# [IMAC S2] Programmation Back-end Projet Final

## Sujet

Vous réaliserez un  service Web Restful avec comme technologies :

- Python en langage back (Avec le Framework Flask)
- MySQL en tant que SGBD
- Côté front:  peu importe . Tant que le client arrive à lancer des requêtes HTTP.
- Git pour le système de versionning (suivez les tutos !)

Le site doit être accessible en ligne si possible.
L'API côté back est la priorité du projet . Le but étant de pouvoir manipuler une base de données avec les 4 principales actions :  **Create, Read, Update, Delete** .

Cependant  une interface côté client est demandée . Le site doit pouvoir réaliser au moins 10 actions sur votre base de données (*le site doit avoir au moins 10 routes/endpoints*). Ne passez pas trop de temps sur la présentation, c'est plutot le back qui m'interesse…

La base de données ne doit pas être trop grosse ( *genre 5-10 tables maxi , c'est déjà bien*). Vous pouvez vous servir de votre propre méthode pour modéliser votre bdd, cependant si vous savez faire du Merise, vous pouvez le fournir. Sinon, faites bien attention, des erreurs de conception sont courantes..
Ayez en tête que les sessions et les systèmes de login/mdp ne sont pas demandés (pas le temps de voir en cours). Pensez alors à faire votre site qui ne soit pas restreint par l'authentification (cependant vous pouvez en intégrer dans votre site, vous faites ce que vous voulez).

Le thème du site est libre ! Mais si vous n'avez pas d'idées, j'ai quelques exemples pour vous :

- un système de playlist (musique, séries, bouquins/bd...)
- un dashboard (contrôle et suivi d'activités quelconque)
- un outil pour les associations (genre location matériel 803Z, gestion du BDK, du FDK...)
- un pokédex-like,
- un mini réseau-social,
- un site qui permet de préparer l'arrivée des 'années folles' post covid (Recensement de sorties à faire, d'activités à prévoir, possibilité de s'y inscrire !)

# Livrables

- Repo Git : le ou la chef.fe de projet devra envoyer un lien du repo Git à communiquer dès le début du projet. Date limite : **dimanche 9 avril** à 23:59
- Pitch du projet : À envoyer en même temps que le repo Git. Courte description de votre projet qui va RÉVOLUTIONNER le ouèbe 2.0. Date limite : **dimanche 9 avril** à 23:59.

- Base de données : ébauche du MCD si vous avez, ou toute description de votre base de données (Schéma relationnel en intension) à envoyer avant le **dimanche 16 avril** à 23:59. La version finale du schema doit figurer dans le rapport.
- Site :
  - premier hébergement avec Python/Flask qui tourne à envoyer avant le **dimanche 1 mai** à 23:59 (pas grave si ça ne ressemble à rien).
  - Livraison FINALE **dimanche 28 mai** à 23:59 !

Rapport : document de 10-20 pages en format PDF. Date limite : en même temps que la livraison finale du site (**dimanche 28 mai** à 23:59).

- Soutenance : debut juin (prévue ***debut juin***)
- A déposer sur le ELEARNING (seul.e le ou la chef.fe de projet dépose, en indiquant bien le nom du groupe et la liste des participant.e.s

TOUTE LIVRAISON QUI A ÉTÉ FAITE APRÈS LA DATE LIMITE DEMANDÉE DONNERA AU GROUPE UNE PÉNALITÉ, TENEZ MOI AU COURANT SI VOUS AVEZ DES PROBLÈMES

# Notation

## Rapport

- Version PDF d'environ 10 à 20 pages (sans les annexes)
- Qualité de l'analyse et de l'orthographe à prendre en compte
- exemple de contenu :
  - présentation de l'équipe,
  - présentation du projet (description du site, du contexte, motivations, objectifs, concept),
  - répartition du groupe et planning,
  - (éventuellement recherches graphiques),
  - architecture et gabarit du site (arborescence, schéma de la bdd),
  - tâches et fonctionnalités à implémenter (liste des endpoints)
  - gestion de projet (avancement, méthodes utilisées),
  - rendu, satisfaction du projet,
  - post-mortem
  - annexes : tout le processus de la création de la bdd, manuel utilisateur API (liste des endpoints avec: titre, description, méthode, URL, paramètres URL/data(corps de requête) , réponses/erreurs, exemples d'utilisation, …)

## Soutenance (20 minutes max)

Lors de la soutenance, vous vous attacherez à bien présenter ce que vous avez couvert coté back-end : vos endpoints, leurs caractéristiques, et mise en avant de ce dont vous êtes fiers dans le backend (essayez de rester accessibles à tous). Deuxième point important, bien montrer les apports pour chaque membre du groupe, mon objectif étant que, grâce à l'effet de groupe, les débutants dans le domaine aient le plus progressé (et non conforter les plus forts) Le contexte de l'application doit bien sûr être défini, mais très rapidement, juste pour que les auditeurs puissent suivre.

## Composition des groupes

4 à 5 IMAC1 par groupe, dont un.e chef.fe de projet

- Maîtrise de Flask/Python et SQL : Fonctionnalité du code (qualité du code : DRY par exemple, pas de TODO, pas de code mort...), organisation claire du code et de ses fonctionnalités, et bdd normalisée
- Côté front: requêtes HTTP fonctionnelles
- Respect de l'architecture REST
- Nombre de fonctionnalités implémentées par rapport au nombre prévu (respect des endpoints/routes)
- Participation du groupe
- Soutenance : forme, aisance, présentation du site en live

La/Le chef.fe de projet sera moins noté.e sur la qualité de son code, en contrepartie iel aura les responsabilités suivantes:

- Iel sera la/le principal.e interlocut.eur.rice lors du suivi de projet
- Iel s'occupera au bon déroulement du projet: bonne communication entre membres du groupe, bonne répartition des tâches, gestion du planning, gestion des possibles conflits...

Concernant le design du site, prévenir qui s'en est chargé, mais attention, le graphisme joue peu (un tout petit peu) dans la note. (faites le pour vous faire plaisir, mais pas pour moi)

# Temps estimé

C'est évidemment très difficile à évaluer : comme c'est assez libre, attention à ne pas partir dans quelque chose de trop ambitieux (vous pouvez, mais seulement si ça vous fait plaisir, et surtout si ça n'empiète pas sur le reste!!)
À titre indicatif, on pourrait imaginer :

- constitution de l'équipe, définition du type de projet, contour : *2 heures*
- Définition des 10 actions, des fonctionnalités, approche un peu plus détaillée : *2 heures*
- Définition du rôle de chacun, première analyse Merise, création des tables, mise en place du serveur : *4 heures*
- 1 tour de code, réalisation de pages de base, et de requêtes de base, vérification : *(si bien réparti entre vous, 8 heures chacun ?)*
- 1ère remédiation : ajout de requête, de service. On ajuste la proposition : *(bien réparti, 8 heures max chacun de votre coté?)*
- 2ème remédiation ; on finalise. Préparation du rapport, et de la soutenance : *(bien réparti, 4 heures chacun, et 4 heures en commun pour synchroniser?)*

Bon, c'est à titre indicatif, mais compter une semaine en tout me semble bien. Il ne faudrait pas dépasser ce temps là ! N 'oubliez pas de décrire justement cette répartition et organisation, afin :
1) de nous expliquer le déroulé,

2) de progresser sur la prévision de la charge de travail.