

Prácticas MongoDB

1. Crear una aplicación WEB con Python

- Vamos a realizar un pequeño ejemplo de aplicación WEB con Python y MongoDB
- Vamos a usar Flask, que es un framework ligero para crear aplicaciones WEB con Python, más que suficiente para el curso.
- Se trata de una aplicación donde podemos dar de alta, borrar y listar ALUMNOS de Apasoft Training
- Por supuesto tenemos que tener una base de datos Mongo y PyMongo instalado.
- Podemos hacer el ejemplo con Visual Studio Code o con el propio terminal.
- Podemos hacerlo directamente en nuestro entorno Python o generar un Virtual environment como el que he usado en el curso

```
python -m venv mi_entorno
source mi_entorno/bin/activate
```

- En primer lugar tenemos que instalar Flask dentro de nuestro entorno de Python.

```
pip install Flask
```

- Comprobamos que se ha instalado, por ejemplo

```
pip list | grep Flask
Flask                2.0.3
```

- Creamos un directorio para albergar la aplicación, por ejemplo llamado web

```
mkdir web
```

- Dentro creamos un fichero denominado "app.py". En realidad lo podéis llamar como queráis aunque luego hay que especificarlo
- Dentro del fichero indicamos lo primeros imports, tanto de Flask, como de PyMongo y de los componentes que necesitamos:

```
from flask import Flask, render_template, request, url_for,
redirect

from pymongo import MongoClient

from bson.objectid import ObjectId
```

- Luego indicamos el nombre que le damos a la aplicación Flask

```
app = Flask(__name__)
```

- Luego configuramos la conexión a nuestro Mongo y a la colección alumnos.
- Yo voy a usar mi conexión local, no la de ATLAS
- Voy a usar la base de datos “db1” y una colección denominada “alumnos

```
# conexion

CONEXION_LOCAL="mongodb://admin:lepanto@localhost"

CONEXION_ATLAS =
"mongodb+srv://uer:pass@tu_cluster.mongodb.net/?retryWrites=true
&w=majority"

# Usamos MongoClient para conectarnos
cliente = MongoClient(CONEXION_LOCAL)
db = cliente.db1
alumnos = db.alumnos
```

- Añadimos una primera redirección que nos permite acceder al index.html

```
@app.route('/', methods=('GET', 'POST'))
def index():
    return render_template('index.html')
```

- Creamos una directorio llamado “templates” dentro del directorio “web” que nos permita poner los ficheros HTML. Esto es una característica de Flask para facilitar el trabajo

```
mkdir templates
```

- Nos vamos al directorio templates y creamos un fichero denominado “index.html”
- Copiamos el contenido del fichero “index_inicial” que tienes en los recursos del capítulo en el “index.html” que acabas de crear.
- Configuramos el entorno de FLASK. Esto nos permite identificar la aplicación y el entorno a trabajar

```
export FLASK_APP=app
```

```
export FLASK_ENV=development
```

- Lanzamos el entorno. Nos aseguramos de estar en el directorio “web” que es donde tenemos la aplicación “app.py”

```
flask run

* Environment: production

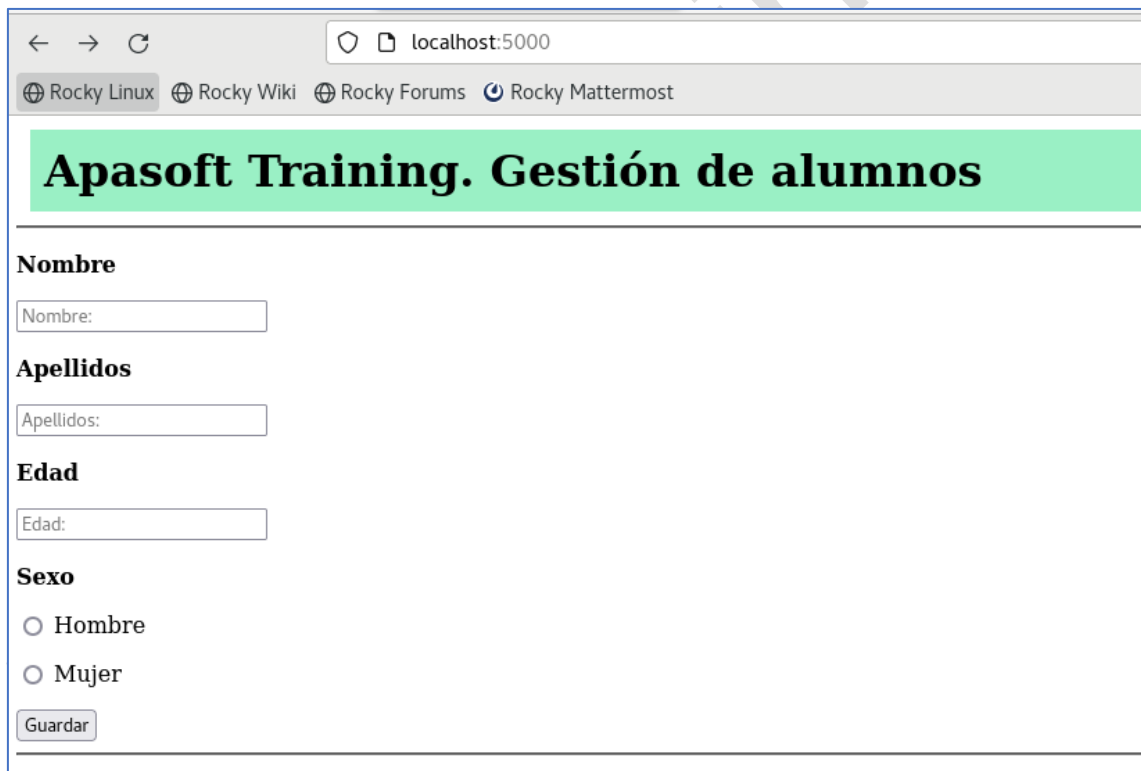
WARNING: This is a development server. Do not use it in a
production deployment.

Use a production WSGI server instead.

* Debug mode: off

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- Si todo va bien, debería abrir una entrada por el puerto 5000
- Nos vamos a un navegador y probamos acceder por ese puerto. Debería salir una página similar a la siguiente



← → ↻ localhost:5000

Rocky Linux Rocky Wiki Rocky Forums Rocky Mattermost

Apasoft Training. Gestión de alumnos

Nombre

Nombre:

Apellidos

Apellidos:

Edad

Edad:

Sexo

☐ Hombre

☐ Mujer

- Ahora vamos a poner el código para insertar alumnos, ya que la página por ahora no hace nada.
- Nos vamos a “app.py” y sustituimos el código de la función “def index()” por este código. Contiene la recuperación de campos del formulario y un insert_one para hacer la inserción de la fila en la colección “alumnos”

```
@app.route('/', methods=('GET', 'POST'))
def index():
    if request.method=='POST':
        nombre = request.form['nombre']
        apellidos = request.form['apellidos']
        edad = request.form['edad']
        sexo = request.form['sexo']
        alumnos.insert_one({'nombre': nombre, 'apellidos':
apellidos,'edad': edad,'sexo': sexo})
        return redirect(url_for('index'))

    all_alumnos= alumnos.find()
    return render_template('index.html', alumnos=all_alumnos)
```

- Grabamos el fichero de nuevo y recargamos la página WEB
- Probamos a grabar un alumno. Si todo es correcto, el alumno debería aparecer en la página

Apasoft Training. Gestión de alumnos

Nombre

Apellidos

Edad

Sexo

☐ Hombre

☐ Mujer

Nombre: pedro perez

Edad: 20 Sexo: Hombre

- Ahora nos vamos a mongoshell o a mongo Compass y comprobamos que se ha creado la colección alumnos y el alumno correspondiente

```

db1> show collections
alumnos
c2
cazas
ciudades
departamentos
edificios
empleados
facturas
libros
libros1
manzanas
países
db1> db.alumnos.find()
[
  {
    _id: ObjectId("6492b3db59bb155d1c59ecae"),
    nombre: 'pedro',
    apellidos: 'perez',
    edad: '20',
    sexo: 'Hombre'
  }
]
db1>

```

- Ahora vamos a añadir la parte del borrado de alumnos.
- Accedemos al fichero app.py y ponemos este código al final. Como vemos tiene un delete_one

```

@app.post('/<id>/delete/')
def delete(id):
    alumnos.delete_one({"_id": ObjectId(id)})
    return redirect(url_for('index'))

```

- En el fichero index.html ponemos el código-formulario para implementar la parte visual del borrado. El bucle “for” debería quedar de la siguiente manera

```
{% for alumno in alumnos %}
<div class="alumnos">
  <p>Nombre: {{ alumno['nombre'] }} {{ alumno['apellidos'] }}</p>
  <p>Edad: {{ alumno['edad'] }} Sexo:{{ alumno['sexo'] }}</i></p>
  <form method="POST" action="{{ url_for('delete',
id=alumno['_id']) }}" >
    <input type="submit" value="Borrar Alumno"
    onclick="return confirm('¿Quieres borrar el alumno?') ">
  </form>
</div>
{% endfor %}
```

- Grabamos todo y recargamos la página WEB
- En cada alumno debe aparecer el botón de "delete"

☐ Hombre

☐ Mujer

Nombre: pedro perez

Edad: 20 Sexo: Hombre

- Podemos probarlo pulsando el botón y comprobar luego en nuestro mongosh que el documento se ha borrado correctamente