



Universidad Tecnológica  
del Norte de Guanajuato  
Organismo Público Descentralizado del Gobierno del Estado  
"Educación y progreso para la vida"



## **Automatización de Infraestructura Digital I**

### **Instrumento de Evaluación Unidad III**

### **Ingeniería en Redes Inteligentes y Ciberseguridad**

#### **Unidad III: Interfaces de Programación de Aplicaciones en la automatización de redes**

*Integrantes:* **Daniel Duarte Velázquez**

*Docente:* **Gabriel Barrón Rodríguez**

*Fecha de Entrega:* **28 de agosto de 2023**

## Tabla de contenido

<b>Creación de Microservicios en Python .....</b>	<b>3</b>
1. Creación de Docker.....	5
1.1. Aplicación Python con Flask.....	5
1.2. Base de datos .....	5
2. Desarrollo de microservicio de login.....	7
3. Creación de microservicio para la administración de usuarios .....	7
3.1. Agregar usuarios .....	7
3.2. Obtener usuarios .....	8
3.3. Actualizar usuarios .....	9
3.4. Eliminar usuarios .....	10
4. Microservicio para la administración de una entidad .....	11
4.1. Obtener alumnos .....	11
4.2. Agregar alumno .....	11
5. Microservicio para la administración de una segunda entidad.....	12
5.1. Agregar materia .....	12
5.2. Obtener materias.....	12
6. Comprobación del funcionamiento de la aplicación.....	13
Agregar usuario .....	13
Obtener usuarios .....	15
Actualizar usuario .....	15
Eliminar usuario .....	16
Agregar materia .....	17
Obtener Materias .....	18
Agregar alumno .....	18
Obtener Alumnos.....	20
7. Repositorio de GitHub.....	20
7.1. Jira .....	20
8. Manual de imagen y contenedor aplicación Python en Docker .....	21

# Creación de Microservicios en Python

## Descripción General

La aplicación está construida con el framework Flask y se conecta a una base de datos MySQL. Permite la autenticación de usuarios, así como la gestión de usuarios, alumnos y materias.

## Configuración

Las credenciales y configuraciones de la base de datos se almacenan en el diccionario **conexion\_config**. Es vital garantizar la seguridad de esta información en un entorno real.

## Funciones Auxiliares

1. **get\_db\_connection():**
  - Establece y devuelve una conexión a la base de datos MySQL.
2. **check\_password(hash\_password, user\_password):**
  - Compara una contraseña proporcionada (**user\_password**) con una versión cifrada (**hash\_password**).
  - Utiliza la biblioteca **bcrypt** para el proceso de comparación.
3. **is\_expiration\_valid(fecha\_expiracion\_str):**
  - Comprueba si una fecha de expiración dada como string es válida o no.
  - Devuelve **False** si la fecha es anterior a la fecha actual.
4. **es\_password\_valida(password):**
  - Evalúa la validez de una contraseña según ciertas reglas (al menos 8 caracteres, contiene al menos una letra mayúscula, una minúscula, un número y un carácter especial).

## Endpoints

1. **Inicio de Sesión (/login):**
  - Método: POST
  - Funcionalidad: Autentica a un usuario verificando sus credenciales contra la base de datos.
  - Datos Requeridos: **usuario, password**
2. **Usuarios (/usuarios):**
  - Método: GET
    - Funcionalidad: Lista todos los usuarios registrados.
  - Método: POST

- Funcionalidad: Registra un nuevo usuario.
  - Datos Requeridos: **correo, password, usuario**
3. **Actualizar Usuario (/usuarios/actualizar):**
- Método: PUT
  - Funcionalidad: Actualiza la información de un usuario.
  - Datos Requeridos: **id** y al menos uno de los siguientes: **correo, usuario, password o fecha\_expiracion**.
4. **Eliminar Usuario (/usuarios/eliminar):**
- Método: DELETE
  - Funcionalidad: Elimina un usuario.
  - Datos Requeridos: **correo**
5. **Alumnos (/alumnos):**
- Método: GET
    - Funcionalidad: Lista todos los alumnos.
  - Método: POST
    - Funcionalidad: Registra un nuevo alumno.
    - Datos Requeridos: **materia\_id, nombre, email, carrera, cuatrimestre, edad, numero\_control, promedio**
6. **Materias (/materias):**
- Método: GET
    - Funcionalidad: Lista todas las materias.
  - Método: POST
    - Funcionalidad: Registra una nueva materia.
    - Datos Requeridos: **usuario\_id, nombre\_materia, carrera, cantidad\_alumnos, area, periodo, maestro, edificio**

### Recomendaciones de Seguridad

1. Las contraseñas se cifran usando **bcrypt**, que es una práctica de seguridad recomendada. Sin embargo, la contraseña de la base de datos está en texto claro en el código, lo que no es seguro. Se recomienda utilizar variables de entorno o un administrador de secretos para esta información.
2. Siempre utilice declaraciones SQL parametrizadas para prevenir la inyección SQL.
3. En un entorno de producción, despliegue Flask detrás de un servidor web, como Nginx o Apache, y asegúrese de que **debug=False**.

## 1. Creación de Docker

### 1.1. Aplicación Python con Flask

```
1 # Importando las bibliotecas necesarias
2 from flask import Flask, jsonify, request
3 import re, bcrypt, mysql.connector
4 from datetime import datetime
5
6 # Inicializando la aplicación Flask
7 app = Flask(__name__)
8
9 # Configuración para conectar a la base de datos MySQL
10 conexion_config = {
11     'host': '192.168.214.112',
12     'user': 'root',
13     'password': 'linux',
14     'database': 'utng'
15 }
16
17 # Función para obtener una conexión a la base de datos
18 def get_db_connection():
19     return mysql.connector.connect(**conexion_config)
20
21 # Función para comprobar si una contraseña coincide con una contraseña cifrada
22 def check_password(hash_password, user_password):
23     return bcrypt.checkpw(user_password.encode('utf-8'), hash_password.encode('utf-8'))
24
25 # Función para validar si una fecha de expiración es válida
```

### 1.2. Base de datos

```
1 • DROP DATABASE IF EXISTS utng;
2 CREATE DATABASE IF NOT EXISTS utng;
3 USE utng;
4
5 CREATE TABLE IF NOT EXISTS usuarios (
6     id INT AUTO_INCREMENT PRIMARY KEY,
7     correo VARCHAR(30) NOT NULL,
8     password VARCHAR(200) NOT NULL,
9     usuario varchar(20) NOT null,
10     fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
11     fecha_expiration date
12 );
13
14 CREATE TABLE IF NOT EXISTS materias (
15     id INT AUTO_INCREMENT PRIMARY KEY,
16     usuario_id INT,
17     FOREIGN KEY (usuario_id) REFERENCES usuarios(id),
18     nombre_materia VARCHAR(50) NOT NULL,
19     carrera VARCHAR(50) NOT NULL,
20     cantidad_alumnos INT NOT NULL,
21     area varchar (40) NOT NULL,
22     periodo varchar (40) NOT NULL,
```

```
23     maestro varchar (40) NOT NULL,  
24     edificio varchar (10) NOT NULL  
25 );  
26  
27 CREATE TABLE IF NOT EXISTS alumnos (  
28     id INT AUTO_INCREMENT PRIMARY KEY,  
29     materia_id INT,  
30     FOREIGN KEY (materia_id) REFERENCES materias(id),  
31     nombre VARCHAR(50) NOT NULL,  
32     email VARCHAR(50) NOT NULL,  
33     carrera VARCHAR(50) NOT NULL,  
34     cuatrimestre INT NOT NULL,  
35     edad INT NOT NULL,  
36     numero_control INT NOT NULL UNIQUE,  
37     promedio FLOAT NOT NULL  
38 );
```

---

## 2. Desarrollo de microservicio de login

```
@app.post('/users/login')
def login():
    data = request.get_json()
    username = data.get('username')
    clave = data.get('clave')

    if not username or not clave:
        return {'error': 'Se requieren username y clave'}, 400

    cnx = mysql.connector.connect(**config)
    cursor = cnx.cursor(dictionary=True)
    query = "SELECT * FROM login WHERE username = %s"
    cursor.execute(query, (username,))
    user = cursor.fetchone()
    cursor.close()
    cnx.close()

    if user and user['clave'] == clave: # Compara la clave proporcionada con la almacenada
        return {'success': 'Login exitoso'}
    else:
        return {'error': 'Username o clave incorrectos'}, 401
```

## 3. Creación de microservicio para la administración de usuarios

### 3.1. Agregar usuarios

```
76 @app.route('/usuarios', methods=['POST'])
77 def add_usuarios():
78     data = request.get_json()
79     required_fields = ['correo', 'password', 'usuario']
80
81     if not all(key in data for key in required_fields):
82         return {'error': 'Faltan campos requeridos'}, 400
83
84     if not es_password_valida(data['password']):
85         return {'error': 'La contraseña es inválida'}, 400
86
87     # Validación de la fecha de expiración
88     if 'fecha_expiracion' in data and not is_expiration_valid(data['fecha_expiracion']):
89         return {'error': 'La fecha de expiración no es válida'}, 400
90
91     cnx = get_db_connection()
92     cursor = cnx.cursor(dictionary=True)
93
94     cursor.execute("SELECT correo FROM usuarios WHERE correo = %s", (data['correo'],))
95     existing_mail = cursor.fetchone()
96
97     if existing_mail:
98         cursor.close()
99         cnx.close()
100         return {'error': 'Este correo ó usuario ya está registrado'}, 409
101
102     hashed_pw = bcrypt.hashpw(data['password'].encode('utf-8'), bcrypt.gensalt())
103     data['password'] = hashed_pw.decode('utf-8')
104
105     insert_query = """
106     INSERT INTO usuarios (correo, password, usuario, fecha_expiracion)
107     VALUES (%s, %s, %s, %s)
108     """
109     cursor.execute(insert_query, (data['correo'], data['password'], data['usuario'], data.get('fecha_expiracion')))
110     cnx.commit()
111
112     cursor.close()
113     cnx.close()
114
115     return {'success': 'Registro agregado con éxito'}, 201
```

```

17 # Función para obtener una conexión a la base de datos
18 def get_db_connection():
19     return mysql.connector.connect(**conexion_config)
20
21 # Función para comprobar si una contraseña coincide con una contraseña cifrada
22 def check_password(hashed_password, user_password):
23     return bcrypt.checkpw(user_password.encode('utf-8'), hashed_password.encode('utf-8'))
24
25 # Función para validar s (parameter) fecha_expiracion_str: Any
26 def is_expiration_valid(fecha_expiracion_str):
27     try:
28         fecha_expiracion = datetime.strptime(fecha_expiracion_str, '%Y-%m-%d').date()
29         if fecha_expiracion < datetime.now().date():
30             return False
31         return True
32     except ValueError:
33         return False
34
35 def es_password_valida(password):
36     if len(password) < 8:
37         return False
38     if not re.match(r"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$", password):
39         return False
40     return True

```

### 3.2. Obtener usuarios

```

63 @app.route('/usuarios', methods=['GET'])
64 def get_usuarios():
65     cnx = get_db_connection()
66     cursor = cnx.cursor(dictionary=True)
67
68     cursor.execute("SELECT id, correo, usuario, fecha_registro, fecha_expiracion FROM usuarios")
69     result = cursor.fetchall()
70
71     cursor.close()
72     cnx.close()

```



### 3.3. Actualizar usuarios

```
117 @app.route('/usuarios/actualizar', methods=['PUT'])
118 def update_usuario():
119     data = request.get_json()
120
121     # Asegúrate de que el ID del usuario se proporcione en el cuerpo de la solicitud
122     if 'id' not in data:
123         return {'error': 'Falta el ID del usuario en el cuerpo de la solicitud'}, 400
124
125     usuario_id = data['id']
126
127     updates = []
128     values = []
129
130     if 'correo' in data:
131         updates.append("correo = %s")
132         values.append(data['correo'])
133
134     if 'usuario' in data:
135         updates.append("usuario = %s")
136         values.append(data['usuario'])
137
138     if 'fecha_expiracion' in data:
139         updates.append("fecha_expiracion = %s")
140         values.append(data['fecha_expiracion'])
141
142     if 'password' in data:
143         hashed_pw = bcrypt.hashpw(data['password'].encode('utf-8'), bcrypt.gensalt())
144         updates.append("password = %s")
145         values.append(hashed_pw.decode('utf-8'))
146
147     if not updates:
148         return {'error': 'No hay cambios para actualizar'}, 400
149
150     sql_update_query = "UPDATE usuarios SET " + ", ".join(updates) + " WHERE id = %s"
151     values.append(usuario_id)
152
153     cnx = get_db_connection()
154     cursor = cnx.cursor(dictionary=True)
155     cursor.execute(sql_update_query, values)
156     cnx.commit()
157
158     if cursor.rowcount == 0:
159         cursor.close()
160         cnx.close()
161         return {'error': 'Usuario no encontrado'}, 404
162
163     cursor.close()
164     cnx.close()
165
166     return {'success': 'Usuario actualizado con éxito'}, 200
```

### 3.4. Eliminar usuarios

```
168 @app.route('/usuarios/eliminar', methods=['DELETE'])
169 def delete_usuario():
170     data = request.get_json()
171
172     if 'correo' not in data:
173         return {'error': 'Falta el campo correo'}, 400
174
175     cnx = get_db_connection()
176     cursor = cnx.cursor(dictionary=True)
177
178     cursor.execute("SELECT id FROM usuarios WHERE correo = %s", (data['correo'],))
179     user = cursor.fetchone()
180
181     if not user:
182         return {'error': 'Usuario con el correo proporcionado no encontrado'}, 404
183
184     cursor.execute("DELETE FROM usuarios WHERE correo = %s", (data['correo'],))
185     cnx.commit()
186
187     cursor.close()
188     cnx.close()
189
190     return {'success': 'Usuario eliminado con éxito'}, 200
```

## 4. Microservicio para la administración de una entidad

### 4.1. Obtener alumnos

```
192 @app.route('/alumnos', methods=['GET'])
193 def get_alumnos():
194     cnx = get_db_connection()
195     cursor = cnx.cursor(dictionary=True)
196
197     cursor.execute("SELECT nombre, email, carrera, cuatrimestre, edad, numero_control, promedio FROM alumnos")
198     result = cursor.fetchall()
199
200     cursor.close()
201     cnx.close()
202
203     return jsonify(result), 200
```

### 4.2. Agregar alumno

```
205 @app.route('/alumnos', methods=['POST'])
206 def add_alumno():
207     data = request.get_json()
208
209     required_fields = ['materia_id', 'nombre', 'email', 'carrera', 'cuatrimestre', 'edad', 'numero_control', 'promedio']
210
211     if not all(key in data for key in required_fields):
212         return {'error': 'Faltan campos requeridos'}, 400
213
214     # Validación de campos vacíos
215     for field in required_fields:
216         if not data[field]:
217             return {'error': f'El campo {field} no puede estar vacío'}, 400
218
219     cnx = get_db_connection()
220     cursor = cnx.cursor(dictionary=True)
221
222     # Verificamos si el numero_control ya existe
223     cursor.execute("SELECT numero_control FROM alumnos WHERE numero_control = %s", (data['numero_control'],))
224     existing_alumno = cursor.fetchone()
225
226     if existing_alumno:
227         cursor.close()
228         cnx.close()
229         return {'error': 'El numero de control ya está registrado'}, 409
230
231     # Verificamos si el materia_id existe en la tabla materias
232     cursor.execute("SELECT id FROM materias WHERE id = %s", (data['materia_id'],))
233     existing_materia = cursor.fetchone()
234
235     if not existing_materia:
236         cursor.close()
237         cnx.close()
238         return {'error': 'El materia_id no existe'}, 404
239
240     # Si ambas validaciones pasan, procedemos a insertar el alumno
241     cursor.execute("INSERT INTO alumnos (materia_id, nombre, email, carrera, cuatrimestre, edad, numero_control, promedio) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)",
242                  (data['materia_id'], data['nombre'], data['email'], data['carrera'], data['cuatrimestre'], data['edad'], data['numero_co
243     cnx.commit()
244
245     cursor.close()
246     cnx.close()
247
248     return {'success': 'Alumno agregado con éxito'}, 201
```

## 5. Microservicio para la administración de una segunda entidad

### 5.1. Agregar materia

```
264 @app.route('/materias', methods=['POST'])
265 def add_materia():
266     data = request.get_json()
267
268     required_fields = ['usuario_id', 'nombre_materia', 'carrera', 'cantidad_alumnos', 'area', 'periodo', 'maestro', 'edificio']
269
270     if not all(key in data for key in required_fields):
271         return {'error': 'Faltan campos requeridos'}, 400
272
273     # Validación de campos vacíos
274     for field in required_fields:
275         if not data[field]:
276             return {'error': f'El campo {field} no puede estar vacío'}, 400
277
278     cnx = get_db_connection()
279     cursor = cnx.cursor(dictionary=True)
280
281     # Verificamos si el usuario_id existe en la tabla usuarios
282     cursor.execute("SELECT id FROM usuarios WHERE id = %s", (data['usuario_id'],))
283     existing_user = cursor.fetchone()
284
285     if not existing_user:
286         cursor.close()
287         cnx.close()
288         return {'error': 'El usuario_id no existe'}, 404
289
290     # Si la validación pasa, procedemos a insertar la materia
291     cursor.execute("INSERT INTO materias (usuario_id, nombre_materia, carrera, cantidad_alumnos, area, periodo, maestro, edificio) VALUES (
292         | | | | | | | |
293         (data['usuario_id'], data['nombre_materia'], data['carrera'], data['cantidad_alumnos'], data['area'], data['periodo'], d
294     cnx.commit()
295
296     cursor.close()
297     cnx.close()
298
299     return {'success': 'Materia agregada con éxito'}, 201
```

### 5.2. Obtener materias

```
251 @app.route('/materias', methods=['GET'])
252 def get_materias():
253     cnx = get_db_connection()
254     cursor = cnx.cursor(dictionary=True)
255
256     cursor.execute("SELECT nombre_materia, carrera, cantidad_alumnos, area, periodo, maestro, edificio FROM materias")
257     result = cursor.fetchall()
258
259     cursor.close()
260     cnx.close()
261
262     return jsonify(result), 200
```

## 6. Comprobación del funcionamiento de la aplicación

### Agregar usuario

POST http://127.0.0.1:5000/usuarios

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "correo": "user4@gmail.com",
3   "password": "Linux@123",
4   "usuario": "user4",
5   "fecha_expiracion": "2023-08-26"
6 }
```

Body Cookies Headers (5) Test Results 400 BAD REQUEST 19 ms 237 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": "La fecha de expiración no es válida"
3 }
```

Automatización / Add\_Usuarios Save

POST http://127.0.0.1:5000/usuarios Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "correo": "user4@gmail.com",
3   "password": "Linux123",
4   "usuario": "user4",
5   "fecha_expiracion": "2023-08-29"
6 }
```

Body Cookies Headers (5) Test Results 400 BAD REQUEST 9 ms 227 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": "La contraseña es inválida"
3 }
```

HTTP

Automatización / Add\_Usuarios

Save

POST

▼

http://127.0.0.1:5000/usuarios

Send

▼

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

Cookies

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ▼

Beautify

1

2

3

4

5

6

```
{
  "correo": "user5@gmail.com",
  "password": "Linux@123",
  "usuario": "user4",
  "fecha_expiracion": "2023-08-29"
}
```

Body

Cookies

Headers (5)

Test Results

409 CONFLICT

157 ms

239 B

Save as Example

⋮

Pretty

Raw

Preview

Visualize

JSON ▼

1

2

3

```
{
  "error": "Este correo ó usuario ya está registrado"
}
```

POST

▼

http://127.0.0.1:5000/usuarios

Send

▼

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

Cookies

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ▼

Beautify

1

2

3

4

5

6

```
{
  "correo": "danyd@danet.com",
  "password": "Linux@123",
  "usuario": "danyd",
  "fecha_expiracion": "2023-08-29"
}
```

Body

Cookies

Headers (5)

Test Results

201 CREATED

325 ms

222 B

Save as Example

⋮

Pretty

Raw

Preview

Visualize

JSON ▼

1

2

3

```
{
  "success": "Registro agregado con éxito"
}
```

## Obtener usuarios

GET http://127.0.0.1:5000/usuarios Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK 29 ms 745 B Save as Example

Pretty Raw Preview Visualize JSON

```
16 {
17   "correo": "danyd@danet.com",
18   "fecha_expiracion": "Tue, 29 Aug 2023 00:00:00 GMT",
19   "fecha_registro": "Mon, 28 Aug 2023 04:19:24 GMT",
20   "id": 4,
21   "usuario": "danyd"
22 }
23 ]
```

## Actualizar usuario

PUT http://127.0.0.1:5000/usuarios/actualizar Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "id": 4,
3   "usuario": "dduarte",
4   "correo": "dduarte@danet.com"
5 }
```

Body Cookies Headers (5) Test Results 200 OK 42 ms 219 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": "Usuario actualizado con éxito"
3 }
```

## Eliminar usuario

DELETE

http://127.0.0.1:5000/usuarios/eliminar

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettingsCookies

noneform-data x-www-form-urlencodedrawbinaryGraphQLJSON

Beautify

1{"correo": "dduarte@danet.com"}

BodyCookiesHeaders (5)Test Results200 OK27 ms217 BSave as Example

PrettyRawPreviewVisualizeJSON

1{"success": "Usuario eliminado con éxito"}

GET

http://127.0.0.1:5000/usuarios

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettingsCookies

Query Params

	Key	Value	Description	Bulk Edit
--	-----	-------	-------------	-----------

BodyCookiesHeaders (5)Test Results200 OK28 ms555 BSave as Example

PrettyRawPreviewVisualizeJSON

2{  
3  "correo": "gbarron@utng.edu.mx",  
4  "fecha\_expiracion": "Sat, 26 Aug 2023 00:00:00 GMT",  
5  "fecha\_registro": "Fri, 25 Aug 2023 13:49:22 GMT",  
6  "id": 1,  
7  "usuario": "gbarron"  
8 },  
9 {  
10  "correo": "user5@gmail.com",  
11  "fecha\_expiracion": "Sat, 26 Aug 2023 00:00:00 GMT",  
12  "fecha\_registro": "Fri, 25 Aug 2023 14:16:21 GMT",  
13  "id": 3,  
14  "usuario": "user5"  
15 }  
16 }



## Agregar materia

**POST**  http://127.0.0.1:5000/materias **Send** 

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

## Cookies

☐ none
 ☐ form-data
 ☐ x-www-form-urlencoded
 ☒ raw
 ☐ binary
 ☐ GraphQL
 **JSON**

## Beautify

```
1  {
2      "usuario_id":5,
3      "nombre_materia":"",
4      "carrera":"Redes Inteligentes y Ciberseguridad",
5      "cantidad_alumnos":11,
6      "area":"TIC",
7      "periodo":"Mayo-Agosto 2023",
8      "maestro":"Gabriel Barron",
9      "edificio":"F"
10 }
```

Body Cookies Headers (5) Test Results

400 BAD REQUEST 6 ms 241 B Save as Example

Pretty Raw Preview Visualize JSON 

```
1 {
2   "error": "El campo nombre_materia no puede estar vacío"
3 }
```

**POST**  http://127.0.0.1:5000/materias **Send** 

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

## Cookies

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ▼

## Beautify

```
1 {
2   ...-"usuario_id":5,
3   ...-"nombre_materia":"Automatizacion",
4   ...-"carrera":"Redes Inteligentes y Ciberseguridad",
5   ...-"cantidad_alumnos":11,
6   ...-"area":"TIC",
7   ...-"periodo":"Mayo-Agosto 2023",
8   ...-"maestro":"Gabriel Barron",
9   ...-"edificio":"F"
10 }
```

Body Cookies Headers (5) Test Results

201 CREATED 23 ms 221 B Save as Example

Pretty Raw Preview Visualize JSON 

```
1 {
2   "success": "Materia agregada con éxito"
3 }
```

## Obtener Materias

GET

http://localhost:5000/materias

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a body

Body Cookies Headers (5) Test Results 200 OK 28 ms 1.05 KB Save as Example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "area": "TIC",
4     "cantidad_alumnos": 11,
5     "carrera": "Redes dd y Ciberseguridad",
6     "edificio": "F",
7     "maestro": "Gabriel Barron",
8     "nombre_materia": "Automatizacion",
9     "periodo": "Mayo-Agosto 2023"
10  },
11  ]
```

## Agregar alumno

POST

http://localhost:5000/alumnos

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {
2   "materia_id": 5,
3   "carrera": "Redes Inteligentes y Ciberseguridad",
4   "cuatrimestre": 9,
5   "edad": 21,
6   "email": "alumno5@gmail.com",
7   "nombre": "",
8   "numero_control": 1220100317,
9   "promedio": 8.0
10 }
```

Body Cookies Headers (5) Test Results 400 BAD REQUEST 7 ms 233 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": "El campo nombre no puede estar vacio"
3 }
```

POST http://localhost:5000/alumnos Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   .... "materia_id":5,
3   .... "carrera": "Redes Inteligentes y Ciberseguridad",
4   .... "cuatrimestre":9,
5   .... "edad":21,
6   .... "email": "alumno5@gmail.com",
7   .... "nombre": "Daniel Duarte",
8   .... "numero_control": 1220100317,
9   .... "promedio": 8.0
10  }
```

body Cookies Headers (5) Test Results 404 NOT FOUND 40 ms 213 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": "El materia_id no existe"
3 }
```

POST http://localhost:5000/alumnos Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   .... "materia_id":4,
3   .... "carrera": "Redes Inteligentes y Ciberseguridad",
4   .... "cuatrimestre":9,
5   .... "edad":21,
6   .... "email": "alumno5@gmail.com",
7   .... "nombre": "Daniel Duarte",
8   .... "numero_control": 1220100317,
9   .... "promedio": 8.0
10  }
```

body Cookies Headers (5) Test Results 201 CREATED 111 ms 220 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": "Alumno agregado con éxito"
3 }
```

## Obtener Alumnos

GET ▼ http://localhost:5000/alumnos Send ▼

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

---

Body Cookies Headers (5) Test Results 200 OK 48 ms 819 B Save as Example ⋮

Pretty Raw Preview Visualize JSON ▼ 🔍

```
19  },
20  {
21    "carrera": "Redes Inteligentes y Ciberseguridad",
22    "cuatrimestre": 9,
23    "edad": 21,
24    "email": "alumno5@gmail.com",
25    "nombre": "Daniel Duarte",
26    "numero_control": 1220100317,
27    "promedio": 8.0
28  }
29 ]
```

## 7. Repositorio de GitHub

<https://github.com/duartdany/Recuperacion-1>

### 7.1. Jira

<https://automaredesgric3091.atlassian.net/jira/software/projects/AUT/boards/4/timeline>

Video de explicación

<https://drive.google.com/file/d/1oYGAItZVpK5FQTl5n-euN4iHcAK5iITf/view?usp=sharing>

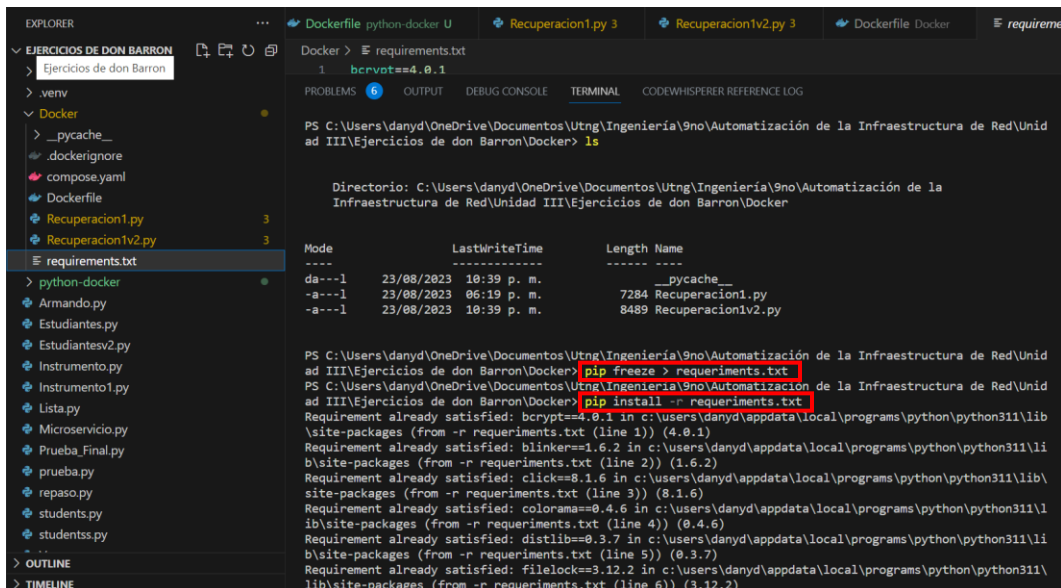
## 8. Manual de imagen y contenedor aplicación Python en Docker

**Paso 1:** Instalar un entorno virtual para trabajar. *Pip install virtualenv.*

```
127.0.0.1 - - [23/Aug/2023 22:40:34] "GET /usuarios HTTP/1.1" 200 -
127.0.0.1 - - [23/Aug/2023 22:41:40] "GET /usuarios/eliminar HTTP/1.1" 405 -
PS C:\Users\danyd\OneDrive\Documentos\Utng\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> pip install virtualenv
Collecting virtualenv
  Obtaining dependency information for virtualenv from https://files.pythonhosted.org/packages/17/8d/6989e
5dcd812520cbf9f31be2b08643ae3a895586601bbab501df8ed6e54/virtualenv-20.24.3-py3-none-any.whl.metadata
  Downloading virtualenv-20.24.3-py3-none-any.whl.metadata (4.5 kB)
Collecting distlib<1,>=0.3.7 (from virtualenv)
  Obtaining dependency information for distlib<1,>=0.3.7 from https://files.pythonhosted.org/packages/43/a
0/9ba967fdbd55293bacfc1507f58e316f740a3b231fc00e3d86dc39bc185a/distlib-0.3.7-py2.py3-none-any.whl.metadata
  Downloading distlib-0.3.7-py2.py3-none-any.whl.metadata (5.1 kB)
Collecting filelock<4,>=3.12.2 (from virtualenv)
  Obtaining dependency information for filelock<4,>=3.12.2 from https://files.pythonhosted.org/packages/00
/45/ec3407adf6f6b5bf867a4462b2b0af27597a26bd3cd6e2534cb6ab029938/filelock-3.12.2-py3-none-any.whl.metadata
  Downloading filelock-3.12.2-py3-none-any.whl.metadata (2.7 kB)
Collecting platformdirs<4,>=3.9.1 (from virtualenv)
  Obtaining dependency information for platformdirs<4,>=3.9.1 from https://files.pythonhosted.org/packages
/14/51/fe5a0d6ea589f0d4a1b97824fb518962ad48b27cd346cdcf2405187997a/platformdirs-3.10.0-py3-none-any.whl.m
etadata
  Downloading platformdirs-3.10.0-py3-none-any.whl.metadata (11 kB)
Downloading virtualenv-20.24.3-py3-none-any.whl (3.0 MB)
3.0/3.0 MB 628.0 kB/s eta 0:00:00
Downloading distlib-0.3.7-py2.py3-none-any.whl (468 kB)
468.9/468.9 kB 611.5 kB/s eta 0:00:00
Downloading filelock-3.12.2-py3-none-any.whl (10 kB)
Downloading platformdirs-3.10.0-py3-none-any.whl (17 kB)
Installing collected packages: distlib, platformdirs, filelock, virtualenv
Successfully installed distlib-0.3.7 filelock-3.12.2 platformdirs-3.10.0 virtualenv-20.24.3
PS C:\Users\danyd\OneDrive\Documentos\Utng\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> ls
```

**Paso 2.1:** Crear un archivo que contenga las dependencias necesarias de flask. *Pip freeze > requirements.txt.*

**Paso 2.2:** Instalar las dependencias que se acaban de instalar. *Pip install -r requirements.txt.*

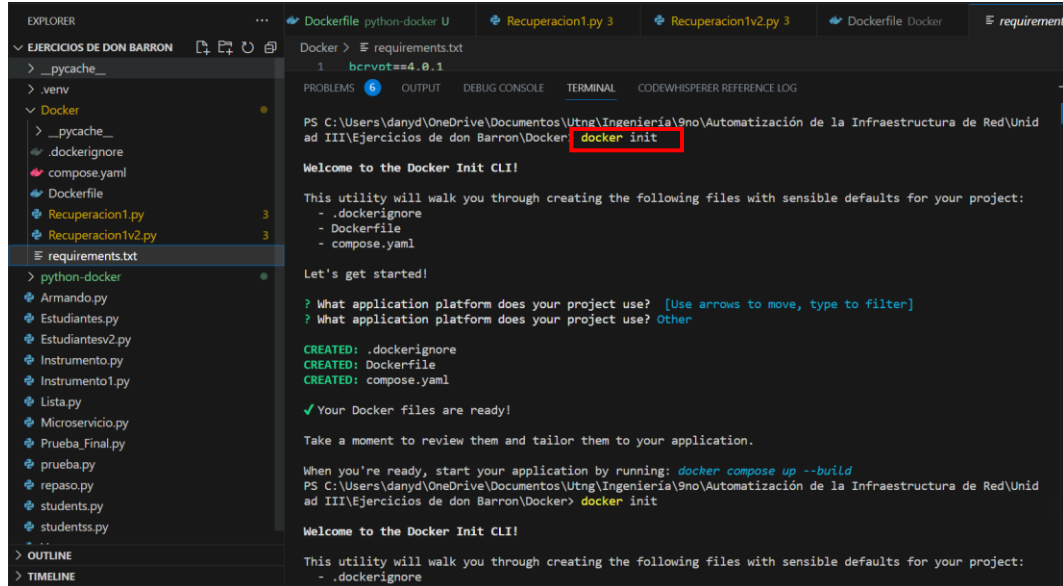


The screenshot shows a Windows File Explorer window on the left, displaying the contents of a folder named 'EJERCICIOS DE DON BARRON'. The folder contains several files, including 'requirements.txt', 'python-docker', 'Armando.py', 'Estudiantes.py', 'Estudiantesv2.py', 'Instrumento.py', 'Instrumento1.py', 'Lista.py', 'Microservicio.py', 'Prueba\_Final.py', 'prueba.py', 'repaso.py', 'students.py', and 'studentss.py'. The 'requirements.txt' file is selected.

On the right, a terminal window is open, showing the output of the 'pip install virtualenv' command. The output indicates that the virtual environment was successfully installed. Below this, the terminal shows the command 'pip freeze > requirements.txt' being executed, which creates the 'requirements.txt' file. The terminal also shows the command 'pip install -r requirements.txt' being executed, which installs the dependencies listed in the 'requirements.txt' file. The output of this command shows that the requirements are already satisfied, indicating that the dependencies are already installed.

```
PS C:\Users\danyd\OneDrive\Documentos\Utng\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> pip freeze > requirements.txt
PS C:\Users\danyd\OneDrive\Documentos\Utng\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> pip install -r requirements.txt
Requirement already satisfied: bcrypt==4.0.1 in c:\users\danyd\appdata\local\programs\python\python311\lib
\site-packages (from -r requirements.txt (line 1)) (4.0.1)
Requirement already satisfied: blinker==1.6.2 in c:\users\danyd\appdata\local\programs\python\python311\li
b\site-packages (from -r requirements.txt (line 2)) (1.6.2)
Requirement already satisfied: click==8.1.6 in c:\users\danyd\appdata\local\programs\python\python311\lib\
site-packages (from -r requirements.txt (line 3)) (8.1.6)
Requirement already satisfied: colorama==0.4.6 in c:\users\danyd\appdata\local\programs\python\python311\l
ib\site-packages (from -r requirements.txt (line 4)) (0.4.6)
Requirement already satisfied: distlib==0.3.7 in c:\users\danyd\appdata\local\programs\python\python311\li
b\site-packages (from -r requirements.txt (line 5)) (0.3.7)
Requirement already satisfied: filelock==3.12.2 in c:\users\danyd\appdata\local\programs\python\python311\
lib\site-packages (from -r requirements.txt (line 6)) (3.12.2)
```

**Paso 3:** Iniciar el Docker con el comando *Docker init*.



```
EXPLORER
  EJERCICIOS DE DON BARRON
    > __pycache__
    > .venv
    > Docker
      > __pycache__
      > .dockerignore
      > compose.yaml
      > Dockerfile
      > Recuperacion1.py
      > Recuperacion1v2.py
      > requirements.txt
    > python-docker
      > Armando.py
      > Estudiantes.py
      > Estudiantesv2.py
      > Instrumento.py
      > Instrumento1.py
      > Lista.py
      > Microservicio.py
      > Prueba_Final.py
      > prueba.py
      > repaso.py
      > students.py
      > students.py
    > OUTLINE
    > TIMELINE

Docker > requirements.txt
1  bcrvot==4.0.1

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG

PS C:\Users\danyd\OneDrive\Documents\Utn\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> docker init

Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
- .dockerignore
- Dockerfile
- compose.yaml

Let's get started!

? What application platform does your project use? [Use arrows to move, type to filter]
? What application platform does your project use? Other

CREATED: .dockerignore
CREATED: Dockerfile
CREATED: compose.yaml

✓ Your Docker files are ready!

Take a moment to review them and tailor them to your application.

When you're ready, start your application by running: docker compose up --build
PS C:\Users\danyd\OneDrive\Documents\Utn\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> docker init

Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
- .dockerignore
```

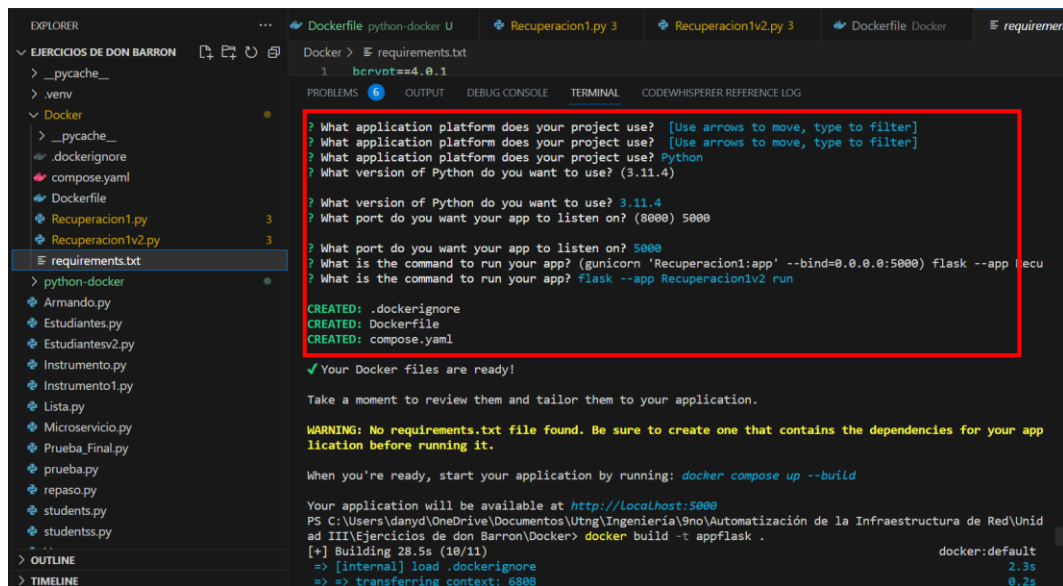
**Paso 4:** Contestar las preguntas de la siguiente forma:

*What application platform does your project use?* *Python*

*What version of python do you want to use?* *3.11.4*

*What port do you want your app to listen on?* *5000*

*What is the comand to run your app?* *Flask -app (name) run*



```
EXPLORER
  EJERCICIOS DE DON BARRON
    > __pycache__
    > .venv
    > Docker
      > __pycache__
      > .dockerignore
      > compose.yaml
      > Dockerfile
      > Recuperacion1.py
      > Recuperacion1v2.py
      > requirements.txt
    > python-docker
      > Armando.py
      > Estudiantes.py
      > Estudiantesv2.py
      > Instrumento.py
      > Instrumento1.py
      > Lista.py
      > Microservicio.py
      > Prueba_Final.py
      > prueba.py
      > repaso.py
      > students.py
      > students.py
    > OUTLINE
    > TIMELINE

Docker > requirements.txt
1  bcrvot==4.0.1

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG

? What application platform does your project use? [Use arrows to move, type to filter]
? What application platform does your project use? [Use arrows to move, type to filter]
? What application platform does your project use? Python
? What version of Python do you want to use? (3.11.4)

? What version of Python do you want to use? 3.11.4
? What port do you want your app to listen on? (8000) 5000

? What port do you want your app to listen on? 5000
? What is the command to run your app? (gunicorn 'Recuperacion1:app' --bind=0.0.0.0:5000) flask --app recu
? What is the comand to run your app? flask --app Recuperacion1v2 run

CREATED: .dockerignore
CREATED: Dockerfile
CREATED: compose.yaml

✓ Your Docker files are ready!

Take a moment to review them and tailor them to your application.

WARNING: No requirements.txt file found. Be sure to create one that contains the dependencies for your app
lication before running it.

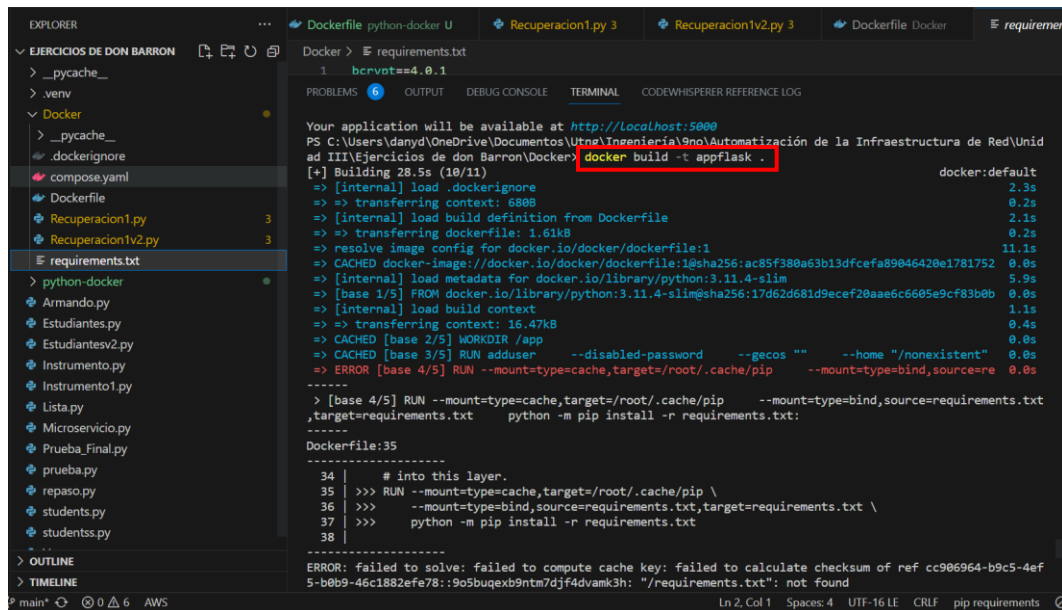
When you're ready, start your application by running: docker compose up --build

Your application will be available at http://localhost:5000
PS C:\Users\danyd\OneDrive\Documents\Utn\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> docker build -t appflask .

[+] Building 28.5s (10/11)
=> [internal] load .dockerignore 2.3s
=> => transferring context: 680B 0.2s
```

Una vez finalizado se crearan tres archivos, .dockerignore, Dockerfile y compose.yaml

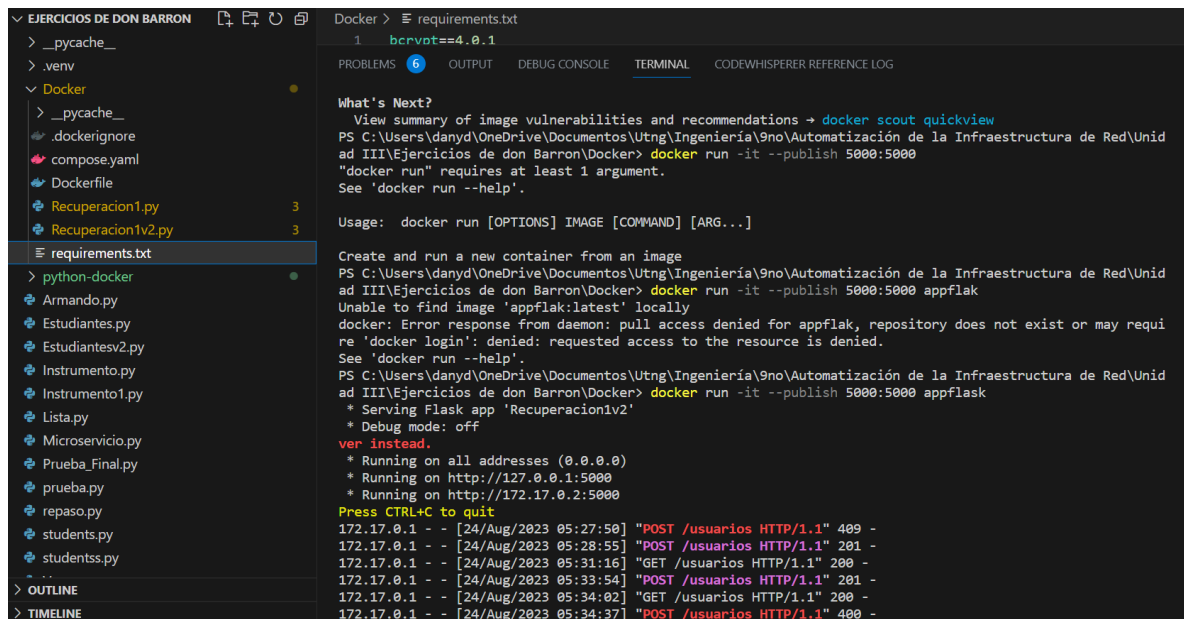
**Paso 5:** Construir la imagen con el siguiente comando: *Docker build -t [name].*



```
1 bcrvot==4.0.1

Your application will be available at http://localhost:5000
PS C:\Users\danyd\OneDrive\Documentos\Utn\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> docker build -t appflask .
[+] Building 28.5s (10/11)                                docker:default
=> [internal] load .dockerignore                          2.3s
=> => transferring context: 680B                          0.2s
=> [internal] load build definition from Dockerfile        2.1s
=> => transferring dockerfile: 1.61kB                      0.2s
=> resolve image config for docker.io/docker/dockerfile:1 11.1s
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752 0.0s
=> [internal] load metadata for docker.io/library/python:3.11.4-slim 5.9s
=> [base 1/5] FROM docker.io/library/python:3.11.4-slim@sha256:17d62d681d9ecf20aaec6605e9cf83bb0 0.0s
=> [internal] load build context                          1.1s
=> => transferring context: 16.47kB                       0.4s
=> CACHED [base 2/5] WORKDIR /app                        0.0s
=> CACHED [base 3/5] RUN adduser --disabled-password --gecos "" --home "/nonexistent" 0.0s
=> ERROR [base 4/5] RUN --mount=type=cache,target=/root/.cache/pip --mount=type=bind,source=re 0.0s
-----
> [base 4/5] RUN --mount=type=cache,target=/root/.cache/pip --mount=type=bind,source=requirements.txt
target=requirements.txt python -m pip install -r requirements.txt:
-----
Dockerfile:35
-----
34 | # into this layer.
35 | >>> RUN --mount=type=cache,target=/root/.cache/pip \
36 | >>> --mount=type=bind,source=requirements.txt,target=requirements.txt \
37 | >>> python -m pip install -r requirements.txt
38 |
-----
ERROR: failed to solve: failed to compute cache key: failed to calculate checksum of ref cc906964-b9c5-4ef
5-b0b9-46c1882efe78:905buqexb9ntm7djf4dvamk3h: "/requirements.txt": not found
Ln 2, Col 1 Spaces: 4 UTF-16 LE CRLF pip requirements
```

**Paso 6:** Correr la imagen creada con el comando: *Docker run -it - --publish 5000:5000 [name]*



```
1 bcrvot==4.0.1

What's Next?
View summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\danyd\OneDrive\Documentos\Utn\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> docker run -it --publish 5000:5000
"docker run" requires at least 1 argument.
See 'docker run --help'.

Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Create and run a new container from an image
PS C:\Users\danyd\OneDrive\Documentos\Utn\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> docker run -it --publish 5000:5000 appflask
Unable to find image 'appflask:latest' locally
docker: Error response from daemon: pull access denied for appflask, repository does not exist or may requi
re 'docker login': denied: requested access to the resource is denied.
See 'docker run --help'.
PS C:\Users\danyd\OneDrive\Documentos\Utn\Ingenieria\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\Docker> docker run -it --publish 5000:5000 appflask
* Serving Flask app 'Recuperacion1v2'
* Debug mode: off
ver instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [24/Aug/2023 05:27:50] "POST /usuarios HTTP/1.1" 409 -
172.17.0.1 - - [24/Aug/2023 05:28:55] "POST /usuarios HTTP/1.1" 201 -
172.17.0.1 - - [24/Aug/2023 05:31:16] "GET /usuarios HTTP/1.1" 200 -
172.17.0.1 - - [24/Aug/2023 05:33:54] "POST /usuarios HTTP/1.1" 201 -
172.17.0.1 - - [24/Aug/2023 05:34:02] "GET /usuarios HTTP/1.1" 200 -
172.17.0.1 - - [24/Aug/2023 05:34:37] "POST /usuarios HTTP/1.1" 400 -
```

## 9. Manual para desplegar una aplicación de GitHub en un servidor local con Ubuntu 22.04

### 1. Actualizar el sistema:

Antes de instalar cualquier paquete, es una buena práctica actualizar la lista de paquetes y el propio sistema.

```
sudo apt update && sudo apt upgrade -y
```

### 2. Instalar Node.js y npm:

Ubuntu 22.04 podría tener una versión más reciente de Node.js en los repositorios oficiales. Instala Node.js y npm con:

```
sudo apt install -y nodejs npm
```

### 3. Clonar el repositorio de GitHub:

Primero, navega al directorio donde deseas clonar el repositorio y luego clona el repositorio usando git. Si no tienes git instalado, instálalo con **sudo apt install git**.

```
cd /ruta/del/directorio git clone https://github.com/USUARIO/REPOSITORIO.git
```

Reemplaza **USUARIO/REPOSITORIO** con la ruta correcta de tu repositorio.

### 4. Instalar las dependencias:

Dentro del directorio clonado, es probable que exista un archivo **package.json** que contiene una lista de dependencias. Usa **npm** para instalar estas dependencias.

```
cd REPOSITORIO npm install
```

### 5. Ejecutar la aplicación:

Si todo ha ido bien hasta ahora, deberías poder iniciar tu aplicación de Express con el comando **start** definido en tu **package.json** o directamente con **node**.

# Si tienes un script de inicio en package.json: `npm start` # O, si no tienes un script de inicio específico: `node nombreDelArchivo.js`

Reemplaza **nombreDelArchivo.js** con el nombre del archivo que inicia tu aplicación (por ejemplo, **app.js**, **server.js**, etc.).

### 6. Acceder a la aplicación:



Si tu aplicación de Express se está ejecutando en el puerto 3000 (es un puerto común para aplicaciones Express), simplemente abre tu navegador web y dirígete a:

arduinoCopy code

`http://localhost:3000`

¡Eso es todo! Ahora deberías tener tu aplicación de GitHub ejecutándose en tu servidor local con Ubuntu 22.04.

### Notas adicionales:

- Asegúrate de verificar cualquier configuración específica que pueda requerir tu aplicación, como variables de entorno o conexiones a bases de datos.
- Si planeas desplegar esta aplicación en un entorno de producción, considera usar un administrador de procesos como **pm2** para mantener tu aplicación en ejecución incluso si ocurre un fallo o si el sistema se reinicia.

## 10. Manual para desplegar una aplicación de GitHub en un servidor local con Ubuntu 22.04

### 1. Crear una instancia EC2:

1. Inicia sesión en la consola de AWS y ve al servicio EC2.
2. Haz clic en "Launch Instance" para crear una nueva instancia.
3. Selecciona "Ubuntu Server 22.04 LTS".
4. Elige el tipo de instancia (por ejemplo, t2.micro que está dentro del nivel gratuito).
5. Configura los detalles de la instancia como sea necesario.
6. Añade reglas de seguridad para abrir los puertos necesarios:
  - SSH (Puerto 22) para la conexión.
  - HTTP (Puerto 80) o cualquier otro puerto en el que esté escuchando tu aplicación.
7. Revisa y lanza la instancia.
8. Selecciona un par de claves existente o crea uno nuevo para conectar a la instancia. Descarga y guarda este archivo **.pem** en un lugar seguro.

## 2. Conectarte a tu instancia EC2:

Utiliza SSH para conectarte a tu instancia desde tu terminal local:

bashCopy code

```
chmod 400 /ruta/a/tu/archivo.pem ssh -i /ruta/a/tu/archivo.pem ubuntu@direccion-IP-de-tu-  
instancia
```

## 3. Configura el entorno:

Una vez conectado a la instancia EC2, puedes seguir los pasos anteriores para configurar tu entorno Ubuntu (instalar Node.js, npm, git, etc.)

## 4. Clonar y preparar tu aplicación:

Igual que antes, clona tu repositorio de GitHub y prepara tu aplicación instalando las dependencias con **npm install**.

## 5. Configurar un servidor proxy (Opcional):

Si tu aplicación no escucha en el puerto 80 (o 443 para HTTPS) directamente, querrás configurar un servidor proxy, como Nginx, para reenviar las solicitudes al puerto en el que tu aplicación esté escuchando.

Para instalar Nginx:

bashCopy code

```
sudo apt install nginx
```

Luego, configura Nginx para reenviar las solicitudes al puerto de tu aplicación y reinicia Nginx.

## 6. Iniciar tu aplicación:

Utiliza **npm start** o **node** para iniciar tu aplicación como lo hiciste localmente.

## 7. Acceder a la aplicación:

En tu navegador, dirígete a la dirección IP pública de tu instancia EC2. Si configuraste todo correctamente, deberías ver tu aplicación en ejecución.

## Notas adicionales:

- Para entornos de producción, considera usar un administrador de procesos como **pm2** para mantener tu aplicación funcionando de manera constante.
- Si planeas tener la aplicación en producción, asegúrate de configurar el firewall correctamente, usar HTTPS (considera Certbot para obtener certificados SSL gratuitos) y seguir otras mejores prácticas de seguridad.
- AWS ofrece otros servicios como Elastic Beanstalk y ECS que pueden simplificar el proceso de despliegue de aplicaciones, pero requieren una curva de aprendizaje adicional.