Sujets

sujet 1 : Réseau basé sur les goûts musicaux 🎵

Tuteur : Cyriel Mallart

Présentation

Tu veux parler musique avec des gens qui comprennent tes goûts ? Avec (NomDeVotreWebService), c'est facile ! Enregistre-toi, dis-nous ce que tu écoutes, et on t'envoie des gens qui vibrent au même rythme que toi !

Vous allez créer dans ce projet une API qui met en relation des gens, basée sur leurs goûts musicaux. Un.e utilisateur.ice s'inscrira, sélectionnera et enregistrera ses morceaux, artistes et genres préférés. L'API Deezer vous donnera accès à toutes les caractéristiques de ces morceaux. A partir de cela, vous utiliserez une touche d'IA pour trouver celui ou celle qui a le profil le plus similaire, et lui enverrez une petite notification pour mettre tout le monde en contact. Si ça ne matche pas, ou si l'utilisateur.ice veut découvrir de nouvelles personnes, on trouve le profil similaire le plus proche en excluant les gens déjà en contact ou refusés.

Fonctionnalités de base (attendues) :

- gestion de plusieurs utilisateurs
- enregistrement des morceaux, artistes, genres, etc. utiles en base de données
- appels à l'API Deezer pour obtenir les informations sur les morceaux
- obtention, stockage et mise à jour des profils similaires
- gestion des notifications de mise en contact : accepté, refusé, notification à tous les utilisateur.ices impliquée.e.s

Fonctionnalités avancées (une au choix si les fonctionalités de base sont finies):

- authentification par mot de passe
- gestion des "contacts" d'un utilisateur : déletion, signalement, création de groupes
- recommendation de nouveaux morceaux basés sur ceux des contacts
- playlists crées par fusion des morceaux des utilisateurs, export
- sélection ergonomique des morceaux (suggestions, autocomplétion, ...)
- algorithme de recommendation plus élaboré
- ...
- tout autre idée qui vous semble logique, intéressante, sous réserve qu'elle soit faisable techniquement dans les temps impartis

Stack technique :

- Backend : FastAPI
- Front end : Jinja2 (templates HTML dans FastAPI). D'autres choix sont possibles (React.js, Vue.js, ...) si vous les maîtrisez, mais attention, votre tuteur ne pourra vous suivre que sur React.js, et la courbe d'apprentissage est raide.
- BDD : MongoDB (NoSQL) , Neo4j (NoSQL, graphe) ou Postgres (SQL, disponible à l'ENSAI)
- API Deezer : https://developers.deezer.com/api (https://developers.deezer.com/api)

Fausses bonnes idées

- Créer une appli de messagerie : le but de ce projet est uniquement une mise en contact où l'on partagera une adresse mail par exemple
- Se casser la tête sur un algorithme super-classe : les performances de l'algorithme de recommendation ne sont pas le point important ici, ne passez pas tout votre projet dessus. Si vous avez envie d'essayer une idée d'algorithme, allez-y, même si elle ne fonctionne pas bien au final. Sinon, votre tuteur pourra vous aider sur un algorithme de base
- Une interface réctive super-évoluée avec des graphiques et des previews de morceaux : on attend une interface basique et simple, c'est-à-dire, des pages HTML statique écrit en noir sur fond blanc avec un ou deux boutons.

sujet 2 : Aide à la recherche de stage 📰💰🏷️

Tuteur : Colin Leverger, sujet 2023

Objectif

Vous êtes un groupe d'étudiants de l'ENSAI en deuxième année, et souhaitez développer une application en ligne de commande pour faciliter vos futures recherches de stage. Votre startup s'appelle ... et votre application ... .

Votre application permettra à un utilisateur de rechercher des stages de son choix sur un ou plusieurs sites de recherche de stage (Hellowork, welcometothejungle, ...). Les stages peuvent être publiés dans différentes spécialités (informatique, data analyse, machine learning, ...). S'il est authentifié, l'utilisateur pourra sauvegarder dans l'application les stages qui l'intéressent dans sa « liste d'envies », et pourquoi pas répondre aux offres plus facilement (automatiquement, ou au minimum récupérer les contacts vers les employeurs). La navigation et les recherches de tout utilisateur seront historisées.

Deux types d'utilisateurs authentifiés pourront cohabiter : utilisateur « élève » et utilisateur « professeur ». Un « administrateur » de l'app pourra également gérer l'application et les utilisateurs. Un utilisateur non authentifié sera par défaut dans la catégorie « élève ». On peut imaginer que les besoins des professeurs et des élèves seront différents...

L'application permettra aux utilisateurs authentifiés de gérer leurs comptes, préférences, mots de passe, listes, et à tous les utilisateurs d'exporter/importer leurs recherches courantes (dans le format texte que vous jugerez bon).

Les utilisateurs pourront également géo localiser les stages par rapport à leur position actuelle (temps de trajet,...).

Fonctionnalités requises (numérotées, mais non ordonnées)

- F1 : la recherche de résultats sur un site que vous choisirez (Hellowork, welcometothejungle, ...)
- F2 : authentification et gestion du profil utilisateur/profil administrateur
- F3 : gestion de l'import/export de données au format choisi
- F4 : gestion de l'historique des recherches
- F5 : recherche par catégorie/filtre de recherches

Fonctionnalités optionnelles

- FO1 : lancer des recherches sur plusieurs sites en parallèle
- FO2 : géo localisation de l'utilisateur et des annonces/distance entre l'utilisateur et l'annonce
- FO3 : alertes automatiques si de nouvelles annonces de stage remplissant les critères sont publiées
- FO4 : ... à vous de jouer !

  NOTE : ce sujet regorge de « ... ». En effet, une participation active est vivement recommandée et vos idées, pour rendre votre projet unique, sont les bienvenues !

Fonctionnalités non notées (et déconseillées)

- les statistiques/modèles complexes que vous pourrez imaginer (une moyenne, OK, mais un modèle expo logarithmique quantique d'ordre 10, non)
- l'interface graphique type GUI (AUCUN point bonus sur une très jolie interface !)

sujet 3 : GreenStream : API d'aide à la réduction de l'impact carbone de la VOD 🌿

Tuteur : Maxence Lagalle

Présentation

GreenStream est une API REST qui permet de réduire l'impact carbone d'un service de VOD (Netflix, Amazon Prime, Disney+...). Pour le consommateur, elle calcule l'impact carbone d'une vidéo à partir de sa durée et de sa résolution, en utilisant le modèle "1byte" de The Shift Project (https://theshiftproject.org/article/climat-insoutenable-usage-video/) ou d'autres modèles d'estimation. Pour le fournisseur de VOD, elle aide à choisir des serveurs dans la meilleure zone géographique des fournisseurs Cloud en fonction de l'impact carbone de la production d'électricité. Ces données sur l'impact carbone sont fournies par l'API ElectricityMaps (https://static.electricitymaps.com/api/docs/index.html). GreenStream est un projet innovant et écologique, qui accompagne dans la transition vers des services de VOD plus vert et plus responsables.

ElectricityMaps (https://www.electricitymaps.com) est partenaire de GreenStream et offre un accès à la version payante de son API pour toute la durée du projet.

Fonctionnalités de base

- **Estimation de l'empreinte carbone d'une vidéo** selon le modèle "1byte" de The Shift Project
  - Prise en compte de la localisation géographique du fournisseur de VOD et de l'utilisateur
  - Prise en compte de la durée et de la qualité de la vidéo, du type de connexion et du matériel utilisé pour regarder la vidéo
  - Utilisation des données d'impact carbone en temps réel fournies par l'API ElectricityMaps
- **Recommandation du meilleur service Cloud** pour diffuser une vidéo à un utilisateur
  - Choix du service Cloud à l'impact carbone le plus faible en fonction de la zone géographique de l'utilisateur
  - Détermination des services Cloud éligibles en fonction de la localisation de l'utilisateur (ceux dans le même pays ou un pays limitrophe)
  - Utilisation des données de prévision de l'impact carbone pour la durée de la vidéo fournies par l'API ElectricityMaps
- **Gestion de l'offre des fournisseurs Cloud**
  - Implémentation de la liste des zones géographiques proposées par au moins deux fournisseurs Cloud (AWS, GCP, Azure...)
  - Utilisation d'une base de données SQL pour stocker les informations sur les services Cloud disponibles
- **Information sur l'état du service GreenStream**
  - Fourniture d'un service permettant de s'assurer que GreenStream fonctionne correctement et que son lien avec ElectricityMaps est actif

Fonctionnalités avancées

- **Simulation de l'impact carbone d'un changement de comportement**
  - Création d'un service qui permet à l'utilisateur de simuler l'impact carbone qu'il aurait s'il changeait certains paramètres de sa consommation de vidéos, tels que la qualité, la durée, le type de connexion ou le matériel utilisé
  - Utilisation du modèle "1byte" ou d'autres modèles d'estimation pour calculer l'empreinte carbone selon les différents scénarios

- **Utilisation de modèles d'estimation de l'empreinte carbone plus sophistiqués**
  - Prise en compte d'une consommation électrique variable selon les fournisseurs Cloud
  - Utilisation d'autres modèles d'estimation publiés, tels que le modèle "Carbon Footprint of Video Streaming" de Carbon Trust (https://www.carbontrust.com/our-work-and-impact/guides-reports-and-tools/carbon-impact-of-video-streaming) ou d'autres modèles à rechercher
- **Intégration de critères avancés dans la recommandation du service Cloud**
  - Prise en compte du coût financier des services Cloud et d'un arbitrage entre le coût et l'empreinte carbone
  - Utilisation d'un graphe de connexions entre les zones pour déterminer la liste des services Cloud éligibles pour diffuser une vidéo à un utilisateur
- **Suivi de la consommation totale du service et de son empreinte carbone**
  - Enregistrement SQL de la consommation de vidéos : durée, zone géographique de l'utilisateur, service Cloud associé, empreinte carbone
  - Accès aux données brutes ou à une synthèse statistique via l'API
- **Fonctionnement en mode dégradé**
  - Utilisation de données historiques d'intensité carbone de la production d'électricité en cas de défaillance de l'API ElectricityMaps

L'initiative est fortement encouragée dans ce projet, et d'autres idées de fonctionnalités avancées ou d'amélioration des fonctionnalités de base peuvent être proposées par les élèves.

sujet 4 : Prix du carburant🛢️💰

Tuteur : Thierry Mathé

Objectif

Le ministère des Finances met à disposition un fichier des prix des carburants dans les stations françaises. Ce fichier au format XML contient entre autre pour chaque station, ses coordonnées, son adresse, ses heures d'ouvertures, le prix des carburants disponibles (mais l'enseigne de la station n'y figure pas).
Le but du projet de créer une API (application programming interface) qui exploite les données contenues dans ce fichier. Cette API devra permettre de :

- Consulter la liste des stations se trouvant à proximité d'un point donné il sera aussi possible de préciser le carburant.
- Créer et mettre à jour des listes de stations qu'il sera possible de consulter par exemple pour voir les prix sur les stations se trouvant sur un trajet régulier.

A chaque requête, l'API doit s'assurer qu'elle utilise bien les dernières données disponibles et au besoin télécharger les nouvelles données disponibles sur le site (mises à jour toutes les 10 minutes) : https://donnees.roulez-eco.fr/opendata/instantane (https://donnees.roulez-eco.fr/opendata/instantane)

Fonctionnalités de base

Stations à proximités

Les paramètres donnés à cette requête sont les coordonnées (longitude et latitude) d'un point, une distance en km et un type de carburant. L'API va alors extraire l'ensemble des stations se trouvant dans la zone ainsi définie et disponsant du carburant indiqué.

La réponse sera données au format JSON et contiendra:

- les parmètres de la requête
- la date et l'heure d'exécution
- le nombre de stations trouvées
- la liste des stations : pour chaque station on aura:
  - l'id, les coordonnées et l'adresse de la station
  - le prix du carburant

Stations préférées

La gestion des listes de stations doit permettre de:

- Créer une liste: le paramètre en entrée est un nom donné par l'utilisateur. La requête revoie un identifiant qui devra être passé comme paramètre pour toutes les autres actions,
- Consulter la liste des listes crées: aucun paramètre, la requête renvoie les couple identifiant-nom des listes défines,
- Ajouter une une plusieurs stations à un liste: les paramètres en entrée sont l'identifiant de la liste et le ou les identifiants des stations à ajouter,
- Supprimer une liste: le paramètre à donner est l'identifiant de la liste,
- Consulter la liste: le paramètre à donner est l'identifiant de la liste. La requête revoie les informations au format JSON qui contiennent:
  - l'identifiant et le nom de la requête
  - la date et l'heure d'exécution
  - le nombre de stations contenues dans la liste
  - la liste des stations. Pour chaque station on aura:
    - l'id, les coordonnées et l'adresse de la station
    - le prix du carburant

Fonctionnalités avancées

Voici quelques idées de fonctions avancées pour les groupes qui auraient effectué l'ensemble des fonction de bases. Toutes autres idées pourra bien sûr être proposées en cours de projet.

Stations à proximités

- Possibilité de passer comme parmètre une adresse au lieu de coordonnées
- Possibilité de préciser l'heure de passage et de ne conserver que les stations ouvertes à cette heure.

Station préférées

- Possilité de retirer une ou plusieur stations d'une liste
- Imposer une identification pour consulter les listes. Chaque utilisateurs n'aurait alors accés qu'à ses liste.

sujet 5 : Outil statistique d'analyse des parties de League of Legends 🎮

Tuteur : Aloïs DE OLIVEIRA

Contexte

League Of Legends (connu sous le sigle LoL) est un jeu video, plus précisément un Multiplayer Online Battle Arena (MOBA), développé par Riot. LoL comptait plus de 100M de joueurs actifs en 2022 et le chiffre n'est pas voué à diminuer dans les prochaines années.
Cet engouement force Riot à se renouveler en créant de nouveaux champions, de nouveaux items, des ajustements sur l'équilibrage du jeu (appelés patch). Les joueurs apprécient donc fortement les outils leur permettant d'avoir des analyses simples mais rapides afin de les accompagner et aider dans leurs choix.

Ce sujet ne nécessite pas de connaître le jeu ou d'y avoir déjà joué. Au contraire, il vous arrivera souvent de devoir dans un premier temps comprendre un nouveau contexte métier et vos données en entrée.

Sujet

Objectif

Vous êtes mobilisés afin de mettre en place une solution capable de fournir aux joueurs les premières informations essentielles.
Ces analyses seront simples mais nécessiteront d'analyser tout l'historique des parties que vous aurez à votre disposition, et donc que vous stockerez.
Votre solution a pour but d'être pensée dans une logique d'amélioration continue. Ainsi un **code documenté utilisant les principes de la Programmation Orientée Objet (POO)** est attendu.

Les données

LoL met à disposition une API afin de permettre aux personnes qui le souhaitent de pouvoir utiliser leurs données. Une brève explication du fonctionnement de l'API de LoL (https://developer.riotgames.com/) sera effectuée mais vous aurez à votre disposition un ensemble de données déjà collectées.

Fonctionnalités

Lors de ce projet, vous aurez des fonctionnalités obligatoires et avancées. **Il est nécessaire de se focaliser sur ces premières et d'entreprendre les secondes que lorsque vous avez une base solide pour répondre à l'ensemble des fonctionnalités obligatoires**.

Fonctionnalités obligatoires

Niveau base de données

FO1 : Création de votre base de données :

- Création d'une base de données de votre choix (SQL/NoSQL) qui vous servira à stocker l'historique des parties. Le modèle de données est important, il doit être adapté afin de faciliter toutes modifications.
- Création d'un pipeline de données. L'ensemble des étapes de la collecte au chargement, en passant par la transformation doivent être le plus réutilisable possible.
- L'ensemble des tâches doit être effectué en Python (de la création, à l'insertion et aux éventuelles modifications).

Niveau API

Pour la réalisation de ce projet, vous devrez réaliser une API (à l'aide FastAPI (https://fastapi.tiangolo.com/) ou Flask (https://flask.palletsprojects.com/en/2.3.x/)) qui vous permettra de renvoyer les différents résultats souhaités. Ces résultats seront à retourner sous format json.
Il existera différentes classes, décrites ci-dessous, avec chacune ses méthodes. Toutes les méthodes présentes dans les descriptions des classes sont à titre indicatif. N'hésitez pas à être force de proposition.

FO2 : Proposer des statistiques de base avec une classe invité :

- Cela ne nécessitera aucune connexion.
- Cette classe devra comporter certaines méthodes permettant de fournir des statistiques globales sur le jeu, telles que :
  - Afficher les statistiques d'un champion : le nombre de games jouées, le winrate, le gold lead à Xmin, etc.
  - Trier les champions (globalement ou par lane) selon le critère au choix parmi : nombre de games jouées, le winrate, etc.

FO3 : Proposer des statistiques plus précises avec une classe nécessitant une connexion :

- Cela nécessitera de se connecter à un compte user.
- Cette classe héritera de l'ensemble des méthodes propres à la classe invité.
- Cette classe devra comporter certaines méthodes permettant de fournir des statistiques liées au compte en question, telles que :
  - Afficher les statistiques globales/par lane/par champion du compte : le nombre de games jouées, le winrate, le kda, etc.

FO4 : Gestion de l'ensemble des données (user et LoL) à l'aide d'une classe admin :

- Cela nécessitera de se connecter à un compte admin. 1 seul sera nécessaire.
- Cette classe héritera de l'ensemble des méthodes précédentes.
- Cette classe aura pour but de permettre à l'admin de pouvoir gérer les comptes et les données.

Fonctionnalités avancées

Les fonctionnalités avancées sont triées selon le niveau de difficulté , cependant vous êtes totalement libres sur les fonctionnalités avancées que vous souhaitez réalisées.

FA1 : Ajout de nouvelles données quotidiennes :

Jusqu'ici vous utilisez les données mises à disposition sans enrichissement. Cette fonctionnalité aura pour but de permettre à l'admin de pouvoir ajouter des nouvelles parties récentes.
Les différents points attendus sont :

- Gestion de la clé (permettant l'accès à l'API de Riot) : au choix entre en brute en paramètre d'entrée ou permettre d'aller la récupérer directement sur le site depuis l'API.
- Récupération des données depuis l'API de Riot selon certains critères de sélection.
- Ajout des données dans vos bases de données.
- Ajout de la fonctionnalité au sein de votre API.

Les critères de sélection seront libres et donc proposés par votre groupe, en fonction de ce que l'API vous permettra de faire.

FA2 : Mise en place d'un workflow collaboratif :

Le workflow Gitflow vous permet de faciliter la gestion des branches et la séparation des différentes tâches.
Utilisez le framework Gitflow (https://www.atlassian.com/fr/git/tutorials/comparing-workflows/gitflow-workflow) pour la gestion de votre projet.

FA3 : Conteneurisation de l'application :

Conteneuriser votre application aura pour but de faciliter son déploiement et son partage. En effet, cela permettra de réunir le code et ses composants (frameworks, bibliothèques, etc) dans un conteneur. Il sera alors possible d'exécuter le conteneur sans se soucier des différentes dépendances (l'installation des bibliothèques avec la bonne version par exemple).

Utilisez l'outil Docker (https://fastapi.tiangolo.com/deployment/docker/) pour conteneuriser votre application.

FA4 : Mise en place d'un modèle prédictif :

Pour le moment, vous ne faites remonter que des statistiques descriptives. Ici il s'agit de mettre à profit l'ensemble des données disponibles pour permettre à l'utilisateur de prédire l'équipe gagnante en fonction des 5 champions de chaque équipe.

L'enjeu de cette fonctionnalité n'est pas de vous faire trouver le meilleur modèle de prédiction mais d'ajouter une fonctionnalité à votre API basé sur ce modèle. Vous ne serez donc aucunement noté sur la qualité des prédictions.

Les différents points attendus sont :

- Choix d'un modèle de prédiction : vous ne serez certes pas noté sur la qualité des prédictions mais une courte réflexion sur le choix du modèle est attendue.
- Entrainement dudit modèle : vous êtes libres sur le choix des librairies.
- Sérialisation du modèle : vous devez sauvegarder votre modèle.
- Ajout de la fonctionnalité au sein de votre API.

sujet 6 : Mangez mieux ! 🍓 🥩 🍬

Tuteur : Sophie HERBERT

Objectif :

Recommander des produits meilleurs pour la santé selon plusieurs critères !

Vous allez explorer la célèbre API Open Food Facts qui regroupe des produits alimentaires divisés en plusieurs catégories (Viandes, Snacks, Aliments d'origine végétale etc.) et pour lesquels nous avons un grand nombre de variables renseignées par la communauté : nutriscore, nova score, ecoscore, taux de protéine, taux de sucre, énergie en kcal...
L'objectif de l'application est d'utiliser un échantillon de ces données en vous concentrant sur quelques produits de quelques catégories que vous aurez choisis, de les afficher tel un catalogue et de pouvoir les trier selon le critère sélectionné : le moins de sucre possible, le meilleur eco-score, la plus faible valeur énergétique... En allant sur une fiche « Produit », vous afficherez le meilleur produit de la même catégorie selon ce critère.
Un système d'identification devra être mis en place. Si l'utilisateur est identifié, pourquoi pas sauvegarder ses articles dans un panier virtuel ?

Les fonctionnalités requises :

Les fonctionnalités requises :

- F1 : affichage simple en console d'un catalogue de produits alimentaires par catégorie de produit
- F2 : ordonner les produits selon un critère nutritionnel
- F3 : sélection d'une fiche « Produit » recommandant un autre produit de la même catégorie avec un critère meilleur que le produit consulté (si meilleur il y a)
- F4 : système d'authentification et de gestion du profil (changement de mot de passe...)

Et en option :

- F5 : sauvegarde de produits dans un panier

- …
  Des fonctionnalités optionnelles pourront être développées à votre guise et selon votre inspiration sur le sujet ! Les données Open Food Facts étant très riches, d'autres axes d'analyses peuvent être creusés…
  Note : l'interface GUI, la mise en place de modèles statistiques, la gestion des valeurs manquantes ne seront pas inclues dans la notation.

Sujet 7 : JobHub, Simplifier la Recherche d'Emploi dans le Secteur de la Tech

Tuteur : Mansor GUEYE

Description du Projet :

Ce projet a pour objectif de créer une application qui facilite la recherche d'emploi dans le secteur de la technologie en utilisant l'API REST gratuite de Adzuna (https://developer.adzuna.com/) , qui fournit des informations sur les offres d'emploi dans divers secteurs à travers le monde. L'application permettra aux utilisateurs de rechercher des emplois, de filtrer les résultats, de suivre leur progression dans leurs candidatures et de recevoir des alertes pour les nouvelles offres correspondant à leurs compétences.

Objectifs du Projet :

Intégrer les données de l'API Adzuna dans une application conviviale.
Permettre aux utilisateurs de rechercher des offres d'emploi par mots-clés, localisation et catégories.
Fournir des fonctionnalités de suivi des candidatures, y compris la gestion de CV et de lettres de motivation.
Envoyer des alertes aux utilisateurs pour les nouvelles offres correspondant à leurs critères.
Mettre en place un système de profil pour les chercheurs d'emploi.

Les fonctionnalités requises :

Les fonctionnalités requises :
F1 : Intégrer les données de l'API Adzuna pour afficher des offres d'emploi dans divers secteurs.
F2 : Permettre aux utilisateurs de rechercher des emplois en utilisant divers critères.
F3 : Permettre aux utilisateurs de créer des profils de chercheurs d'emploi.

Et en option :

F4 : Fournir des fonctionnalités de suivi des candidatures et de gestion de documents(CVs, lettres de motivation…).
F5 : Envoyer des alertes aux utilisateurs pour les nouvelles offres correspondant à leurs compétences.
F6:…

❗ Vous avez la liberté de développer des fonctionnalités optionnelles selon votre inspiration. Aucune interface graphique n'est demandée dans ce sujet, toute interaction avec l'application devra se faire via l'invité commande (terminal / cmd). Si vous souhaitez tout de même le faire, cela ne saurait se substituer aux fonctionnalités demandées.

Sujet 8 : Velib' Hunter 🚲🔪

Tuteur : Samuel GOUTIN

Objectif :

Ce projet vous propose d'explorer les données mises à disposition par OpenData Paris. En particulier, vous allez jouer avec les données présentant la disponibilité des Velib' en temps réel (https://opendata.paris.fr/explore/dataset/velib-disponibilite-en-temps-reel).
La solution proposera des services pour permettre aux utilisateurs de trouver plus facilement un vélo, ou pour aider la ville de Paris à mieux gérer son parc de vélos.
Pour cela, vous aurez besoin de capturer et de stocker les données disponibles sur l'API d'OpenData Paris (et leur historique!) dans une base de données, puis de créer à votre tour une API pour y mettre à disposition vos services.

Fonctionnalités obligatoires

- **F1** : obtenir le nom de la station la plus proche ayant au moins un vélo disponible à partir de coordonnées géographiques.
- **F2** : obtenir le nom de la station la moins fréquentée sur une période de temps données.
- **F3** : obtenir le numéro de l'arrondissement le plus fréquenté sur une période de temps données.

Fonctionnalités optionnelles

- **F01** - *Recherche de vélo en temps réel* : Un utilisateur utilise la F1 pour trouver un vélo proche de lui. En arrivant sur place: mauvaise surprise, le vélo lui ai passé sous le nez. Pour éviter ce genre de désagréaments, le nom de la station la plus proche sera actualisé en temps réel. Pour cela, vous pouvez améliorer la F1 en proposant une connexion basé sur un WebSocket.
- **FO2** - *Déploiement* : Pour faciliter le déploiement de votre solution, conteneurisez-la en utilisant Docker Compose (https://wiki-tech.io/Conteneurisation/Docker/Docker-Compose).

Les outils

Pour répondre aux fonctionnalités demandés, vous aurez certainement besoin de vous armer de:

- une librairie pour construire des APIs (conseillé: FastAPI)
- une solution de base de données (conseillé: SQLite)
- un outil de versioning de code en équipe (Gitlab ou Github)