ONBOARDING APP UNIR PYTHON

Arquitectura Software

- Nosotros utilizamos la **arquitectura por capas** ya que el código es más modular y fácil de escalar y extender.

  Presentación - Esta capa está implementada con la librería **Flask** donde controlamos la entrada y salida de datos, esta es la encargada de recibir las solicitudes por parte del usuario desde el Front-End o desde Postman, se encuentra en el fichero **routes.py**

  Lógica de Negocio - Contiene la lógica del sistema, como son las reglas, validaciones etc…., se encuentra en el fichero **services.py**

  Capa de Acceso a Datos - Es la encargada de comunicar con la base de datos, definimos las tablas, columnas y relaciones, lo hacemos con la librería **SQLAlchemy** y se encuentra en el fichero **models.py**

  Base de datos - Actualmente utilizamos **PostgresSQL**

  Otros - También utilizamos **Docker** para poder encapsular la aplicación y la BBDD, así formando el entorno de desarrollo, por otra parte también implementamos **pytest**

Como se ejecutan los tests

- Los test están alojados en la siguiente ruta:
  app/test_routes.py
- Para lanzar los test, tendremos que irnos a esa ruta y ejecutar el siguiente comando:
  pytest test_routes.py
- Para configurar un nuevo test, tendremos:
  - Agregar una nueva llamada en el fichero **routes.py**
  - Posteriormente tendremos que agregar en el fichero **test_routes.py,** la funcionalidad con el mismo route, siguiendo el modelo de los demás tests.

OBSERVACIÓN: Cuando se lanza los test, generamos una nueva bbdd, entonces la que tenemos cuando lanzamos el entorno perdemos la conexión, tendremos que lanzar el fichero **manage.py** para poder lanzar de nuevo la BBDD.

Como se ejecuta localmente el entorno para pruebas

- Para poder desplegar el entorno tendremos que:
  1º Lanzar **docker-compose build –no-cache**
  2º Lanzar **docker-compose up**

  En el primer comando estaremos construyendo la imagen de nuestra máquina local.
  En el segundo comando estamos lanzando los contenedores que hemos definido en nuestro yml

NORMAS DE COLABORACIÓN

- Ramas: Siempre trabajaremos de la siguiente forma:
    - **Main**: En esta rama solo estará aquél código que sea apto para producción, solo se subirá cambios desde develop, con la MR aceptada y asignada al TL.
    - **Develop**: En esta rama estaremos trabajando con funcionalidades que provengan desde una rama **feature,** y donde se probará esa nueva funcionalidad con el código que ya esté en producción para hacer los test correspondientes
    - **Feature:** Cada nueva funcionalidad vendrá de esta rama, esta ramá deberá siempre ser creada a raíz de develop y posteriormente mergear a develop con las pruebas en local ya realizadas y con el funcionamiento correcto

¡ATENCIÓN! NO SE DESARROLLA DIRECTAMENTE EN MAIN

- Todo merge tendrá que ser seguido de una **MR**, que será asignada al TL, como se ha comentado anteriormente.

- Utilizar tags en los commits, **feat/fix: <mensaje descriptivo>**.