



# Projet TP7 – Développement d’une application web

---

- **Formation** : 2025-D04 Data Engineer
- **Encadrants** :
  - Référent pédagogique & formateur : Robin HOTTON (GitHub : [robinhotton](#))
  - Client : Valentin Momin

## Sommaire

- 🎯 Objectifs du projet
- 📁 Contexte du projet
- 📁 Livrables attendus
- ⚙️ Exigences techniques
- 📄 Mini Cahier des Charges – DigiCheese
- 🛠️ Stratégie de tests
- 🧠 Critères de réussite / notation
- 📢 Présentation finale – Lundi 30 Juillet 2025

## Objectifs du projet

Ce projet s'inscrit dans le cadre du Bloc 3 de la formation Data Engineer et vise à :

- Faire suite à la conception de l'application réalisée au préalable.
- Mettre en pratique les concepts appris ainsi que de nouveaux concepts pour développer une API backend en Python.
- Structurer le code et la documentation technique dans un environnement professionnel.
- Livrer un projet fonctionnel, testé, documenté et versionné via Git.
- Répondre à un mini cahier des charges : **DigiCheese**.

## Contexte du projet

Le projet consiste à réaliser une API, avec la documentation Swagger, une base de données MySQL/MariaDB, une gestion propre des tests et un dépôt Git collaboratif. Le travail est à effectuer en groupe de 3 à 4 personnes.

Veuillez créer/remplir un fichier groupes.xmlx à la racine de vos support de cour.

## Livrables attendus

- **Une arborescence Git propre :**
  - Une branche par développeur.
  - Les branches : **test**, **dev**, **master**.
- **Dossier **src/** :**
  - Les **sources Python** (**models.py**, **database.py**...).
  - Scripts SQL (**scripts\_mysql/**) pour initialiser la base.
- **Dossier **tests/** :**
  - Scénarios de tests.
  - Scripts de tests unitaires (via **pytest** ou **unittest**).
- **Dossier **docs/** (Documentation technique) :**
  - Architecture logicielle.
  - Description des éléments modifiés selon le cahier des charges.
  - Spécifications des serveurs virtuels Python.
- **Fichier **README.md** :**
  - Explication du projet, installation, usage, endpoints Swagger.
- **Fichier **requirements.txt** :**
  - Dépendances Python du projet.
- **Fichier **.env** :**
  - Variables d'environnement (non versionné dans Git).
- **Dossier **.venv/** :**
  - Environnement virtuel Python (optionnel mais recommandé localement).

Attention, certains de ces fichiers NE DOIVENT PAS être versionnés dans Git (comme **.env**).  
Créez un fichier template comme **.env.template** pour indiquer les variables attendues.

## Exigences techniques

- API Backend développée en **Python**.
- Base de données : **MySQL / MariaDB**.
- Documentation Swagger automatique via FastAPI.
- Tests API obligatoires avec **pytest** ou **unittest**.
- Typage fort (**type hints**) obligatoire.
- Respect des normes de codage Python (PEP8) vu en cours et pendant le projet.
- Ajout des formateurs & clients présentés en première page au dépôt GitHub.

## Mini Cahier des Charges – DigiCheese

Ce document présente les besoins fonctionnels et techniques d'une entreprise fictive : DigiCheese. L'objectif est d'implémenter une API répondant aux spécifications métiers fournies dans ce cahier. Les fichiers de départ (**models.py**, **database.py**, **main.py** et **run.py**) guident la structure attendue.

## Stratégie de tests

Une bonne stratégie de tests inclura :

- **Niveaux de tests** :
  - Unitaires.
- **Objectifs** :
  - Garantir la stabilité, la fiabilité et la sécurité du code.
- **Responsabilités** :
  - Chaque membre développe et documente les tests de ses fonctions.
- **Tâches principales** :
  - Écrire des tests reproductibles.
- **Critères d'entrée / sortie** :
  - Entrée : tests définis pour chaque endpoint ou module.
  - Sortie : couverture > 80%, tests automatisés réussis.

## Critères de réussite / notation

Critère	Description
<input checked="" type="checkbox"/> Application fonctionnelle	L'API fonctionne avec les fonctionnalités prévues
<input checked="" type="checkbox"/> Respect du cahier des charges	DigiCheese correctement interprété et respecté
<input checked="" type="checkbox"/> Code lisible et structuré	Commentaires, nommage, architecture claire
<input checked="" type="checkbox"/> Conventions respectées	Typage, normes Python, pas de duplication
<input checked="" type="checkbox"/> Tests efficaces	Tests unitaires et intégration pertinents
<input checked="" type="checkbox"/> Documentation complète	Technique, fonctionnelle et conforme au projet
<input checked="" type="checkbox"/> Gestion Git propre	Branches, commits, pull requests clairs

## Présentation finale – Lundi 30 Juillet 2025

Chaque groupe présentera son projet de façon professionnelle, comme dans un **appel d'offre client** :

- **Support** au choix : Diaporama, démonstration en direct, vidéo, etc...
- **Durée** :
  - ± 20 minutes de présentation
    - (5 minutes de tolérance)
  - ± 10 minutes de questions
- **Ordre de passage** :
  - Groupe 1 : 14h30
  - Groupe 2 : 15h30
  - Groupe 3 : 16h30