New Technology School

Tokio.

New
Technology
School

Óscar Rodríguez Jiménez

RESPONSABLE DEL DESARROLLO DE LA APLICACIÓN

Fecha: 06/03/2023

STACK TECNOLÓGICO

- Python 3. Como lenguaje de programación base.
- Jetbrains Pycharm Community. IDE escogido para el desarrollo del proyecto.

- Flask. Framework web para Python. Simple, minimalista pero muy potente.
- SQLite. Base de datos SQL rápida y potente para instalaciones de tamaño moderado.

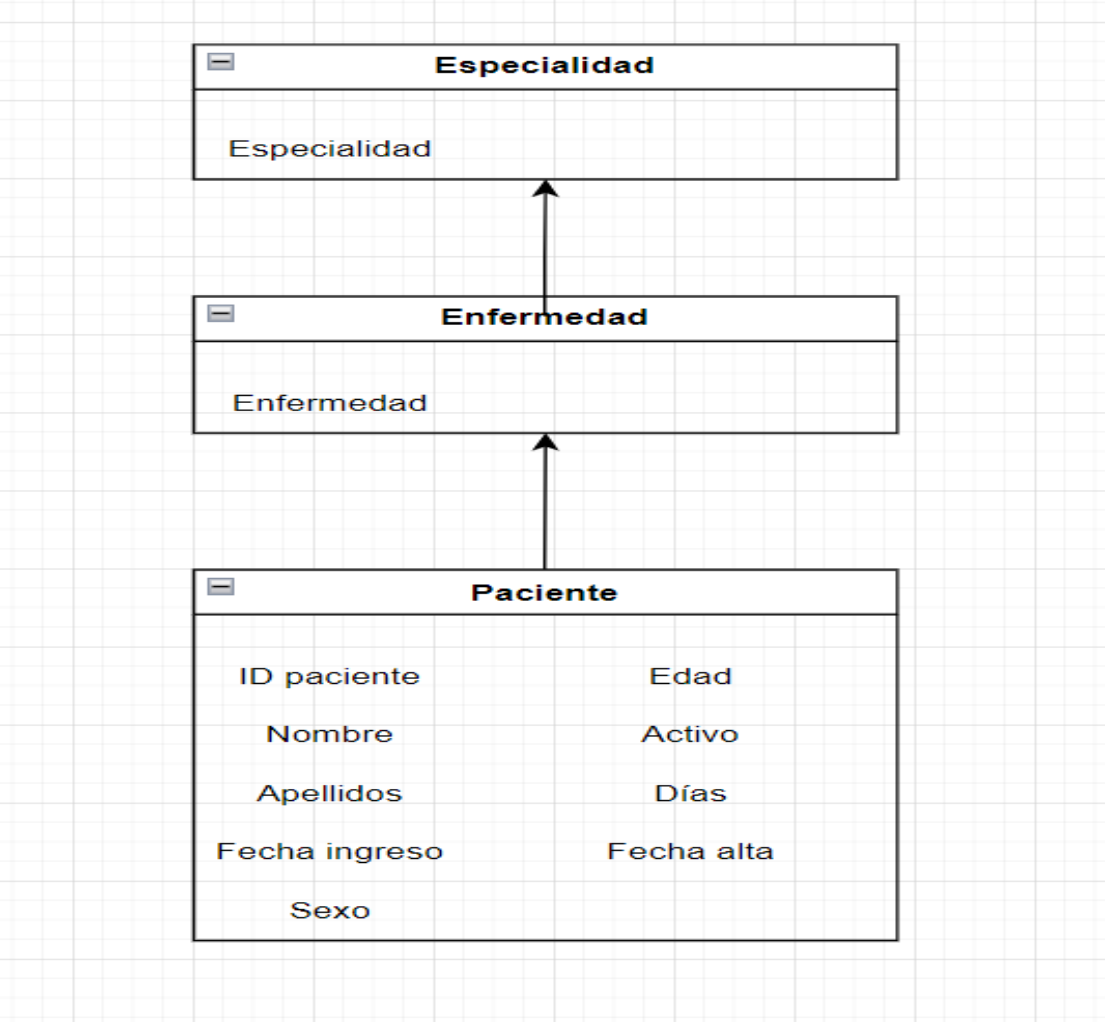
- Virtualenv. Entorno virtual de Python donde se programará el proyecto.
- SQLAlchemy. Es un módulo de Python, el cual hace de Aplicación ORM (Mapeo objeto-relacional) que permitirá trabajar con la base de datos (SQLite en este caso) de forma más sencilla, trabajando con objetos de programación y no con las tablas, sintaxis y particularidades de la base de datos escogida. En resumen, es una aplicación que facilitará la gestión y comunicación con la base de datos desde Python.

- Jinja. Motor de renderizado de páginas web.
- Bootstrap. Librería de componentes gráficos y maquetador de diseño.

DESARROLLO

1. Nos piden hacer una aplicación de gestión de enfermería con la cual guardamos todos los datos de los pacientes a una base de datos. Para ello se han creado 3 clases en el fichero models.py. La clase paciente con los atributos: id (generado automáticamente), nombre, apellidos, edad, sexo, fecha de ingreso (la cual se obtiene de un dato temporal al guardar el paciente), activo (que está a True en el momento de crear el objeto), fecha de alta (la cual se obtiene de un dato temporal al borrar el paciente) y días (que es el número de días transcurrido entre la fecha de alta y la de ingreso).

   A su vez hay 2 superclases cuyos atributos hereda la clase Paciente. Una es la clase Enfermedad (con un único atributo llamado enfermedad) y otra de la cual hereda la clase Enfermedad llamada clase Especialidad (con un único atributo llamado especialidad)



2. Al abrir la aplicación se nos muestra un formulario el cual usaremos para guardar la información en la base de datos:

Después de rellenar los distintos campos le daremos al botón "Guardar" que llamará a la función crear() del main.py:

```python
@app.route("/crear-paciente", methods=["POST"])
def crear():
    paciente=Paciente(especialidad=request.form["contenido_especialidad"], enfermedad=request.form["contenido_enfermedad"],
                      nombre=request.form["contenido_nombre"], apellidos=request.form["contenido_apellidos"],
                      fecha_ingreso=datetime.now(),sexo=request.form["contenido_sexo"],
                      edad=request.form["contenido_edad"], activo=True, fecha_alta=None, dias=None)


    paciente.fecha_ingreso = paciente.fecha_ingreso.strftime('%d-%m-%Y')
    print(type(paciente.fecha_ingreso))
    db.session.add(paciente)
    db.session.commit()
    return redirect(url_for('home'))
```

Dicha función pasará esa información a la base de datos y, hecho esto, borrará los campos que hemos rellenado.

	id_paciente	especialidad	enfermedad	nombre	apellidos	fecha_ingreso	sexo	edad	fecha_alta	activo	dias
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	1	Cardiología	Infarto	José	Martín	06-03-2023	H	68	NULL	1	NULL
2	2	Neurología	Ictus	Juan	Fernán...	06-03-2023	H	85	NULL	1	NULL
3	3	Traumatolo...	Fractura	Ana	Pérez	06-03-2023	M	37	NULL	1	NULL

Los campos de fecha_alta y días aparecen vacíos porque al estar ingresado actualmente no tiene fecha de alta ni se pueden calcular los días (fecha de alta-fecha de ingreso).

3. Para conocer todos los pacientes que se encuentran actualmente dados de alta el usuario deberá clicar sobre el botón “Ingresos” que llamará a la función ver_ingresos() del main.py y muestra todos los pacientes que tienen el atributo activo a True:

```python
@app.route("/ingresos",methods=["POST","GET"])
def ver_ingresos():

    todos_los_pacientes = db.session.query(Paciente).all()

    return render_template("ingresos.html", lista_de_pacientes=todos_los_pacientes)
```

```
{%for i in lista_de_pacientes:%}
    {%if i.activo  %}
        <div class="row">
            <tr>
                <td>{{i.nombre}}</td>
                <td>{{i.apellidos}}</td>
                <td>{{i.edad}}</td>
                <td>{{i.sexo}}</td>
                <td>{{i.fecha_ingreso}}</td>
                <td>{{i.especialidad}}</td>
                <td>{{i.enfermedad}}</td>
                <td><a href="/borrar-paciente/{{i.id_paciente}}" class="btn btn-primary btn-lg btn-block">Borrar</a></td>
                <td><a href="/editar-paciente/{{i.id_paciente}}" class="btn btn-primary btn-lg btn-block">Editar</a></td>
            </tr>
        </div>
    {% endif %}
{% endfor %}
```

Esta función derivará a una ventana donde se muestran todos los pacientes en forma de tabla.

Pacientes en la enfermería

Nombre	Apellidos	Edad	Sexo	Fecha de ingreso	Especialidad	Enfermedad		
José	Martín	68	H	06-03-2023	Cardiología	Infarto	Borrar	Editar
Juan	Fernández	85	H	06-03-2023	Neurología	Ictus	Borrar	Editar
Ana	Pérez	37	M	06-03-2023	Traumatología	Fractura	Borrar	Editar

Formulario
Altas

Cada paciente lleva asociado 2 botones. El botón “Borrar” significa que se le ha dado el alta al paciente (deja de estar ingresado). Esto se ha hecho creando un atributo llamado “activo” en la clase Paciente que está en True en el momento de crear un objeto de dicha clase (al ingresarlo) y pasa a False al darle al botón “Borrar” el cual llama a la función desactivar() del main.py que tiene como parámetro el id del paciente al que queremos dar el alta.

```python
@app.route("/borrar-paciente/<id>")
def desactivar(id):
    paciente = db.session.query(Paciente).filter_by(id_paciente=id).first()
    paciente.activo = not(paciente.activo)
```

El botón “Editar” servirá para casos en los que se haya introducido algún dato erróneo. Este llamará a la función editar() que se encuentra en el fichero main.py que tendrá el id del paciente que queramos editar como parámetro

```python
@app.route("/editar-paciente/<id>")
def editar(id):
    return render_template("editar.html",id_paciente=id)
```

y nos derivará a una página en la que se encuentran los mismos campos que aparecen en el formulario donde introduciremos la información de nuevo.

Al darle al botón “Guardar” se llamará a la función modificar() del main.py que actualizará los datos del paciente seleccionado en la base de datos.

```python
@app.route("/editar-paciente",methods=["POST"])
def modificar():
    paciente = db.session.query(Paciente).filter_by(id_paciente=int(request.form["contenido_id_paciente"])).first()
    paciente.nombre = request.form["contenido_nombre"]
    paciente.apellidos = request.form["contenido_apellidos"]
    paciente.enfermedad = request.form["contenido_enfermedad"]
    paciente.especialidad = request.form["contenido_especialidad"]
    paciente.edad = request.form["contenido_edad"]
    paciente.sexo = request.form["contenido_sexo"]
    try:
        db.session.commit()
    except:
        db.session.rollback()
    return redirect(url_for("ver_ingresos"))
```

4. Tanto en la página del formulario como en la de los ingresos aparece un botón “Altas”. Al pulsar este botón se llama a la función ver_altas() del main.py que muestra todos los pacientes que tienen el atributo activo a False.

```python
@app.route("/altas",methods=["POST","GET"])
def ver_altas():
    todos_los_pacientes = db.session.query(Paciente).all()

    return render_template("altas.html", lista_de_pacientes=todos_los_pacientes)
```

```
{%for i in lista_de_pacientes:%}
    {%if i.activo==False %}
        <div class="row">
            <tr>
                <td>{{ i.nombre}}</td>
                <td>{{i.apellidos}}</td>
                <td>{{i.edad}}</td>
                <td>{{i.sexo}}</td>
                <td>{{i.fecha_ingreso }}</td>
                <td>{{i.fecha_alta }}</td>
                <td>{{i.especialidad}}</td>
                <td>{{i.enfermedad}}</td>
                <td>{{i.dias}}</td>
                <td><a href="/eliminar-paciente/{{i.id_paciente}}" class="btn btn-primary btn-lg btn-block">Eliminar</a></td>
            </tr>
        </div>
    {% endif %}
{% endfor %}
```

El usuario es redirigido a una página llamada Altas en la cual se muestran los pacientes que ya no están ingresados, es decir se les ha dado el alta.

Pacientes dados de alta

Nombre	Apellidos	Edad	Sexo	Fecha de ingreso	Fecha de alta	Especialidad	Enfermedad	Dias ingresado	
José	Martín	68	H	06-03-2023	06-03-2023	Cardiología	Infarto	0	Eliminar

Formulario Ingresos

5. En esta página cada paciente tiene dos atributos nuevos: la fecha de alta (el día en el que el paciente ha pasado de activo a inactivo) y los días ingresados (número de días transcurridos entre la fecha de ingreso y la fecha de alta) y que se han calculado con la función desactivar().

```
paciente.dias = paciente.fecha_alta-paciente.fecha_ingreso
paciente.dias=paciente.dias.days
```

	id_paciente	especialidad	enfermedad	nombre	apellidos	fecha_ingreso	sexo	edad	fecha_alta	activo	dias
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	1	Cardiología	Infarto	José	Martín	06-03-2023	H	68	06-03-2023	0	0

6. Por último, cada paciente en la tabla de altas lleva asociado un botón de "Eliminar" que llama a la función eliminar() del main.py y que tiene como parámetro el id del paciente que queremos eliminar que lo borra definitivamente de la base de datos.

```
@app.route("/eliminar-paciente/<id>")
def eliminar(id):
    paciente = db.session.query(Paciente).filter_by(id_paciente=id).delete()
    db.session.commit()
    return redirect(url_for("ver_altas"))
```

Así queda la base de datos al eliminar de la base de datos al paciente "José Martín":

	id_paciente	especialidad	enfermedad	nombre	apellidos	fecha_ingreso	sexo	edad	fecha_alta	activo	dias
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	2	Neurología	Ictus	Juan	Fernán...	06-03-2023	H	85	*NULL*	1	*NULL*
2	3	Traumatología	Fractura	Ana	Pérez	06-03-2023	M	37	06-03-2023	0	0

CONCLUSIONES

Este proyecto tiene como objetivo la gestión los pacientes de una enfermería. Permite conocer de forma rápida los ingresados y dados de alta, así como sus datos personales relevantes y los días que han estado ingresados.

A su vez permite realizar modificaciones sobre dichos datos, borrarlo y visualizar información sobre el tiempo de ingreso.

Algunas de las posibles mejoras a realizar en el futuro serían crear una ventana nueva que permitiera filtrar la información por los distintos campos (según lo requiriera el usuario). También sería interesante implementar una funcionalidad que permitiera conocer datos estadísticos sobre la información almacenada en la base de datos, como por ejemplo, calcular el número de días promedio que está la gente ingresada, la enfermedad más frecuente, etc.