# Projet : création d'une webapp Streamlit d'analyse de données

**Groupe** : 3 personnes par groupe
**Deadline : 1er décembre à 23h59**

## Contexte

Vous travaillez pour l'entreprise Mangetamain, leader dans la recommandation B2C de recettes de cuisine à l'ancienne bio.

Afin de rester attractive sur le recrutement de profils techs, cette dernière vient de rendre public un extrait de sa base de données.
Votre entreprise souhaite dans la foulée, rendre accessible au grand public une application web interactive contenant une étude poussée réalisée sur ce dataset.
Pour se faire, elle vous choisit, vous, ainsi que deux de ses meilleurs éléments, des data analyst / data scientist émérites.

Mangetamain compte sur vous pour que cette application web soit représentative du niveau technique de ses (meilleurs) employés. En effet, elle rendra également public le code source de cette application web, qui devra donc répondre à un ensemble de critères, des bonnes pratiques de code à la qualité du travail d'équipe (visible sur Git).

## Objectif

L'objectif de ce projet est de mettre en pratique les concepts et les compétences que vous avez appris en cours sur le développement Python pour la production et créer, puis déployer, une application web incluant une partie analyse de données.

Vous allez vous concentrer sur la structure du projet Git, la gestion de l'environnement Python avec Poetry, la programmation orientée objet, la programmation d'une webapp Streamlit, le type hinting, la gestion des logs, le respect des normes PEP 8, la gestion des exceptions, la sécurité, les tests unitaires avec pytest, le test coverage, la documentation avec Sphinx, et la mise en place d'un pipeline CI/CD avec GitHub Actions.

Planification des événements :

- Date début du projet : 26 Septembre
- Cours avec Charlotte : 4 et 25 Octobre
- Deadline : 1er Décembre à 23h59

## Process à suivre

1) **Construire votre Dream team**
   Retrouver vos deux collègues de travail émérites

2) **Récupérer la matière première**
   Télécharger le jeu de donnees que l'entreprise Mangetamain a mis a disposition au grand public
   https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions

3) **Réfléchir avant de miner**
   Conscient que ce dataset vous permet de réaliser différentes analyses pertinentes, Mangetamain a décidé de vous faire pleinement confiance et vous laisse l'opportunité de choisir l'axe d'analyse que vous trouverez le plus intéressant à développer et mettre en avant.
   Ainsi, vous devez identifier une question / un thème qui définissent le fil principal de votre étude, par exemple :
   - le profil des utilisateurs contribuant le plus au site ?
   - Quelles sont les caractéristiques des recettes les plus populaires ?
   - Recommandation de recette en fonction du temps dont on dispose, avec une interface permettant de sélectionner le temps/les ingrédients
   - Visualisation des recettes par réduction de dimension pour déterminer les recettes qui se rapprochent de celles que l'on a déjà faites
   - Générer un nom de recette à partir des ingrédients et des techniques utilisées pour la réaliser
   - Est-ce que les utilisateurs les plus anciens obtiennent de meilleures notes à leurs recettes ?
   - Etc.

   **Cette partie pourra se faire conjointement avec les séances de Charlotte**

4) **Au charbon !**
   Réaliser votre étude en écrivant le code python permettant d'analyser ces données et répondre à vos questions :
   - Sélectionner les variables d'intérêt
   - Manipuler la donnees afin d'extraire les statistiques / insights intéressants
   - Visualiser ces résultats dans des visualisation appropriées et dynamiques (vous pouvez par exemple utiliser Plotly)

5) **L'extérieur de la mine compte aussi**
   Mangetamain ne vous paye pas seulement pour réaliser cette étude, mais attends également de vous que vous puissiez rendre accessible ces résultats sous forme d'une webapp, avec un minimum de storytelling et d'interactivité pour que le grand public puisse se l'approprier et comprendre les tenants et les aboutissants de votre étude.
   Ainsi, vous devez créer une application web en utilisant Streamlit, qui encapsulera votre analyse. N'hésitez pas a mettre des widgets afin d'ajouter de l'interactivité, comme par exemple, la possibilité pour l'utilisateur de changer une valeur dans une

liste pour obtenir un graphique différent.
Mangetamain vous laisse libre court à votre imagination, tant que cette application web en met plein la vue évidemment.

6) **Donner les clefs de la mine**
Mangetamain sera pleinement satisfait lorsque votre webapp sera accessible au monde entier. La concurrence pourra ainsi en juger et lui donner une note (sur 20 !). Pour ce faire, vous pouvez déployer votre application sur Streamlit Cloud, AWS, Azure, GCP, Heroku ou sur un serveur à vous.

## Exigences techniques

Le repository Git devant être publique sur Github pour le rayonnement de l'entreprise (et le vôtre !), un minimum de qualité est attendu.

La gestion du projet

- **Structure du projet** : organisez votre projet en respectant une structure cohérente. Utilisez des **packages** et des **modules** pour diviser votre code en composants logiques. Vous pouvez utiliser **Visual Studio Code** si vous n'avez pas encore d'IDE préféré.
- **Environnement Python** : utilisez un **gestionnaire d'environnement** Python ou **Poetry** pour gérer les **dépendances** de votre projet. Choisissez bien votre **version** de Python. Assurez-vous d'avoir un fichier requirements.txt ou pyproject.toml correctement configuré.
- **Git** : initialiser un **dépôt Git** pour votre projet et suivez les meilleures pratiques de gestion de code avec des **commits** (assurez-vous de committer régulièrement), et dans la mesure du possible, des **branches** et des **Pull Request** pour travailler en équipe. Assurez-vous d'inclure un fichier **README.md** qui explique comment installer, exécuter, déployer et utiliser votre application.
Optionnel : créez éventuellement des **tags** de version pour marquer les **versions** stables de votre application.
- **Streamlit** : développer votre webapp avec une expérience utilisateur (UX) simple et intuitive, en laissant à l'utilisateur la possibilité d'interagir avec vos données pour bien comprendre la storytelling que vous lui raconterez. Cette storytelling doit comporter des insights au travers de graphiques (charts, etc.) et doit répondre à votre problématique / question initiale.

La programmation

- **Programmation orientée objet** : dans la mesure du possible, utilisez le paradigme orienté objet. Utilisez les principes de l'**encapsulation** et de l'**héritage** si approprié. Utilisez également les bonnes **structures de données**.
- **Type Hinting** : utilisez des annotations de type pour améliorer la lisibilité de votre code.

- **PEP 8** : assurez-vous que votre code respecte les normes PEP 8 pour la lisibilité et la cohérence du code. Utilisez un formateur (par exemple black)
- **Gestion des exceptions** : gérez les erreurs de manière appropriée en utilisant des exceptions personnalisées lorsque nécessaire. Par exemple en cas de saisie incorrecte de l'utilisateur.
- **Logger** : utilisez le module logging pour enregistrer les actions de l'utilisateur et les événements importants dans un fichier de log. Créer un fichier de log pour le **debug**, et un autre pour les **erreurs** (ERROR et CRITICAL).
- **Sécurité** : assurez-vous (un minimum) que les bibliothèques que vous utilisez sont connues et n'ont pas de vulnérabilités de sécurité évidentes. Si vous autorisez une entrée utilisateur, ne pas utiliser la fonction `eval`, évitez les mots de passe/token en clair dans le code, etc.
-

Les tests

- **Tests unitaires** : écrivez des tests unitaires approfondis pour chaque composant de votre application en utilisant **pytest**. Vérifiez que la logique de votre application fonctionne correctement.
- **Test coverage** : utilisez un outil de test coverage (comme **pytest-cov**) pour mesurer la couverture de vos tests et assurez-vous d'avoir une couverture suffisante (**90%** de couverture minimum).

La documentation du projet

- **Commentaires :** assurez-vous d'inclure des commentaires pertinents dans votre code pour expliquer la logique complexe ou les décisions de conception importantes.
- **Docstrings :** utilisez des docstrings pour documenter vos classes, méthodes et fonctions de manière détaillée, en expliquant leur but, leurs paramètres et leurs valeurs de retour. Vous pouvez utiliser la **convention** qu'il vous plaira : Google, NumPy ou reStructuredText (reST).
- **Documentation** : créez une documentation claire et concise pour votre application en utilisant **Sphinx**. Documentez les classes, les méthodes, et expliquez comment installer et utiliser votre application.

La CI

- **Pipeline CI/CD** : configurez un pipeline de CI avec **GitHub Actions** pour checker que **pep8** est bien respecté, que les **docstrings** sur les fonctions / méthodes, classes, modules sont bien présentes, pour **automatiser les tests** et vérifier que le **test coverage** est supérieur à **90%** du code. Les tests unitaires doivent être exécutés automatiquement à chaque push sur une branche en review, et lors du merge de la branche en review sur master.
  Optionnel : inclure votre phase de déploiement de l'application dans votre CI/CD

## Livraison du projet

Vous devez soumettre votre projet sous la forme d'un **dépôt GitHub public**, incluant :

- Code source Python bien structuré
- Documentation générée avec Sphinx
- Fichiers de logs
- Tests unitaires vérifiant le bon fonctionnement de l'application
- Le pipeline CI/CD
- Le lien vers votre webapp Streamlit déployée

Avant la deadline, envoyez le lien de votre projet GitHub à : prillard.martin@gmail.com

## Évaluation

Votre projet sera évalué en fonction de la qualité du code, du respect des bonnes pratiques, de la couverture de tests, de la documentation, de la qualité visuelle de la webapp (graphiques, pertinences des insights, etc.) et du bon fonctionnement de l'application.

## Conseils

- N'hésitez pas à vous répartir le travail et à fusionner votre code sur Git avec des Pull Request
- Faites **simple** au début dans les fonctionnalités pour mettre en place le tout, puis itérer avec d'autres fonctionnalités plus complexes
- Utiliser **tout ce que vous voulez**, Internet, ChatGPT, Codium, Copilot, etc. tant que vous **maîtrisez** votre code

## Pour aller plus loin (bonus)

- **Utilisation d'une base de données** : comment pourriez-vous migrer votre système de stockage de données vers une base de données SQL ou NoSQL pour gérer un volume plus important de tâches et améliorer les performances ?

- **Optimisation des performances** : comment pourriez-vous optimiser les performances de votre bibliothèque, en particulier lors de la gestion de grandes quantités de données ?

- Liste non exhaustive. Libre à vous d'améliorer davantage votre web-app !