

INGENIERÍA DEL SOFTWARE III ACTUALIDAD INFORMÁTICA

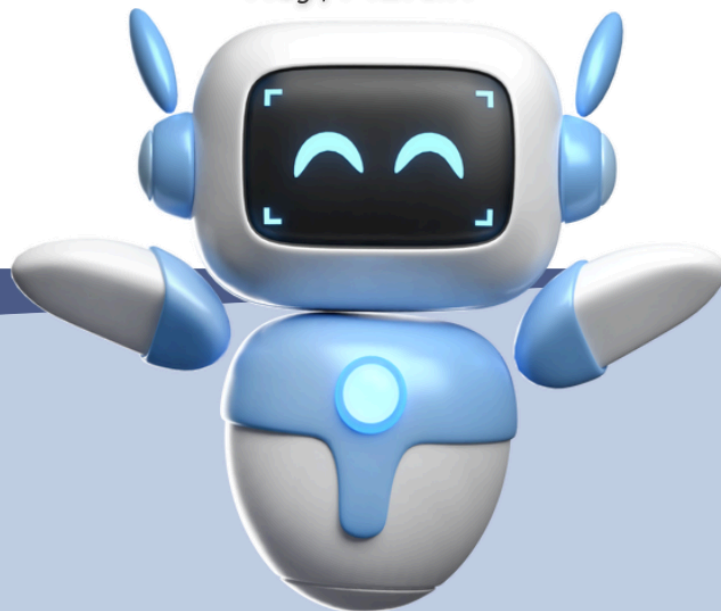
TRABAJO PRÁCTICO N° 5

Integrantes:

Antúñez, Elías Emanuel
Savallich, Milagros Antonella

Equipo de cátedra:

Caballero, Sergio
Rey, Martín



1. Creamos los Issues para trabajar con el TP-05, y asignamos uno a cada uno de los integrantes del grupo, sin contar la documentación que se asignó a ambos miembros:

```
Implementación Inicial del Entorno #12
Implementación Completa del Entorno #13
Documentación de la Gestión de Entor... #14
```

2. Primero clonamos el repositorio en nuestra máquina local y abrimos visual studio code para trabajar con la implementación inicial del entorno (issue #12).

```
MINGW64:/d/Tp05-GegtiondeEntornos/Antunez-Savallich-TP4-IS3ACI-
Milagros@Mili MINGW64 /d/Tp05-GegtiondeEntornos
$ git clone https://github.com/EliasAntunez/Antunez-Savallich-TP4-IS3ACI-.git
Cloning into 'Antunez-Savallich-TP4-IS3ACI-'...
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 31 (delta 4), reused 17 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (31/31), 1.02 MiB | 194.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.

Milagros@Mili MINGW64 /d/Tp05-GegtiondeEntornos
$ ls
Antunez-Savallich-TP4-IS3ACI-/

Milagros@Mili MINGW64 /d/Tp05-GegtiondeEntornos
$ cd Antunez-Savallich-TP4-IS3ACI-

Milagros@Mili MINGW64 /d/Tp05-GegtiondeEntornos/Antunez-Savallich-TP4-IS3ACI- (m
ain)
$ code .|
```

3. Creamos una rama “features/implementación-inicial-entorno” para trabajar con él issue #12.

```
feature/implemetacion-inicial-entorno
```

4. Generamos y activamos un entorno virtual, con los siguientes comandos:

```
python -m venv .venv --prompt="eventos"
```

```
PS D:\Tp05-GegtiondeEntornos\Antunez-Savallich-TP4-IS3ACI-> .venv/Scripts/activate
```

5. Instalamos los paquetes con los que vamos a trabajar, en este caso de ejemplo: Flask y psycopg2-binary

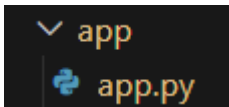
```
(eventos) PS D:\Tp05-GegtiondeEntornos\Antunez-Savallich-TP4-IS3ACI-> pip install flask  
collecting flask
```

```
pip install psycopg2-binary
```

6. Ejecutamos el siguiente comando para congelar las dependencias exactas:

```
(eventos) PS D:\Tp05-GegtiondeEntornos\Antunez-Savallich-TP4-IS3ACI-> python -m pip freeze > requirements.txt
```

7. Creamos un directorio “app” y dentro creamos el archivo “app.py” con un contenido básico de “Hola mundo desde Flask + Docker!”.



8. Creamos el archivo “Dockerfile” en la raíz del proyecto, con el siguiente contenido:

```
FROM python:3.13-alpine  
  
# Establecemos el directorio de trabajo  
WORKDIR /app  
  
# Copiamos el archivo de dependencias  
COPY requirements.txt .  
  
# Instalamos las dependencias  
RUN pip install --no-cache-dir -r requirements.txt  
  
# Copiamos el resto de los archivos  
COPY app/ .  
  
# Exponemos el puerto 8000  
EXPOSE 8000  
  
# Comando para ejecutar la aplicación  
CMD ["python", "app.py"]
```

9. Construimos el contenedor con el comando:

```
docker build -t nombre-del-contenedor
```

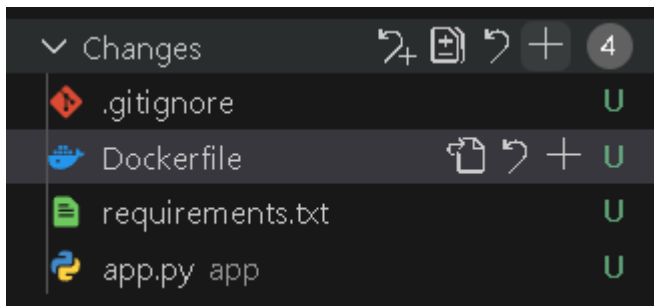
10. Ejecutamos el contenedor anteriormente creado con el comando:

```
docker run -it --rm -p 5000:5000 nombre-del-contenedor
```

11. Abrimos el navegador para verificar que el contenedor funcione correctamente. Para ello utilizamos un código en Python con flask sencillo que imprime “Hola mundo desde Flask + Docker!”




12. Añadimos todos los cambios realizados al área de stage, para posteriormente hacer el commit con el mensaje “Implementación Inicial del Entorno de desarrollo. Closes #12”, cerrando de esa forma el issue con el que estábamos trabajando (#12). Posteriormente sincronizamos los cambios con el repositorio remoto.



13. Entramos a GitHub para hacer el merge y unir la rama principal con la rama creada para la realización del issue #12 “Implementación Inicial del Entorno de desarrollo.”. (nos olvidamos las capturas)

14. Creamos otra rama “features/implementacion-completa-entorno” para trabajar con el Issue #13 (“Implementación completa del entorno de desarrollo”)

 features/implementacion-completa-entorno

15. Modificamos el archivo “app.py” para:

- Crear la ruta para inicializar la BD:

```
def obtener_conexion():
    return psycopg2.connect(DATABASE_URL)

# Ruta para inicializar la base de datos
@app.route("/inicializar-bd", methods=['GET'])
def inicializar_bd():
    try:
        # Obtenemos conexión y cursor
        conexion = obtener_conexion()
        cursor = conexion.cursor()

        # Creamos la tabla de eventos si no existe
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS eventos (
                id SERIAL PRIMARY KEY,
                nombre VARCHAR(100) NOT NULL,
                fecha DATE NOT NULL,
                lugar VARCHAR(100)
            )
        """)
        conexion.commit()

        return "Base de datos inicializada correctamente", 200

    except Exception as e:
        return f"Error: {str(e)}", 500
    finally:
        if 'conexion' in locals():
            conexion.close()
```

- Crear la ruta para crear un nuevo evento:

```
# Ruta para crear un nuevo evento
@app.route("/eventos", methods=['POST'])
def crear_evento():
    try:
        # Obtenemos los datos del cuerpo de la petición
        datos = request.get_json()
        nombre = datos['nombre']
        fecha = datos['fecha']
        lugar = datos.get('lugar', '') # Campo opcional

        # Insertamos el evento en la base de datos
        conexion = obtener_conexion()
        cursor = conexion.cursor()
        cursor.execute(
            "INSERT INTO eventos (nombre, fecha, lugar) VALUES (%s, %s, %s) RETURNING id",
            (nombre, fecha, lugar))
        id_evento = cursor.fetchone()[0]
        conexion.commit()

        return jsonify({
            "mensaje": "Evento creado exitosamente",
            "id": id_evento
        }), 201

    except Exception as e:
        return jsonify({"error": str(e)}), 400
    finally:
        if 'conexion' in locals():
            conexion.close()
```

- Crear la ruta para listar todos los eventos:

```
# Ruta para listar todos los eventos
@app.route("/eventos", methods=['GET'])
def listar_eventos():
    try:
        conexion = obtener_conexion()
        cursor = conexion.cursor()

        # Obtenemos todos los eventos
        cursor.execute("SELECT id, nombre, fecha, lugar FROM eventos")
        eventos = cursor.fetchall()

        # Formateamos la respuesta
        resultado = []
        for evento in eventos:
            resultado.append({
                "id": evento[0],
                "nombre": evento[1],
                "fecha": evento[2].isoformat(),
                "lugar": evento[3]
            })

        return jsonify(resultado), 200

    except Exception as e:
        return jsonify({"error": str(e)}), 500
    finally:
        if 'conexion' in locals():
            conexion.close()
```

- Inicializamos la aplicación en el puerto 8000:

```
# Iniciamos la aplicación
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

16. Creamos el archivo “docker-compose.yml” en la raíz del proyecto, con el siguiente contenido:

```
services:
  web:
    build: .
    ports:
      - "${FLASK_PORT}:${FLASK_PORT}"
    env_file:
      - .env # Carga todas las variables
    environment:
      -
DATABASE_URL=postgresql://${POSTGRES_USER}:${POSTGRES_PASSWORD}@db:5432
/${POSTGRES_DB}
    depends_on:
      - db

  db:
    image: postgres:13
    env_file:
      - .env
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

volumes:
  postgres_data:
```

17. Para iniciar los servicios, ejecutamos el siguiente comando:

```
docker-compose up --build
```

Nota: Ahora es momento de probar que todo esté funcionando correctamente en el contenedor.

18. Inicializamos la BD enviando una solicitud "GET", utilizando una extensión de VS code "REST Client":

```
Send Request
GET http://localhost:8000/inicializar-bd HTTP/1.1
```

Respuesta:

```
Response(132ms) X
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.13.5
3 Date: Sun, 15 Jun 2025 02:30:59 GMT
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 40
6 Connection: close
7
8 Base de datos inicializada correctamente
```

19. Creamos un evento:

```
Send Request
POST http://localhost:8000/eventos HTTP/1.1
Content-Type: application/json

{
  "nombre": "Curso de Programacion II",
  "fecha": "2025-12-16",
  "lugar": "Aula 2 - FCEQYN"
}
```

Respuesta:

```
Response(95ms) X
1 HTTP/1.1 201 CREATED
2 Server: Werkzeug/3.1.3 Python/3.13.5
3 Date: Sun, 15 Jun 2025 02:33:10 GMT
4 Content-Type: application/json
5 Content-Length: 48
6 Connection: close
7
8 {
9   "id": 4,
10  "mensaje": "Evento creado exitosamente"
11 }
```

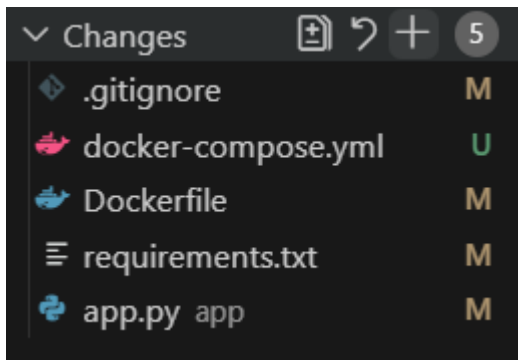
20. Listamos todos los eventos:

```
Send Request  
GET http://localhost:8000/eventos HTTP/1.1
```

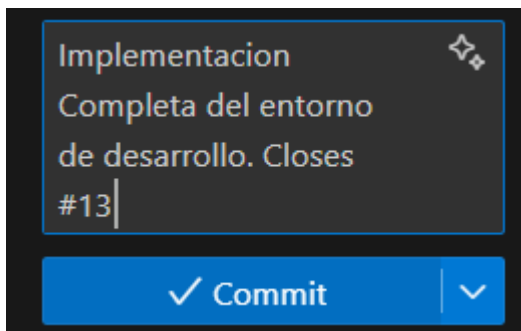
Respuesta:

```
Response(38ms) X  
1 HTTP/1.1 200 OK  
2 Server: Werkzeug/3.1.3 Python/3.13.5  
3 Date: Sun, 15 Jun 2025 02:34:27 GMT  
4 Content-Type: application/json  
5 Content-Length: 422  
6 Connection: close  
7  
8 √ [  
9 √ {  
10     "fecha": "2025-12-15",  
11     "id": 1,  
12     "lugar": "FCEQYN - Modulo Apostoles - Laboratorio",  
13     "nombre": "Curso de Configuracion de Entornos de Desarrollo"  
14 },  
15 √ {  
16     "fecha": "2025-12-16",  
17     "id": 2,  
18     "lugar": "Aula Magna - FCEQYN",  
19     "nombre": "Curso de Electronica Basica"  
20 },  
21 √ {  
22     "fecha": "2025-12-16",  
23     "id": 3,  
24     "lugar": "Aula 2 - FCEQYN",  
25     "nombre": "Curso de Programacion"  
26 },  
27 √ {  
28     "fecha": "2025-12-16",  
29     "id": 4,  
30     "lugar": "Aula 2 - FCEQYN",  
31     "nombre": "Curso de Programacion II"  
32 }  
33 ]
```

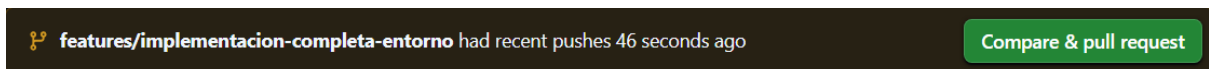
21. Una vez que todo funciona correctamente y ya tenemos la implementación completa del entorno de desarrollo, procedemos a añadir todos los cambios al area de stage:



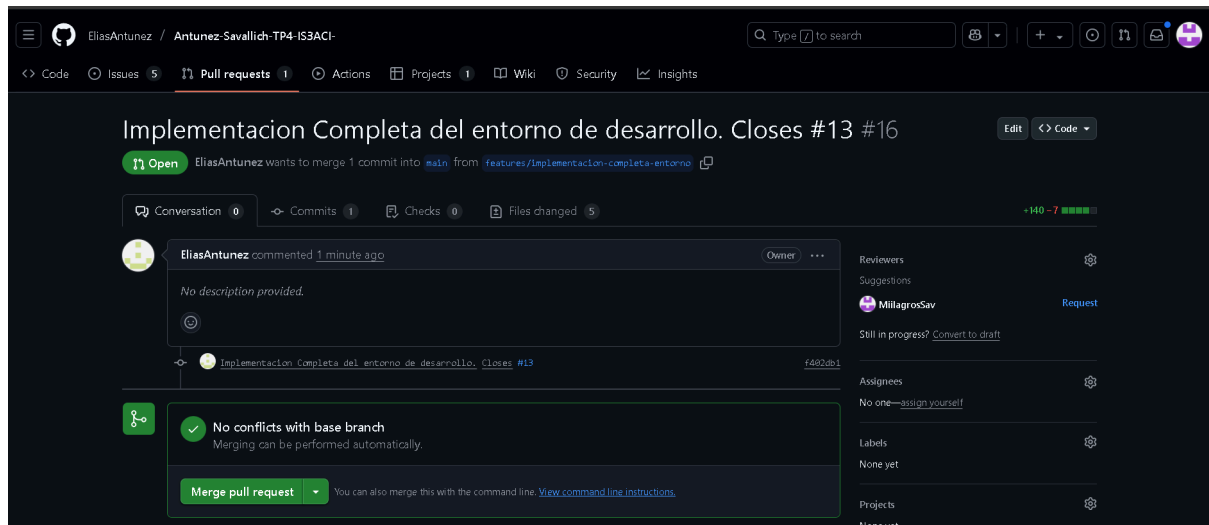
22. Una vez añadidos los cambios, realizamos el commit, cerrando el Issue #13 (Implementación completa del entorno de desarrollo):



23. Realizamos el Pull Request.



24. Otra persona del grupo verifica los cambios y realiza el Merge:



Consideraciones: En el presente trabajo NO se documentaron los archivos “.gitignore” ni “requirements.txt”.

Link del Repositorio:

<https://github.com/EliasAntunez/Antunez-Savallich-TP4-IS3ACI->