SAE : Développement d'une Application Vélib

« Groupe Paris »

Réalisé par :

- Oumaima EL-KHADRAOUI (leader) : ***Stymphale***
- Tassadit OUZIA : ***Crete***
- Kenza HALIL : ***Stymphale***
- Yannel AISSANI : ***Dyomède***
- Youssouf REZZAG-MAHCENE : ***Dyomède***

Encadré par :

- Mme Hanane AZZAG
- M. Bouchaib LEMAIRE

Nom de l'établissement :

- IUT de Villetaneuse, Université Sorbonne Paris Nord

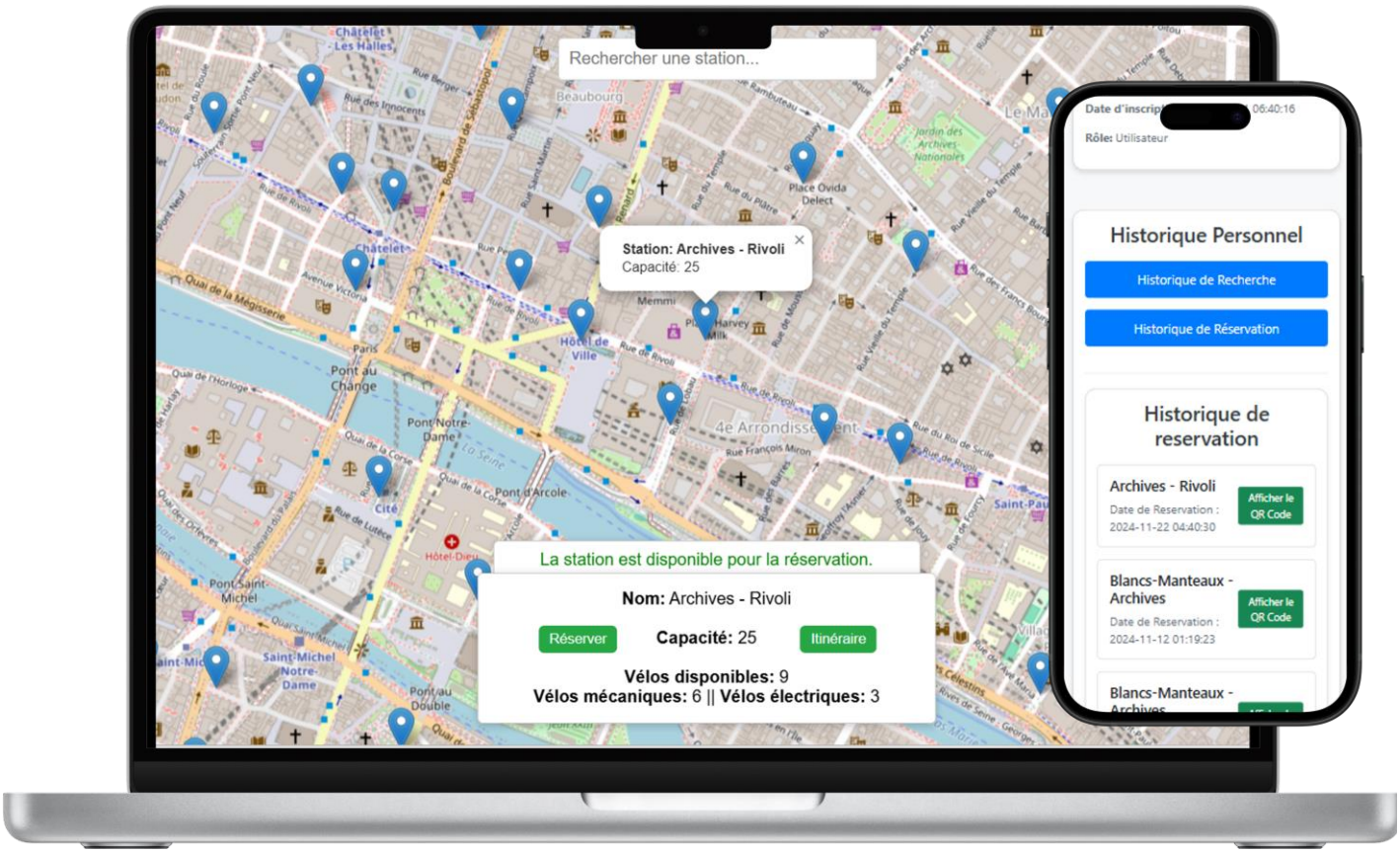
Année Académique :

- 2024/2025

Date de Remise : 27/11/2024

Livrable 1 : Scripts d'administration

« Importation des données disponible en OPENDATA »



« Application web en cours de développement par le groupe PARIS »

Table des Matières :

1) Introduction :

Ce projet SAE a pour objectif de traiter et exploiter les données en Open Data fournies par le service Vélib' Métropole, un réseau de vélos en libre-service. Le service Vélib met à disposition des données en temps réel sur l'état des stations de vélos : nombre de vélos disponibles, bornes de recharge libres, ainsi que des informations géographiques telles que la latitude et la longitude de chaque station. Ces données sont cruciales pour les utilisateurs du service, mais elles sont également utiles pour le développement d'applications et de services tiers.

Les données sont déjà proposées en format JSON, un format léger et facile à manipuler qui est parfaitement adapté aux besoins des applications web modernes. Ce format est compatible avec les technologies actuelles utilisées pour la gestion de données dans les applications, et il permet une intégration rapide et efficace des informations.

L'objectif principal de cette première partie du projet est de récupérer ces données disponibles en Open Data, de les nettoyer et de les organiser pour faciliter leur exploitation. Une fois ces données traitées, nous les exposerons via une API qui permettra à notre application de consulter ces informations en temps réel.

Cette première partie du projet requiert non seulement une gestion efficace des données, mais aussi une collaboration étroite au sein de notre équipe afin de diviser les tâches, choisir les bons outils et garantir une mise en œuvre réussie.

2) Analyse du Besoin :

Dans ce premier livrable, nous visons à exploiter en temps réel les données de Vélib, disponibles en format JSON. Ces données comprennent des informations essentielles sur les stations de Vélib', telles que la disponibilité des vélos, le nombre de bornes disponibles, et d'autres paramètres critiques pour le bon fonctionnement du service.

Le format JSON, bien qu'idéal pour le stockage et l'échange de données structurées, présente de nombreux avantages dans le cadre des applications modernes. Il est léger, rapide à traiter et facile à manipuler dans les technologies

web actuelles, telles que JavaScript et Python. Ce format est particulièrement adapté aux systèmes de données dynamiques, où les informations changent fréquemment, comme c'est le cas avec les stations Vélib', où la disponibilité des vélos et des bornes peut fluctuer d'une minute à l'autre.

En résumé, il est essentiel de transformer, nettoyer et structurer les données pour répondre aux exigences de fluidité, de rapidité et de fiabilité du projet. Ce processus préparera également les données pour une intégration réussie dans une API, qui permettra d'exposer les informations de manière accessible à notre application web.

3) Gestion du Projet et Répartition des Tâches :

Pour ce premier rendu, notre équipe a mis en place une gestion de projet claire et structurée afin de garantir une répartition efficace des tâches. Chaque membre a été assigné à des responsabilités spécifiques en fonction de ses compétences, des besoins du projet et des préférences de chacun. Voici comment les tâches ont été réparties :

1. Oumaima (Chef de projet) :

En tant que leader du groupe, Oumaima a supervisé l'ensemble du projet. Elle a assuré la coordination entre les membres de l'équipe, a veillé au respect des délais et a facilité la communication au sein du groupe et aussi elle a eu le droit de faire des ajustements sur le fichier README rédiger par Yannel.

2. Yannel (Importation des données) :

Yannel a été responsable de l'importation des données. Il a développé le script permettant de télécharger les données depuis la plateforme Open Data de Vélib en utilisant des requêtes HTTP avec Python. Yannel a également rédiger le fichier README du scripte.

3. Youssouf (Nettoyage et préparation des données) :

Youssouf a pris en charge le nettoyage des données après leur importation. Il a vérifié les valeurs manquantes et améliorer le scripte pour garantir que les données étaient prêtes à être exploitées dans l'API.

4. **Kenza (Répartition des tâches et suivi) :**

Kenza a été responsable de la création du fichier Excel de répartition des tâches. Elle a conçu un tableau détaillant les responsabilités de chaque membre de l'équipe, les délais et les étapes de suivi. Ce fichier a permis de suivre l'avancement du projet de manière efficace et de s'assurer que les délais étaient respectés et aussi elle a eu le droit de faire des ajustements sur le fichier du rapport rédiger par Tassadit.

5. **Tassadit (Rédaction du rapport et suivi des tests) :**

Tassadit a pris en charge la rédaction du rapport, en détaillant chaque étape du projet, ainsi que l'évaluation des performances du système. Elle a aussi veillé à l'intégration des tests pour garantir la fiabilité du scripte avant de l'utiliser définitivement dans notre application web.

4) Réunions et Organisation du Travail :

Pour assurer la réussite de ce projet, une organisation rigoureuse et une bonne coordination au sein de notre équipe étaient primordiales. Étant donné que nous venons de différentes classes et que nos emplois sont par conséquent différents, nous avons dû faire preuve de flexibilité pour nous réunir efficacement et avancer ensemble. Nous avons donc choisi de privilégier des réunions informelles pendant les pauses déjeuner, où nous pouvions discuter de l'avancement du projet et prendre des décisions importantes. En complément, nous avons organisé plusieurs visioconférences pour les points plus techniques, afin d'assurer une meilleure communication et une gestion optimale du travail à distance.

4.1 Réunions principales et prise de décision :

Lors de notre première réunion, nous avons discuté des différents langages et technologies à utiliser pour le développement du projet. L'objectif principal étant de choisir un environnement adapté à l'importation, à la gestion et au nettoyage des données **JSON**, tout en étant facile à exploiter par notre application web. Après une analyse rapide des différentes options, nous avons opté pour Python. Ce langage a été choisi pour plusieurs raisons :

- Sa simplicité d'utilisation.
- Sa large documentation.
- Les bibliothèques Python très efficaces pour la manipulation et la conversion des données en format JSON.
- D'assurer une compatibilité facile avec l'API que nous devions développer par la suite.

4.2 Répartition des tâches :

Dès le début, nous avons mis en place une répartition claire des tâches afin de garantir une avancée parallèle et efficace du projet. Chaque membre de l'équipe a été assigné à une tâche spécifique en fonction de ses compétences et de son temps disponible

Pour organiser cette répartition de manière claire, nous avons créé un fichier Excel de répartition des tâches. Ce fichier contient une liste détaillée des tâches attribuées à chaque membre avec des dates de début et de fin, permettant ainsi de suivre l'avancement de chaque tâche et de nous assurer que les objectifs étaient bien respectés. Ce fichier a également facilité la communication et l'ajustement des priorités en cas de retard ou de problème dans l'avancement des tâches.

5) Technologies Utilisées :

Pour cette étape du projet, nous avons utilisé plusieurs technologies afin de récupérer et de traiter les données provenant de Vélib :

5.1 Python :

A été choisi pour sa simplicité et son efficacité dans le traitement des données, notamment grâce à ses nombreuses bibliothèques. Une fois les données obtenues, nous les avons filtrées pour ne conserver que les informations essentielles (comme l'ID de la station, son nom, sa localisation, etc).

* La raison principale pour laquelle nous avons choisi Python est sa capacité à créer une API REST grâce à une bibliothèque spécifique, permettant d'afficher les données traitées.

5.2 JSON (Java Script Object Notation) :

Le format JSON est un format de données léger et largement utilisé dans le développement web. Il est plus facile à manipuler dans des environnements modernes, car il est exploitable dans de nombreux langages de programmation.

*Dans notre cas JSON est utiliser pour stocker les données des stations pour qu'elles soient exploitables par notre application web.

5.3 API (Application Programming Interface) :

Une API est une interface qui permet à deux applications de communiquer entre elles. Dans ce cas, l'API de Vélib expose les données des stations de vélos Vélib (comme la disponibilité des vélos et des bornes) sous forme de données JSON ou XML.

*Une API fournit des "points de contact" à travers lesquels les applications peuvent récupérer des données ou envoyer des informations.

5.4 Flask (Framework web Python) :

Flask est un micro-Framework Python utilisé pour créer des API REST. Nous avons utilisé Flask pour développer une API locale (fonctionnant sur le port 5000) qui expose les données que nous avions traité. Grâce à Flask, nous avons pu créer une route "/stations" permettant d'exposer les données des stations renvoyer par notre scripte python (format JSON), et aussi la possibilité d'effectuer des requêtes afin de récupérer ces informations. Flask a été un choix optimal pour créer rapidement une API simple et flexible pour exposer les données traitées.

*Une API REST (Representational State Transfer) est une forme d'interface de programmation d'application qui facilite la communication entre divers systèmes en ligne grâce aux protocoles HTTP.

6) Processus de récupération et de traitement des données :

Dans cette section, nous allons détailler le processus de récupération et de traitement des données provenant de la plateforme Open Data de Vélib :

6.1 Nettoyage des Données et Conversion en JSON :

Nous avons récupéré avec un scripte de python les données des stations Vélib via une API publique. Le processus de nettoyage a consisté à vérifier la validité des données reçues et à ne conserver que celles qui étaient complètes. Les informations importantes telles que l'ID de la station, le nom, la localisation (latitude et longitude) ont été validées pour s'assurer de leur présence et de leur bon type (par exemple, la latitude et la longitude doivent être des nombres flottants).

*Une fois les données validées et filtrées, elles ont été structurées sous forme de JSON.

6.2 Développement de l'API :

Le développement de l'API a été une étape clé de notre projet, car il permet à notre application d'accéder aux données traiter par notre scripte python en temps réel. Nous avons choisi Flask comme Framework pour créer cette API en raison de sa légèreté et de sa simplicité d'utilisation, ce qui le rend idéal pour des projets comme le nôtre où l'objectif est d'exposer des données rapidement et efficacement.

L'API que nous avons développée est composée de plusieurs route (point d'accès). Chaque route est associée à une fonction qui retourne des données au format JSON, permettant ainsi d'accéder à différentes informations traitées dans notre script.

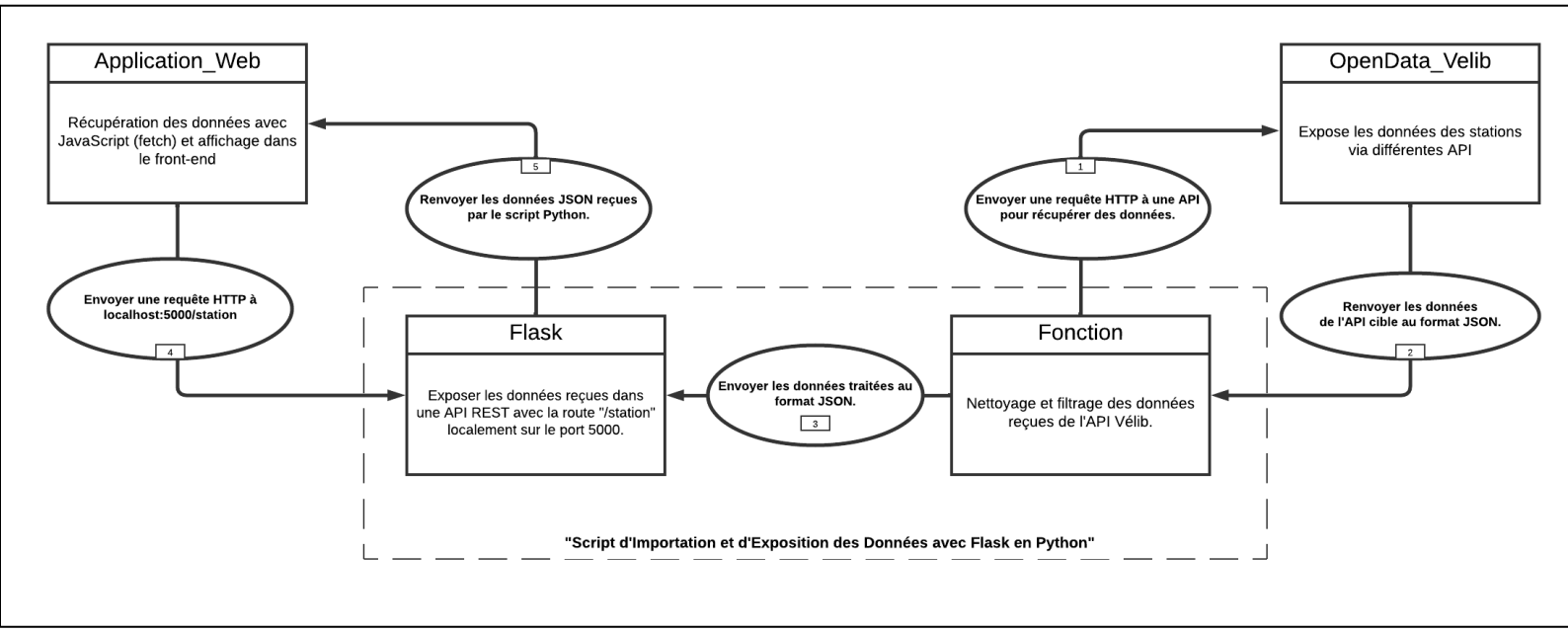
La premier route « /stations », renvoie la liste complète des stations avec leurs informations de base comme l'identifiant, le nom, la latitude et la longitude.

La deuxième route, **«** /station/{id} **»**, est associée à une fonction qui permet d'obtenir des informations détaillées pour une station spécifique (par exemple, le nombre de vélos) en utilisant son identifiant.

Pour route « /station/{id} » et d'autre route développer pur ce projet on ne va pas les détailler sur ce premier livrable

L'architecture de l'API est simple et repose sur des requêtes HTTP de type GET pour récupérer les informations. Lorsqu'une requête est effectuée sur l'un des route, l'API charge les données créer par notre scripte et les renvoie sous forme d'une réponse JSON, facilement consommable par l'applications.

Diagramme UML qui illustre le processus :



7) Difficultés Rencontrées et Solutions :

7.1 Difficulté technique :

1. Problèmes de récupération des données en JavaScript (CORS) :

Nous avons d'abord tenté de récupérer les données directement en JavaScript, mais nous avons été confrontés à des problèmes de CORS (Cross-Origin Resource Sharing), qui ont empêché le téléchargement des données depuis la plateforme Open Data de Vélib. Cette restriction a empêché l'utilisation de JavaScript pour interagir directement avec l'API depuis le navigateur.

- **Solution :**

Nous avons alors opté pour Python, qui a permis de contourner le problème de CROS en récupérant les données via des requêtes HTTP avec la bibliothèque requests.

2. **Récupération des données par le Frontend générées par le script Python :**

Nous nous sommes interrogés sur la manière dont le frontend de notre application web pouvait récupérer les informations produites par notre script Python.

- **Solution :**

Par suite de recherches, nous avons découvert la présence de la bibliothèque Flask qui facilite la création d'une API REST. Cette API permet d'afficher les données générées par notre script Python, simplifiant de ce fait la communication entre le backend et le frontend.

7.2 Difficulté d'organisation :

1. **Difficultés logistiques et organisationnelles liées à la gestion de l'équipe:**

Une autre difficulté a été liée à l'organisation de notre travail en équipe. En effet, notre groupe est composé de membres issus de différentes classes et nos emplois du temps étaient chargés. Nous avons souvent rencontré des difficultés pour nous réunir en personne en dehors de nos heures de cours. Nous avons donc dû organiser nos réunions principalement pendant les pauses déjeuner ce qui limitait le temps disponible pour avancer sur le projet.

- **Solution :**

Pour pallier ce manque de créneaux disponibles, nous avons opté pour des réunions en visioconférence, ce qui nous a permis de discuter et de progresser, même à distance. Nous avons également veillé à ce que toutes les tâches soient clairement définies dans un fichier Excel de répartition des tâches, afin de garantir un suivi régulier du projet.

2. **Difficulté de collaboration en équipe, manque de connaissance mutuelle :**

Une autre difficulté majeure a été le manque de connaissance entre les membres de l'équipe. Comme nous ne nous connaissions pas avant le début du projet, il était difficile de savoir qui avait quelles compétences ou comment chacun allait contribuer efficacement. Ce manque de communication initiale a

parfois rendu les choses compliquées, car chacun de nous était un peu timide et ne savait pas exactement comment les autres allaient aborder leurs responsabilités.

- **Solution :**

Pour surmonter cette difficulté, nous avons commencé par une réunion de présentation où chaque membre a expliqué ses compétences et ses attentes vis-à-vis du projet. Nous avons aussi mis en place une répartition claire des tâches, de manière que chacun ait une responsabilité précise et bien définie. Cela a permis d'établir une meilleure communication et de créer un environnement de travail plus confortable pour tous.

3. **Incertitude sur les niveaux de compétences des membres :**

Une difficulté supplémentaire a été liée à l'incertitude concernant les niveaux de compétences des membres de l'équipe. Personne ne savait exactement le niveau technique des autres, ce qui a causé un certain stress, notamment lorsqu'il a fallu choisir les technologies et répartir les tâches.

- **Solution :**

Afin de mieux gérer cette situation, nous avons échangé sur nos expériences respectives et avons convenu d'utiliser des technologies connues de tous, comme Python, pour éviter de se retrouver bloqués sur des aspects techniques que certains n'auraient pas maîtrisés. Nous avons également ajusté la répartition des tâches pour que chacun puisse travailler sur des aspects du projet correspondant à ses compétences.

8) Conclusion et Perspectives :

Cette première partie du projet a permis à notre équipe de relever plusieurs défis techniques tout en acquérant une expérience précieuse en gestion de données, développement web et travail en équipe. Nous avons réussi à importer les données de Vélib depuis la plateforme Open Data, à les nettoyer et à les convertir en JSON, ce qui a facilité leur manipulation et leur intégration dans une application via une API Flask. Cette API permet désormais à notre application de récupérer en temps réel les données des stations Vélib.

Malgré ces réussites, nous avons dû faire face à des problèmes de communication et d'organisation, notamment en raison de nos emplois du temps et de notre manque de connaissance préalable les uns des autres, ce qui a ralenti le travail initial.

Cependant, ces obstacles ont permis d'améliorer notre gestion de projet et notre capacité à travailler ensemble. Grâce à des outils collaboratifs comme Google Meet et discord, nous avons surmonté les difficultés logistiques et facilité la coordination des tâches. La répartition claire des responsabilités et l'utilisation d'un fichier Excel pour le suivi des tâches ont aussi contribué à maintenir un bon rythme de travail et à garantir que chaque membre de l'équipe savait ce qu'il devait faire.

Perspectives :

Bien que ce premier rendu soit une étape importante, il reste plusieurs améliorations à apporter pour optimiser le projet à long terme. Tout d'abord, l'API pourrait être enrichie pour inclure davantage de fonctionnalités. Par exemple, la gestion des erreurs serait une amélioration importante, permettant à l'API de mieux répondre aux situations imprévues, comme un échec de récupération des données ou des demandes mal formulées.

Enfin, nous envisageons d'optimiser l'architecture du projet pour qu'il soit encore plus modulaire et facile à maintenir, notamment en ajoutant de nouvelles fonctionnalités ou en traitant de nouvelles sources de données.

En résumé, bien que cette première partie projet ait été un succès, il offre de nombreuses possibilités d'améliorations qui pourraient rendre l'application plus robuste et mieux adaptée aux besoins des utilisateurs finaux.