Universidad de Ingenieria y Tecnologia
Departamento de Ciencias de la Computación

Ingeniería de Software I

Lab VII: Pipelines & Project Structure
2024 - II

TEAM	**Segmentation Fault**
INTEGRANTES	● **Manyory Estefany Cueva Mendoza** ● **Christopher Eloy Najarro Mancco** ● **Mauricio Salazar Hillenbrand** ● **Angel Ulises Tito Berrocal** ● **Leonardo Matias Candio Ormeño**

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 devolvera un json sample.

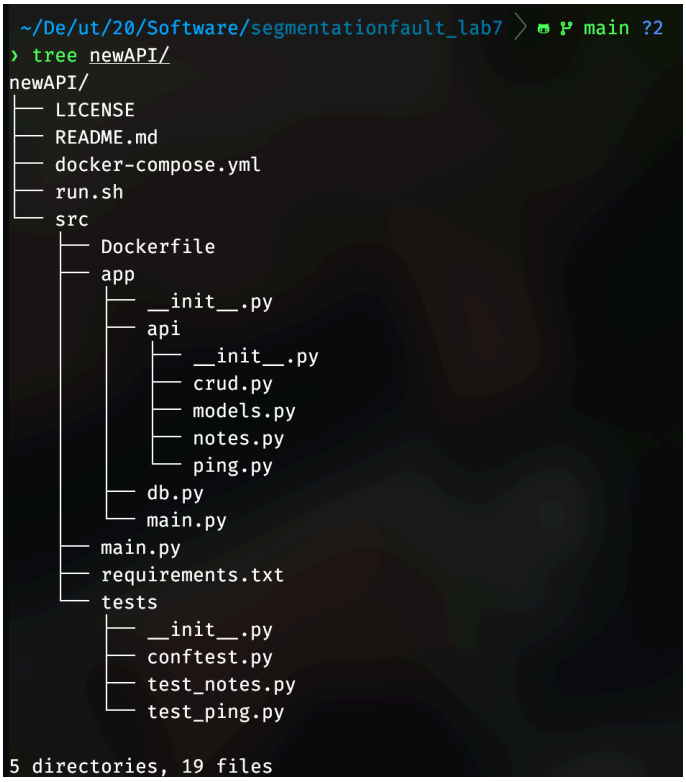
Ejemplo:

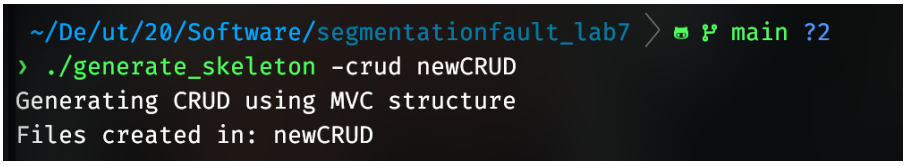
>> generate –crud newProject
..generating CRUD
..output of project structure in : C:/newProject

>>generate -api newAPI
..generating CRUD
..output of project structure in : C:/newAPI

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

https://github.com/utecsw20242/segmentationfault_lab7.git






~/De/ut/20/Software/segmentationfault_lab7 main ?3
› tree newCRUD -L 3
newCRUD
├── Dockerfile
├── LICENSE
├── README.md
├── app
│   ├── constants.py
│   ├── controllers
│   │   ├── auth_controller.py
│   │   ├── page_controller.py
│   │   └── user_controller.py
│   ├── db
│   │   └── context.py
│   ├── db_init.bash
│   ├── db_init.py
│   ├── dev.bash
│   ├── main.py
│   ├── middlewares
│   │   ├── cors_middleware.py
│   │   └── static_middleware.py
│   ├── models
│   │   ├── db.py
│   │   └── dto.py
│   ├── prod.sh
│   ├── repos
│   │   └── user_repo.py
│   ├── services
│   │   ├── jwt_service.py
│   │   └── user_service.py
│   ├── static
│   ├── templates
│   │   └── main.jinja
│   └── utils
│   ├── background_schedule_task.py
│   ├── bcrypt_hashing.py
│   ├── dependencies.py
│   ├── formating.py
│   ├── lifespan.py
│   └── sha256_hashing.py
├── docker-compose.yml
├── install_env.sh
└── requirements.txt
11 directories, 30 files