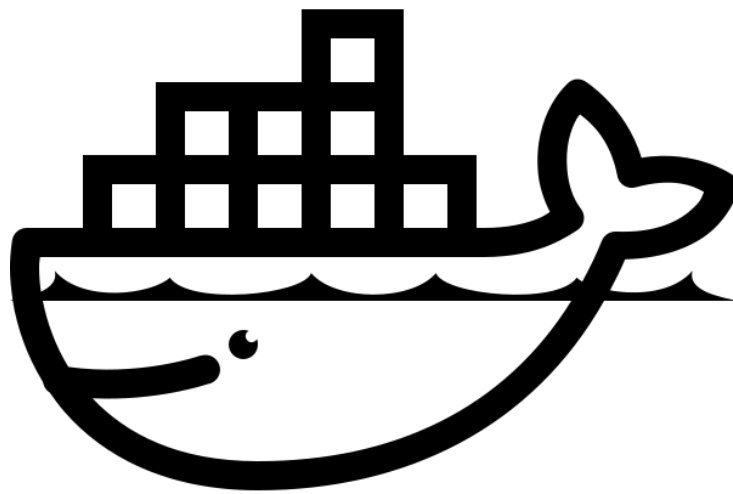


# Reporte de creación de aplicación.



Universidad	Materia	Grupo	Profesor	Carrera
UPSLP	Sistemas Virtuales	T57A	Mtro. Manuel Chávez	ITI

Nombre	Matrícula
Aneth Alejandra López Cerda	173577
Christian Aarón Zavala Sánchez	171817

# 1. Presentación.

**Objetivo:** Desarrollar una aplicación web que permita la captura de datos mediante un formulario, garantizando su almacenamiento en una base de datos MySQL. Además, asegurar la persistencia de los registros incluso en caso de reinicio del contenedor.

## Tecnologías usadas para la aplicación:

- Docker Compose (Servicios: web, db)
- Python 3.11 (Flask)
- MySQL 5.7
- HTML/CSS

# 2. Reporte de creación de aplicación.

## 2.1 Planificación.

Para desarrollar la aplicación, establecimos los siguientes objetivos:

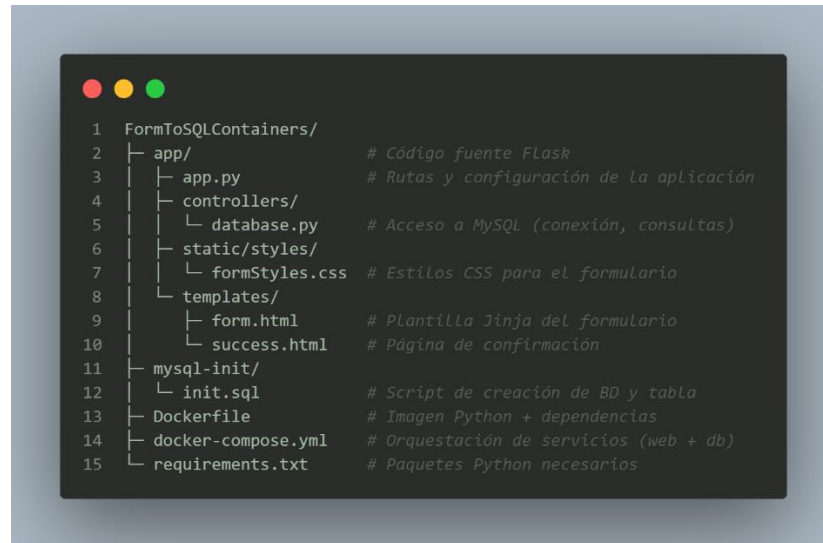
- **Captura de datos:** Implementar un formulario web para el ingreso de información de estudiantes.
- **Almacenamiento persistente:** Utilizar una base de datos MySQL que conserve los registros incluso tras el reinicio del contenedor.
- **Virtualización:** Orquestar los servicios con Docker Compose para facilitar despliegues consistentes en cualquier entorno.

Con base en estos objetivos, diseñamos un plan de acción estructurado:

- **Selección de tecnologías:** Flask, MySQL y Docker como componentes clave.
- **Diseño preliminar:** Definir la estructura del proyecto y la red Docker.
- **Inicialización de la base de datos:** Crear un script SQL (init.sql) para la tabla estudiantes.
- **Desarrollo iterativo:** Incorporar montaje de volúmenes para permitir reinicios rápidos sin pérdida de datos.

## 2.2 Estructura del proyecto.

La disposición de carpetas y archivos refleja la separación de responsabilidades:



## 2.3 Funcionalidades principales

- Formulario de captura: validaciones básicas de tipo (números, fechas, texto) y campos obligatorios.
- Listado en tiempo real: tras enviar, el estudiante aparece en la tabla de la derecha sin recargar manualmente.
- Persistencia: los datos se guardan en MySQL y sobreviven a ***docker-compose down -v*** gracias a un volumen dedicado.
- Pociones de despliegue: un solo **comando (*docker-compose up -d --build*)** levanta ambos servicios, configurados para arrancar en orden.

## 2.4 Guía de Código Documentado.

### Dockerfile

```
1 # Partimos de una imagen ligera de Python
2 FROM python:3.11-slim
3 # Directorio de trabajo
4 WORKDIR /app
5 # Copiamos y instalamos dependencias primero (capa cacheable)
6 COPY requirements.txt .
7 RUN pip install --no-cache-dir -r requirements.txt
8 # Copiamos el código de la aplicación
9 COPY app/ .
10 # Exponemos el puerto en el que corre Flask
11 EXPOSE 5000
12 # Comando de inicio
13 CMD ["python", "app.py"]
```

### docker-compose.yml

```
1 version: '3.8'
2 services:
3   db:
4     image: mysql:5.7
5     environment:
6       MYSQL_ROOT_PASSWORD: password
7       MYSQL_DATABASE: formulario
8     volumes:
9       - db_data:/var/lib/mysql
10
11   web:
12     build: .
13     depends_on:
14       db:
15         condition: service_healthy
16     ports:
17       - "8000:5000" # host:container
18     volumes:
19       - ./app:/app
20     environment:
21       FLASK_ENV: development
22
23 volumes:
24   db_data: {}
```

## app.py (rutas y lógica).

```
1 from flask import Flask, render_template, request, redirect, url_for
2 from controllers.database import Database
3
4 app = Flask(__name__)
5
6 @app.route('/')
7 def home():
8     # Obtenemos todos los estudiantes para mostrar en la tabla
9     estudiantes = Database().get_estudiantes()
10    return render_template('form.html', estudiantes=estudiantes)
11
12 @app.route('/submit', methods=['POST'])
13 def submit():
14     # Insertar estudiante en la BD con datos del formulario
15     data = request.form.to_dict()
16     Database().insert_estudiante(**data)
17     return redirect(url_for('success'))
18
19 @app.route('/success')
20 def success():
21     return render_template('success.html')
22
23 if __name__ == '__main__':
24     # Servidor de desarrollo escucha en todas las interfaces
25     app.run(host='0.0.0.0', port=5000, debug=True)
```

### 3. Manual de uso.

#### Requisitos previos.

- Docker Desktop instalado
- Conexión a Internet para descargar imágenes

#### 3.1 Clonar repositorio.

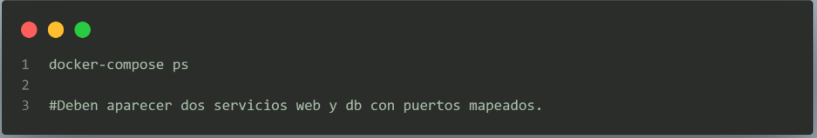
```
1 git clone https://github.com/Christian112b/FormToSQLContainers.git
2
3 cd FormToSQLContainers
```

#### 3.2 Construir y levantar servicios.

```
1 docker-compose down -v # elimina contenedores y volúmenes previos
2
3 docker-compose up -d --build # construye imágenes y arranca servicios
```

```
time="2025-05-25T10:08:16-06:00" level=warning msg="C:\\Users\\USUARIO DELL\\ProyectosGaman\\FormToSQLContainers\\do
ion' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 3/3
  ✓ Container formtosqlcontainers-web-1 Removed
  ✓ Container formtosqlcontainers-db-1 Removed
  ✓ Network formtosqlcontainers_default Removed
(base) PS C:\\Users\\USUARIO DELL\\ProyectosGaman\\FormToSQLContainers> docker-compose up -d --build
time="2025-05-25T10:08:25-06:00" level=warning msg="C:\\Users\\USUARIO DELL\\ProyectosGaman\\FormToSQLContainers\\do
ion' is obsolete, it will be ignored, please remove it to avoid potential confusion"
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 1.5s (12/12) FINISHED
=> [web internal] load build definition from Dockerfile
=> => transferring dockerfile: 547B
=> [web internal] load metadata for docker.io/library/python:3.11-slim
=> [web internal] load .dockerignore
=> => transferring context: 2B
=> [web 1/5] FROM docker.io/library/python:3.11-slim@sha256:dbf1de478a55d6763afaa39c2f3d7b54b25230614980276de5cacdd
=> => resolve docker.io/library/python:3.11-slim@sha256:dbf1de478a55d6763afaa39c2f3d7b54b25230614980276de5cacdd795
=> [web internal] load build context
=> CACHED [web 2/5] WORKDIR /app
=> CACHED [web 3/5] COPY requirements.txt .
=> CACHED [web 4/5] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [web 5/5] COPY app/ .
=> [web] exporting to image
=> => exporting layers
=> => exporting manifest sha256:438d045a21bddcd05cec1df5c38e5ea84aa509396af84bdaf25482a532f37d67
=> => exporting config sha256:60735c04d4d92a673837cbacfb0ehead426d641b2fdc224df88e34a0fff655b5
=> => naming to docker.io/library/formtosqlcontainers-web:latest
=> => unpacking to docker.io/library/formtosqlcontainers-web:latest
=> [web] resolving provenance for metadata file
[+] Running 4/4
  ✓ web Built
  ✓ Container formtosqlcontainers-db-1 Started
  ✓ Container formtosqlcontainers-web-1 Started
```

### 3.3 Verificar Construir y levantar servicios. Estado.



```
1 docker-compose ps
2
3 #Deben aparecer dos servicios web y db con puertos mapeados.
```

### 3.4 Acceder a la aplicación

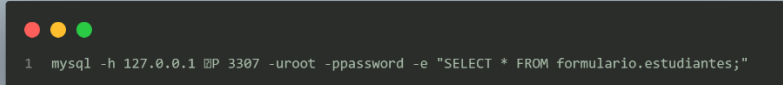
Abrir el navegador en:

<http://localhost:8000>

Completar el formulario y enviar.

### 3.5 Consultar la base de datos

Desde host Windows con cliente MySQL



```
1 mysql -h 127.0.0.1 -P 3307 -uroot -ppassword -e "SELECT * FROM formulario.estudiantes;"
```

### 3.6 Apagar servicios



```
1 docker-compose down
```