Automatización de Infraestructura Digital I

Instrumento de Evaluación Unidad III

Ingeniería en Redes Inteligentes y Ciberseguridad

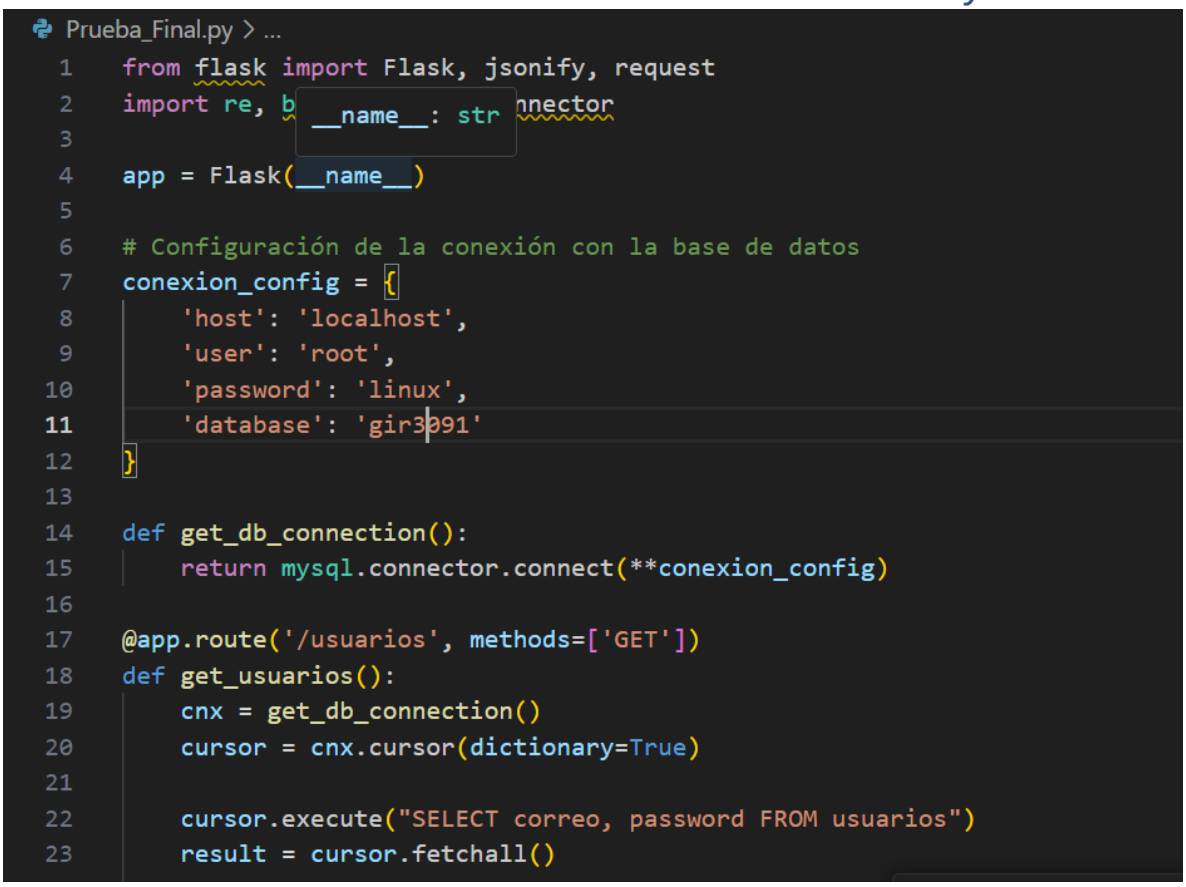
Unidad III: **Interfaces de Programación de Aplicaciones en la automatización de redes**

Integrantes: **Daniel Duarte Velázquez**

Docente: **Gabriel Barrón Rodríguez**

Fecha de Entrega: **22 de agosto de 2023**

Actividad 2: Creación de microservicios en Python



```python
from flask import Flask, jsonify, request
import re, b
__name__: str nnector

app = Flask(__name__)

# Configuración de la conexión con la base de datos
conexion_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'linux',
    'database': 'gir3091'
}

def get_db_connection():
    return mysql.connector.connect(**conexion_config)

@app.route('/usuarios', methods=['GET'])
def get_usuarios():
    cnx = get_db_connection()
    cursor = cnx.cursor(dictionary=True)

    cursor.execute("SELECT correo, password FROM usuarios")
    result = cursor.fetchall()
```

```python
    cursor.close()
    cnx.close()

    return jsonify(result), 200

def es_password_valida(password):
    if len(password) < 8:
        return False
    if not re.match(r"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$", password
        return False
    return True

@app.route('/usuarios', methods=['POST'])
def add_usuarios():
    data = request.get_json()

    if 'correo' not in data:
        return {'error':'Falta el campo correo'}, 400

    if 'password' not in data or not es_password_valida(data['password']):
        return {'error':'La contraseña es inválida'}, 400

    cnx = get_db_connection()
    cursor = cnx.cursor(dictionary=True)
```

```
    cursor.execute("SELECT correo FROM usuarios WHERE correo = %s", (data['correo'],))
    existing_user = cursor.fetchone()

    if existing_user:
        cursor.close()
        cnx.close()
        return {'error':'Este usuario ya está registrado'}, 409

    hashed_pw = bcrypt.hashpw(data['password'].encode('utf-8'), bcrypt.gensalt())
    data['password'] = hashed_pw.decode('utf-8')

    cursor.execute("INSERT INTO usuarios (correo, password) VALUES (%s, %s)", (data['correo'], data['
    cnx.commit()

    cursor.close()
    cnx.close()

    return {'success':'Registro agregado con éxito'}, 201

@app.route('/alumnos', methods=['GET'])
def get_alumnos():
    cnx = get_db_connection()
    cursor = cnx.cursor(dictionary=True)
```

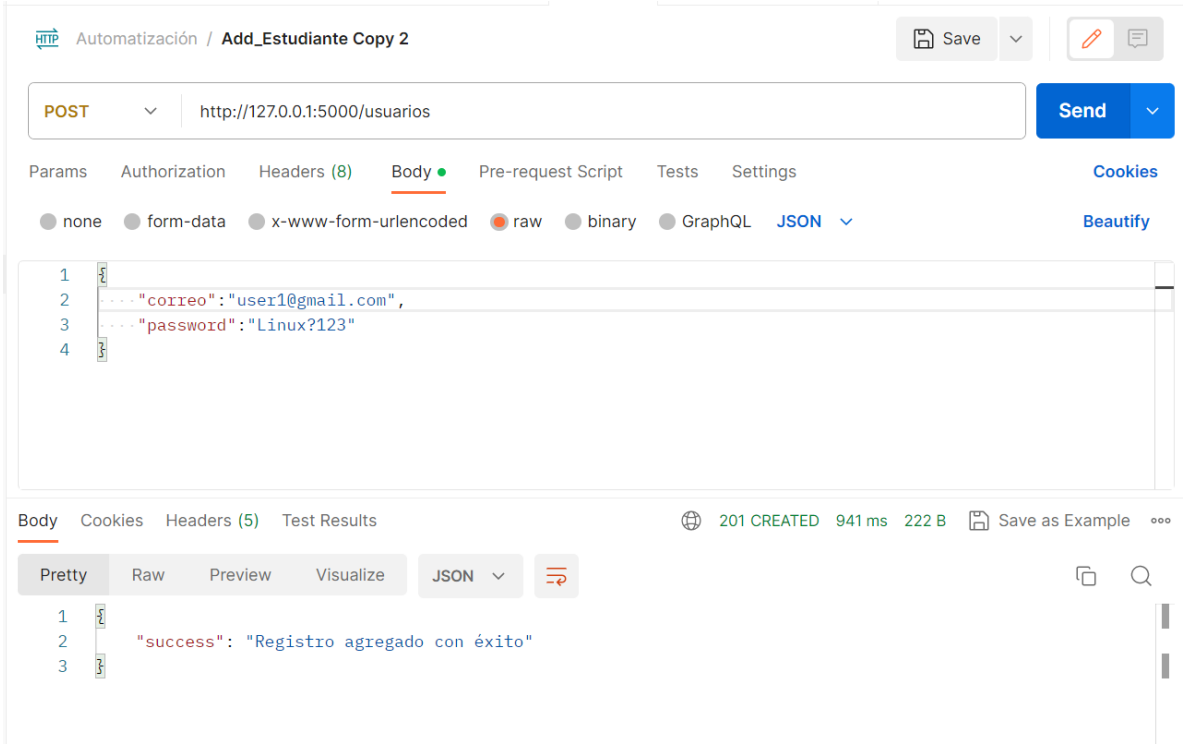
```
    cursor.execute("SELECT nombre, email, carrera, cuatrimestre, edad, numero_control, promedio FROM a
    result = cursor.fetchall()

    cursor.close()
    cnx.close()

    return jsonify(result), 200

@app.route('/alumnos', methods=['POST'])
def add_alumno():
    data = request.get_json()

    required_fields = ['usuario_id', 'nombre', 'email', 'carrera', 'cuatrimestre', 'edad', 'numero_cor

    if not all(key in data for key in required_fields):
        return {'error':'Faltan campos requeridos'}, 400

    cnx = get_db_connection()
    cursor = cnx.cursor(dictionary=True)

    cursor.execute("INSERT INTO alumnos (usuario_id, nombre, email, carrera, cuatrimestre, edad, numer
                   (data['usuario_id'], data['nombre'], data['email'], data['carrera'], data['cuatrime
    cnx.commit()
```

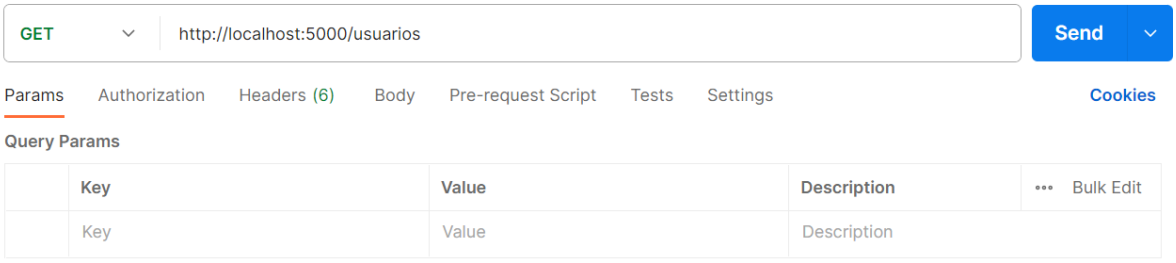
```python
    cursor.close()
    cnx.close()

    return {'success':'Alumno agregado con éxito'}, 201

@app.route('/materias', methods=['GET'])
def get_materias():
    cnx = get_db_connection()
    cursor = cnx.cursor(dictionary=True)

    cursor.execute("SELECT nombre_materia, carrera, cantidad_alumnos, area, periodo, maestro, edificio
    result = cursor.fetchall()

    cursor.close()
    cnx.close()

    return jsonify(result), 200

@app.route('/materias', methods=['POST'])
def add_materia():
    data = request.get_json()

    required_fields = ['usuario_id', 'nombre_materia', 'carrera', 'cantidad_alumnos', 'area', 'periodo
```

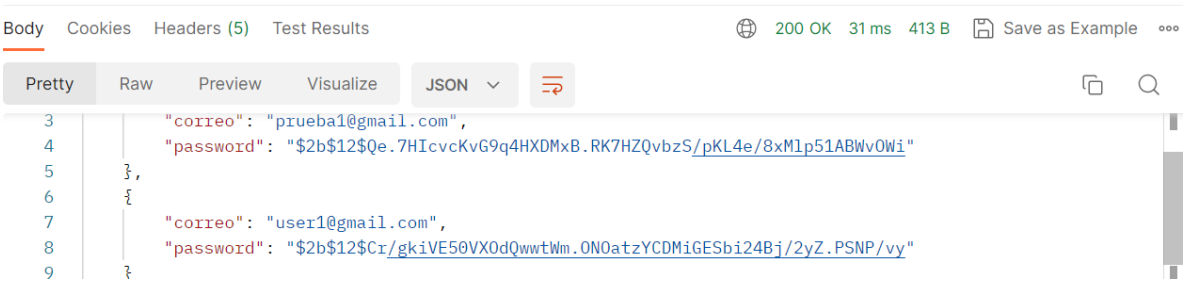
Prueba_Final.py > ...

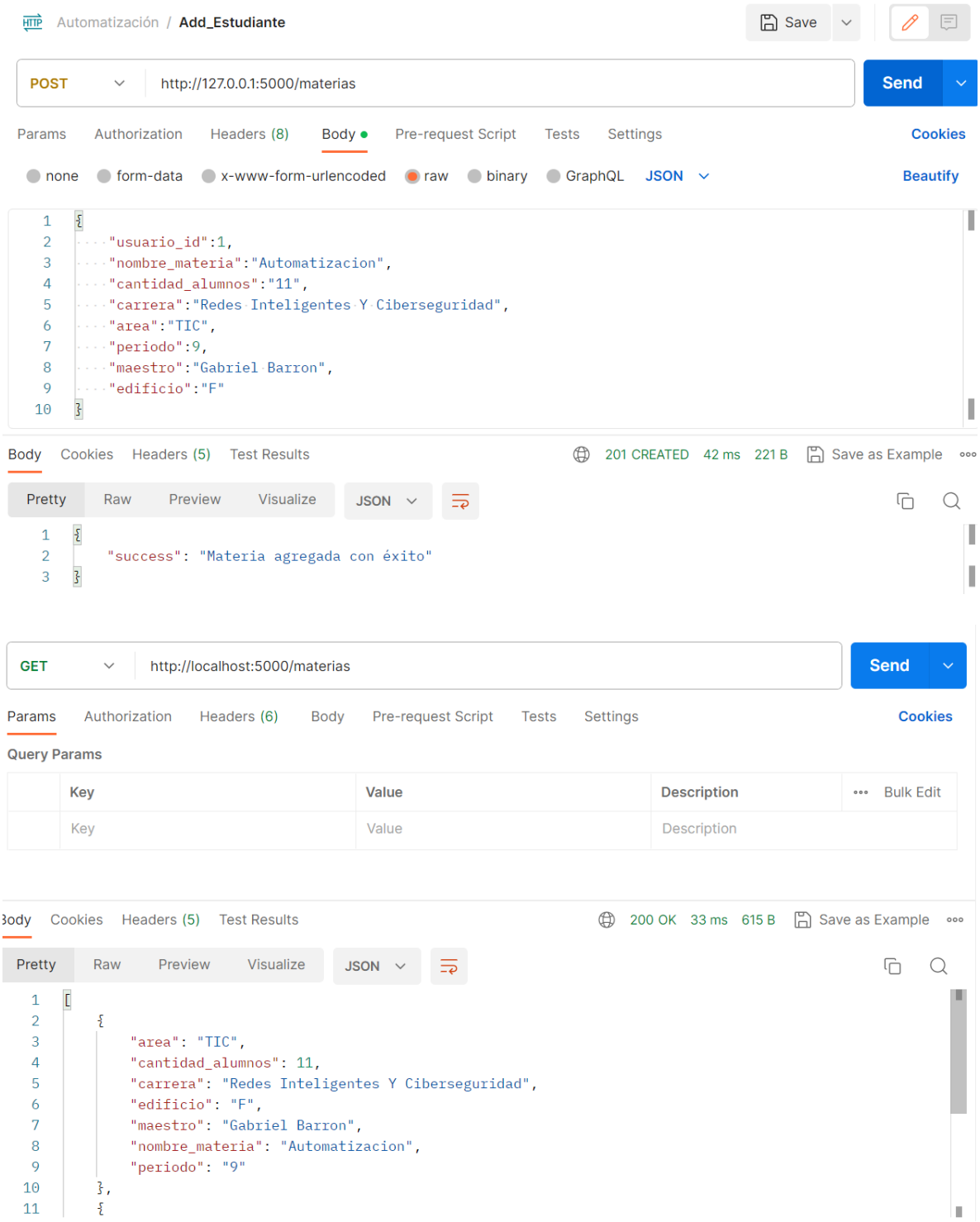
```python
    required_fields = ['usuario_id', 'nombre_materia', 'carrera', 'cantidad_alumnos', 'area', 'periodo

    if not all(key in data for key in required_fields):
        return {'error':'Faltan campos requeridos'}, 400

    cnx = get_db_connection()
    cursor = cnx.cursor(dictionary=True)

    cursor.execute("INSERT INTO materias (usuario_id, nombre_materia, carrera, cantidad_alumnos, area
                   (data['usuario_id'], data['nombre_materia'], data['carrera'], data['cantidad_alumn
    cnx.commit()

    cursor.close()
    cnx.close()

    return {'success':'Materia agregada con éxito'}, 201

if __name__ == "__main__":
    app.run(debug=True)
```

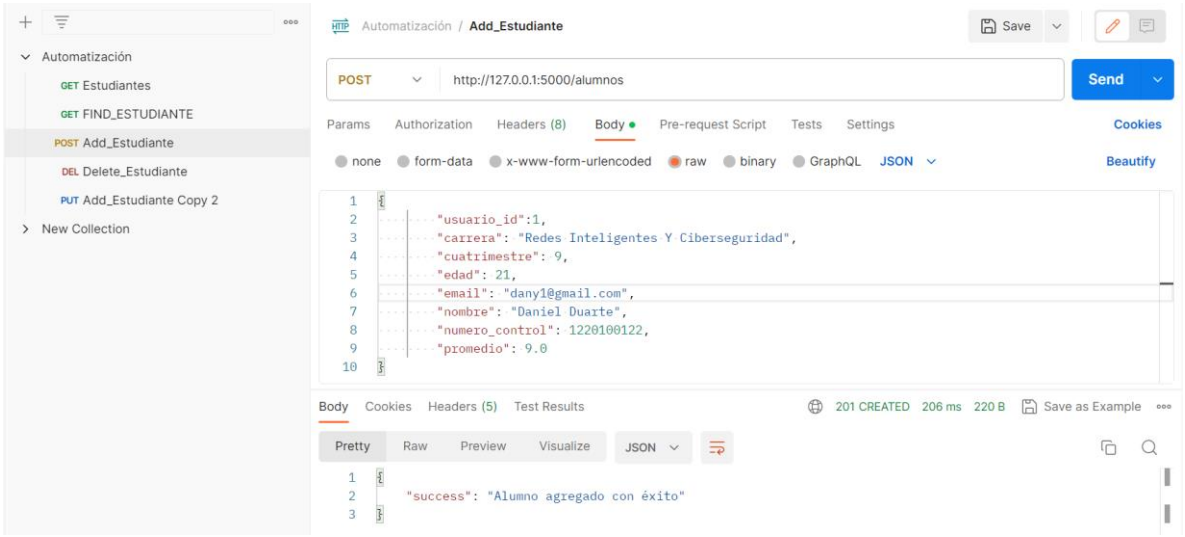
```
PS C:\Users\danyd\OneDrive\Documentos\Utng\Ingeniería\9no\Automatización de la I
ad III\Ejercicios de don Barron>                                ^C
PS C:\Users\danyd\OneDrive\Documentos\Utng\Ingeniería\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron> flask --app Prueba_Final.py run --debug
 * Serving Flask app 'Prueba_Final.py'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI ser
ver instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 282-868-922
```

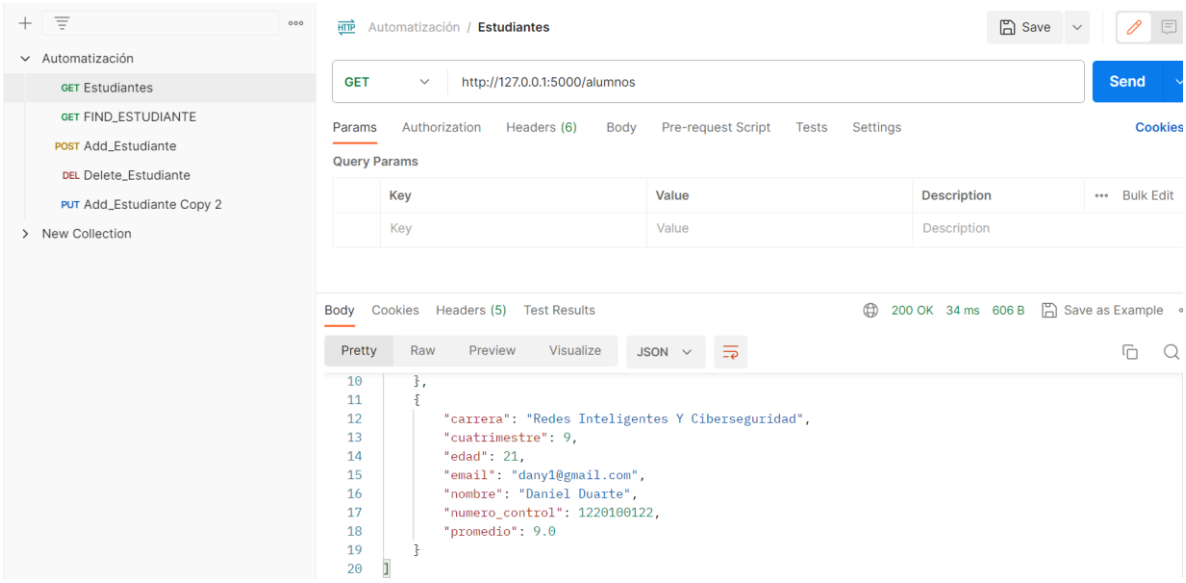

Automatización / Add_Estudiante Copy 2
Save
POST
http://127.0.0.1:5000/usuarios
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"correo":"user1@gmail.com",
"password":"Linux?123"
}
Body
Cookies
Headers (5)
Test Results
201 CREATED
941 ms
222 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"success": "Registro agregado con éxito"
}

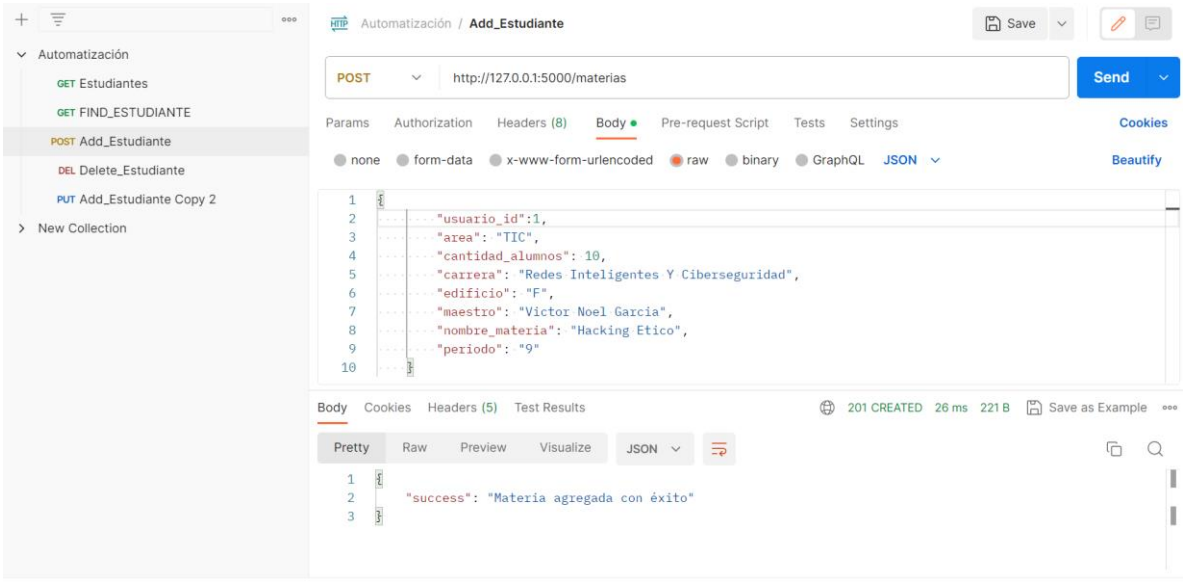

GET
http://localhost:5000/usuarios
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Key
Value
Description

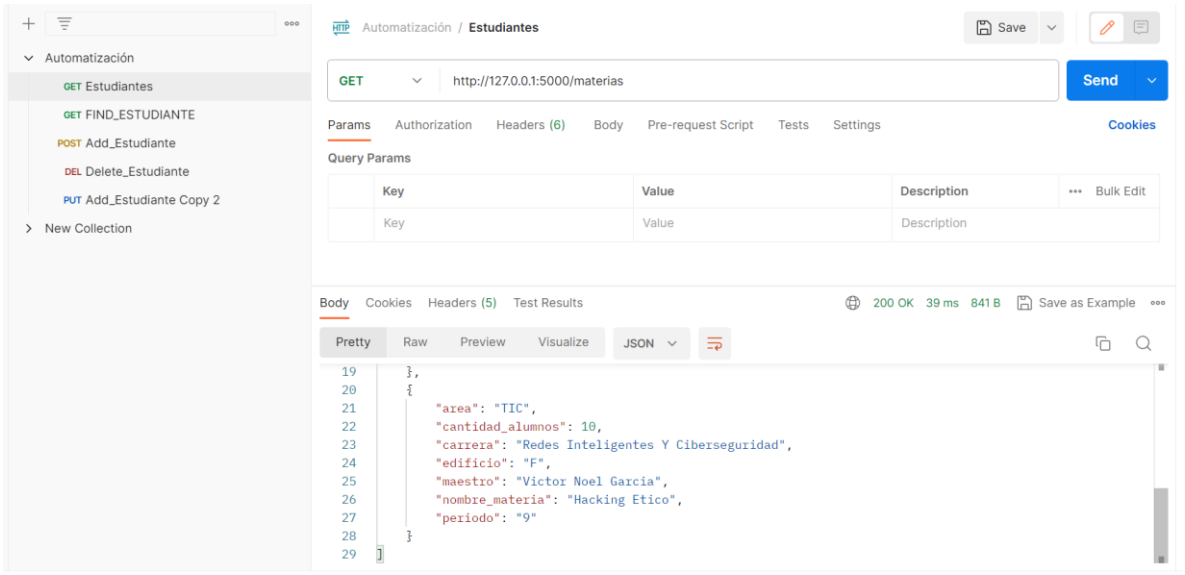

Body
Cookies
Headers (5)
Test Results
200 OK
31 ms
413 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
"correo": "prueba1@gmail.com",
"password": "$2b$12$Qe.7HIcvcKvG9q4HXDMxB.RK7HZQvbzS/pKL4e/8xMlp51ABWvOWi"
},
{
"correo": "user1@gmail.com",
"password": "$2b$12$Cr/gkiVE50VXOdQwwtWm.ONOatzYCDMiGESbi24Bj/2yZ.PSNP/vy"
}

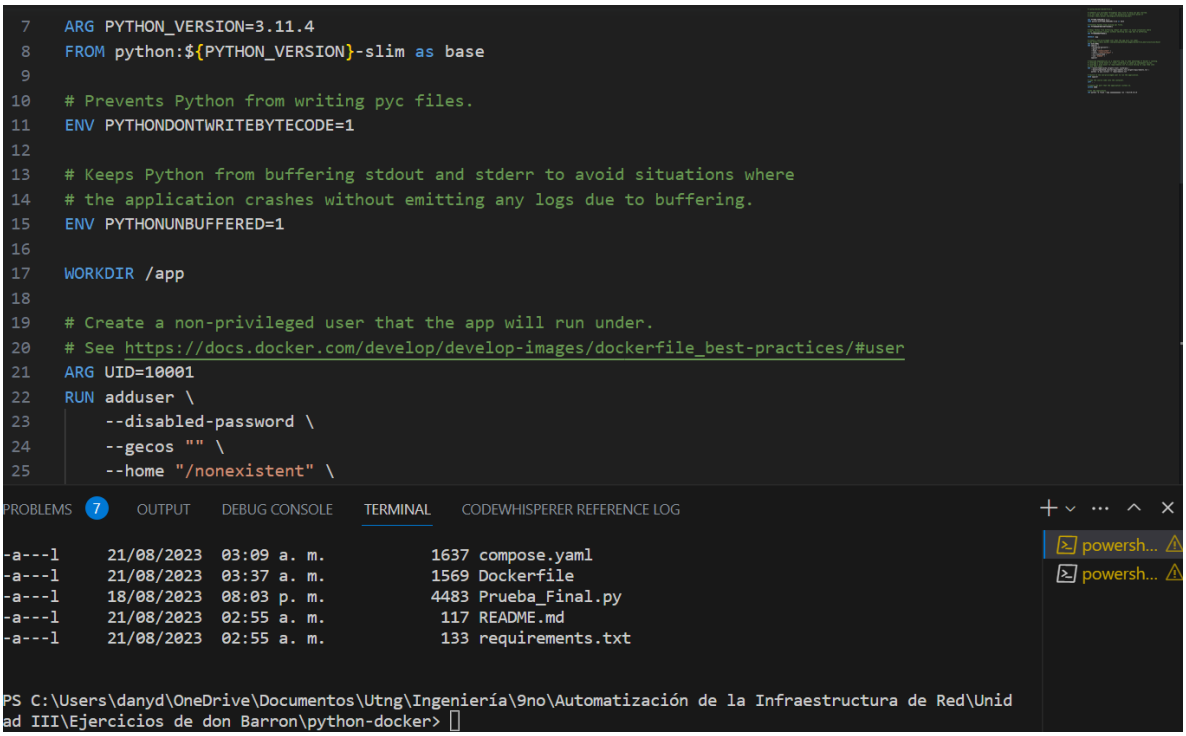

Automatización / Add_Estudiante
Save
POST
http://127.0.0.1:5000/materias
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"usuario_id":1,
"nombre_materia":"Automatizacion",
"cantidad_alumnos":"11",
"carrera":"Redes Inteligentes Y Ciberseguridad",
"area":"TIC",
"periodo":9,
"maestro":"Gabriel Barron",
"edificio":"F"
}
Body
Cookies
Headers (5)
Test Results
201 CREATED
42 ms
221 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"success": "Materia agregada con éxito"
}
GET
http://localhost:5000/materias
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
33 ms
615 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
[
{
"area": "TIC",
"cantidad_alumnos": 11,
"carrera": "Redes Inteligentes Y Ciberseguridad",
"edificio": "F",
"maestro": "Gabriel Barron",
"nombre_materia": "Automatizacion",
"periodo": "9"
},
{


Automatización
GET Estudiantes
GET FIND_ESTUDIANTE
POST Add_Estudiante
DEL Delete_Estudiante
PUT Add_Estudiante Copy 2
New Collection
Automatización / Add_Estudiante
Save
POST
http://127.0.0.1:5000/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
{
"usuario_id":1,
"carrera": "Redes Inteligentes Y Ciberseguridad",
"cuatrimestre": 9,
"edad": 21,
"email": "dany1@gmail.com",
"nombre": "Daniel Duarte",
"numero_control": 1220100122,
"promedio": 9.0
}
Body Cookies Headers (5) Test Results
201 CREATED 206 ms 220 B
Save as Example
Pretty Raw Preview Visualize JSON
{
"success": "Alumno agregado con éxito"
}


Automatización
GET Estudiantes
GET FIND_ESTUDIANTE
POST Add_Estudiante
DEL Delete_Estudiante
PUT Add_Estudiante Copy 2
New Collection
Automatización / Estudiantes
Save
GET
http://127.0.0.1:5000/alumnos
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 34 ms 606 B
Save as Example
Pretty Raw Preview Visualize JSON
},
{
"carrera": "Redes Inteligentes Y Ciberseguridad",
"cuatrimestre": 9,
"edad": 21,
"email": "dany1@gmail.com",
"nombre": "Daniel Duarte",
"numero_control": 1220100122,
"promedio": 9.0
}
]

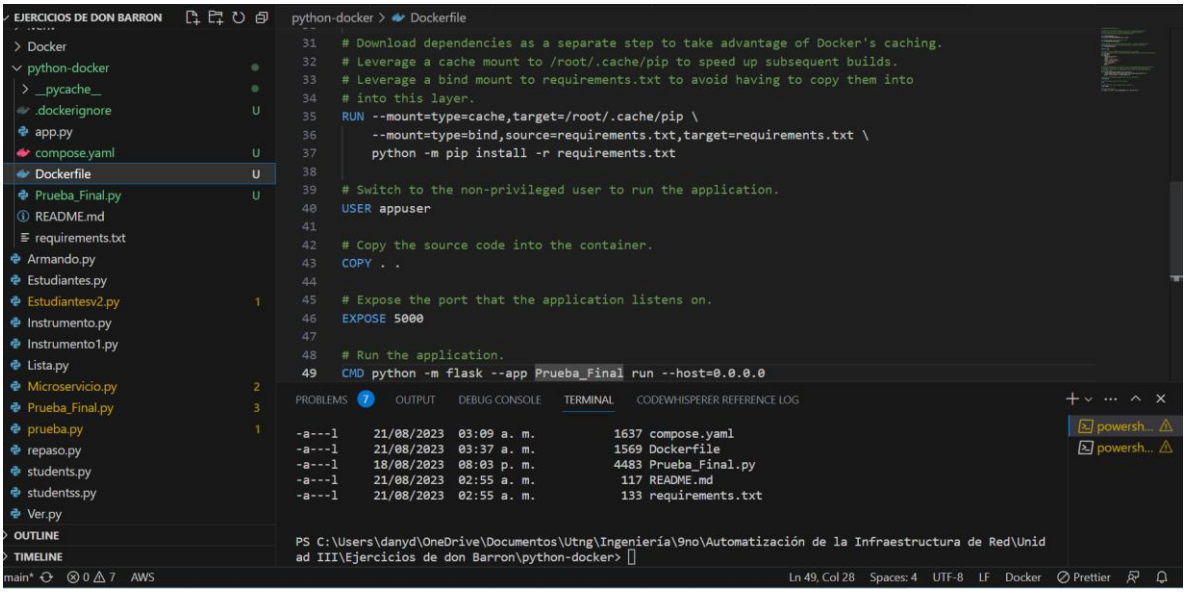

Automatización
GET Estudiantes
GET FIND_ESTUDIANTE
POST Add_Estudiante
DEL Delete_Estudiante
PUT Add_Estudiante Copy 2
New Collection
Automatización / Add_Estudiante
Save
POST
http://127.0.0.1:5000/materias
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"usuario_id":1,
"area": "TIC",
"cantidad_alumnos": 10,
"carrera": "Redes Inteligentes Y Ciberseguridad",
"edificio": "F",
"maestro": "Victor Noel Garcia",
"nombre_materia": "Hacking Etico",
"periodo": "9"
}
Body
Cookies
Headers (5)
Test Results
201 CREATED
26 ms
221 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"success": "Materia agregada con éxito"
}


Automatización
GET Estudiantes
GET FIND_ESTUDIANTE
POST Add_Estudiante
DEL Delete_Estudiante
PUT Add_Estudiante Copy 2
New Collection
Automatización / Estudiantes
Save
GET
http://127.0.0.1:5000/materias
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
39 ms
841 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
},
{
"area": "TIC",
"cantidad_alumnos": 10,
"carrera": "Redes Inteligentes Y Ciberseguridad",
"edificio": "F",
"maestro": "Victor Noel Garcia",
"nombre_materia": "Hacking Etico",
"periodo": "9"
}
]



```
ARG PYTHON_VERSION=3.11.4
FROM python:${PYTHON_VERSION}-slim as base

# Prevents Python from writing pyc files.
ENV PYTHONDONTWRITEBYTECODE=1

# Keeps Python from buffering stdout and stderr to avoid situations where
# the application crashes without emitting any logs due to buffering.
ENV PYTHONUNBUFFERED=1

WORKDIR /app

# Create a non-privileged user that the app will run under.
# See https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#user
ARG UID=10001
RUN adduser \
    --disabled-password \
    --gecos "" \
    --home "/nonexistent" \
```

```
PROBLEMS 7  OUTPUT  DEBUG CONSOLE  TERMINAL  CODEWHISPERER REFERENCE LOG

-a---1     21/08/2023  03:09 a. m.          1637 compose.yaml
-a---1     21/08/2023  03:37 a. m.          1569 Dockerfile
-a---1     18/08/2023  08:03 p. m.          4483 Prueba_Final.py
-a---1     21/08/2023  02:55 a. m.           117 README.md
-a---1     21/08/2023  02:55 a. m.           133 requirements.txt

PS C:\Users\danyd\OneDrive\Documentos\Utng\Ingeniería\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\python-docker>
```



```
EJERCICIOS DE DON BARRON
  Docker
  python-docker
    __pycache__
    .dockerignore U
    app.py
    compose.yaml U
    Dockerfile U
    Prueba_Final.py U
    README.md
    requirements.txt
  Armando.py
  Estudiantes.py
  Estudiantesv2.py 1
  Instrumento.py
  Instrumento1.py
  Lista.py
  Microservicio.py 2
  Prueba_Final.py 3
  prueba.py 1
  repaso.py
  students.py
  studentss.py
  Ver.py
OUTLINE
TIMELINE
```

```
python-docker > Dockerfile

# Download dependencies as a separate step to take advantage of Docker's caching.
# Leverage a cache mount to /root/.cache/pip to speed up subsequent builds.
# Leverage a bind mount to requirements.txt to avoid having to copy them into
# into this layer.
RUN --mount=type=cache,target=/root/.cache/pip \
    --mount=type=bind,source=requirements.txt,target=requirements.txt \
    python -m pip install -r requirements.txt

# Switch to the non-privileged user to run the application.
USER appuser

# Copy the source code into the container.
COPY . .

# Expose the port that the application listens on.
EXPOSE 5000

# Run the application.
CMD python -m flask --app Prueba_Final run --host=0.0.0.0
```

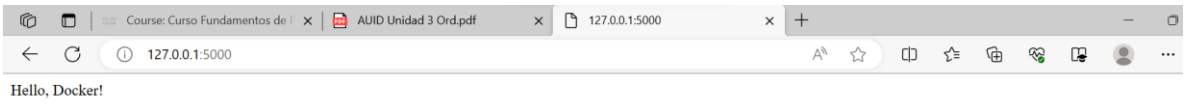
```
PROBLEMS 7  OUTPUT  DEBUG CONSOLE  TERMINAL  CODEWHISPERER REFERENCE LOG

-a---1     21/08/2023  03:09 a. m.          1637 compose.yaml
-a---1     21/08/2023  03:37 a. m.          1569 Dockerfile
-a---1     18/08/2023  08:03 p. m.          4483 Prueba_Final.py
-a---1     21/08/2023  02:55 a. m.           117 README.md
-a---1     21/08/2023  02:55 a. m.           133 requirements.txt

PS C:\Users\danyd\OneDrive\Documentos\Utng\Ingeniería\9no\Automatización de la Infraestructura de Red\Unid
ad III\Ejercicios de don Barron\python-docker>
```

```
main*  0 7  AWS                Ln 49, Col 28  Spaces: 4  UTF-8  LF  Docker  Prettier
```

Link de acceso a GitHub: https://github.com/duartdany/Instrumento_Evaluacion_U-3

Link de acceso a jira:

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


Course: Curso Fundamentos de
AUID Unidad 3 Ord.pdf
127.0.0.1:5000
127.0.0.1:5000
Hello, Docker!