

Universidad de Ingenieria y Tecnologia
Departamento de Ciencias de la Computación

Ingeniería de Software I

Lab VII: Pipelines & Project Structure
2024 - II

TEAM	Los Del Fondo
INTEGRANTES	<ul style="list-style-type: none">- José Rodriguez- Yesenia Valero- Jose Bojorquez- Piero Guerrero- Aarón Timaná- Gonzalo Suárez Torres

Background:

Pipelines
Project Structure

CASO DE ESTUDIO (level: advanced):

Devies Inc. tiene un problema constante con sus nuevos empleados que no siguen los estándares que espera management. Se tienen constantes perdidas de tiempo al hacer onboarding de nuevos empleados. Y tambien cuando se hacen integraciones de codigo es muy dificil combinar los diferentes estilos de codigo.

Problema:

Se le contrata como devops para desarrollar herramientas para resolver esta situacion. Por lo que usted decide crear un CLI en donde por comandos se podra generar el esqueleto de la aplicación standard de la empresa.

1. Elegir la estructura de proyecto estandar. (3-tier, eventdriven, etc).
2. Crear una CLI que genere el esqueleto de proyecto en base a lo seleccionado en el punto 1 para un CRUD y otro para crear una API con conexion a base de datos.
3. La API creada debe contener un endpoint /hello que devolviera un json sample.

Ejemplo:

>> generate –crud newProject

..generating CRUD

..output of project structure in : C:/newProject

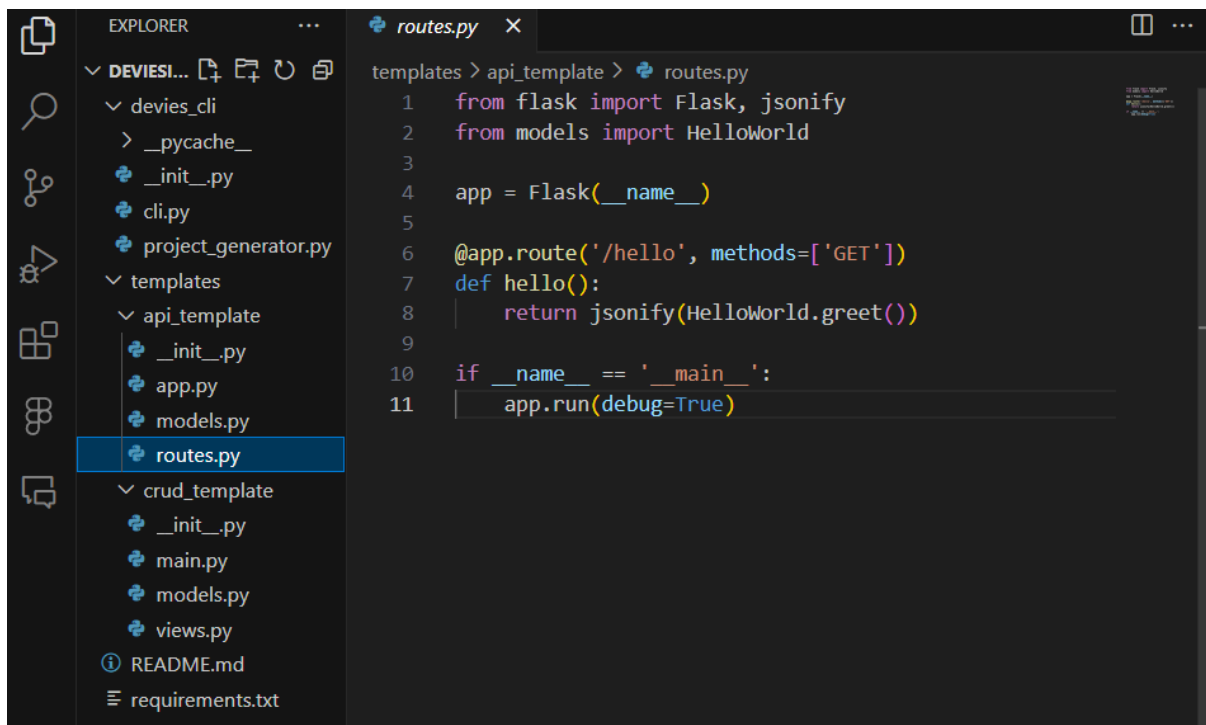
>>generate -api newAPI

..generating CRUD

..output of project structure in : C:/newAPI

Coloque la url del Repo de su CLI y screenshots del output:

1. Se seleccionará la estructura 3-tier ya que es la más común en aplicaciones CRUD y API. Esta se divide en controller, service y repository.
- 2.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree for a project named 'DEVIESI...'. The tree is expanded to show the 'api_template' directory, which contains files like '_init_.py', 'app.py', 'models.py', and 'routes.py'. The 'routes.py' file is selected and highlighted. The main editor area shows the content of 'routes.py', which is a Flask application template. The code includes imports for Flask and jsonify, a Flask app instance, a route for '/hello' with a GET method, a 'hello' function that returns a JSON response from a 'HelloWorld' object, and a main block that runs the app in debug mode.

```
templates > api_template > routes.py
1  from flask import Flask, jsonify
2  from models import HelloWorld
3
4  app = Flask(__name__)
5
6  @app.route('/hello', methods=['GET'])
7  def hello():
8      return jsonify(HelloWorld.greet())
9
10 if __name__ == '__main__':
11     app.run(debug=True)
```

3.

```
import os
import shutil

class ProjectGenerator:
    def generate(self, project_type, project_name):
        if project_type == 'crud':
            self._create_crud_project(project_name)
```

```
        elif project_type == 'api':
            self._create_api_project(project_name)

def _create_crud_project(self, project_name):
    template_dir = 'templates/crud_template'
    self._copy_template(template_dir, project_name)
    print(f'Proyecto CRUD "{project_name}" creado con éxito.')

def _create_api_project(self, project_name):
    template_dir = 'templates/api_template'
    self._copy_template(template_dir, project_name)
    print(f'Proyecto API "{project_name}" creado con éxito.')

def _copy_template(self, template_dir, project_name):
    target_dir = os.path.join(os.getcwd(), project_name)
    shutil.copytree(template_dir, target_dir)
```