# Contexte

L’organisme Digital College a besoin d’un outil Web APIfié (à venir dans un second temps la partie API) qui leur permet de recenser et traiter les ressentis, les retours des apprenants afin de pouvoir ajuster et améliorer certains aspects des parcours pédagogiques, au fil de l’eau.

La solution choisie : Création d’une application Web pour DC avec un Framework Python (Flask ou Django vu en cours), un ORM (pour faire le mapping entre le modèle de données en Python et les tables de la BDD) avec du traitement et de l’analyse de données (dans un second temps).

Ce projet réel permet d'appliquer des compétences pratiques en data engineering, en développement web et en automatisation de tests.

Ce projet, en équipe doit vous permettre d’élaborer une réflexion sur un projet concret de data engineering.

# Fonctionnalités

**Soumission de retour** : Permettre aux apprenants de soumettre des retours sur le bootcamp et l’organisme de formation en général via un formulaire web. Les retours peuvent inclure des évaluations numériques (par exemple, des notes de 1 à 5) et des commentaires textuels. **Le code frontend de l’application vous est fourni.**

**Consultation de retours** : Une interface d'administration pour consulter les retours soumis, filtrer par date, sujet, ou d'autres critères pertinents.

**Analyse de sentiment** : Intégrer une analyse de sentiment sur les commentaires pour obtenir une vue d'ensemble de la perception des élèves. (à venir dans un second temps)

**Rapports et statistiques** : Générer des rapports automatiques sur les données collectées, tels que la moyenne des évaluations, les tendances des sentiments, etc. (à venir dans un second temps)

# Flux de données

**Soumission et stockage des données** : Les apprenants soumettent leurs retours via un formulaire web. Les données sont collectées en format de votre choix (à justifier) et envoyées vers une BDD.

**Traitement des données** : Utiliser SQL pour exécuter des requêtes sur les données stockées dans la BDD de votre choix et préparer les données pour l'analyse.

**Analyse et visualisation** : Les données traitées peuvent être visualisées via une application Python (Flask/Django) qui sert des dashboards en HTML/CSS.

# Architecture

**Frontend** : HTML/CSS pour le design, avec JavaScript pour dynamiser les interfaces de soumission de retour et de visualisation des données.

**Backend** : Une application Python (Flask ou Django) qui gère la logique de l'application, les interactions avec la BDD en utilisant un ORM (ORM Django ou SQLAlchemy en fonction du framework web utilisé), et sert le frontend. Une API Rest via Flask ou Django qui permet de consommer les différents services de l’application. (À venir dans un second temps).

**Stockage des données** : SGBD SQL.