**ReadMe :**

# Equipe :

Nabil HATRI , Karim Marzouk , Rabii Ben Ouirane , Redha REMILI

# Idée derrière le projet :

Notre projet consiste à récupérer les données de la météo en temps réel ainsi que les données des Velib ( sur paris ) également en temps réel, afin de créer une interface web sous python ( Streamlit ) qui permet à un utilisateur de faire toutes les opérations **CRUD** pour la météo et les Velib, en plus de cela nous avons ajouté une page de data visualisation qui permettra à l’utilisateur de visualiser ses différentes données disponibles afin de sortir avec des conclusions intéressantes ( corrélations entre différents champs…)

# Stack technique :

Nous avons réalisé l’intégralité du projet en **Python :**

* **Streamlit** : Streamlit est un Framework open source pour créer des applications web interactives en Python. Il permet de créer des applications de données rapidement et facilement en utilisant des composants prédéfinis comme des graphiques, des tableaux de données et des formulaires. Il est conçu pour être simple à utiliser et facile à apprendre, même pour les développeurs qui n'ont pas d'expérience en développement web. Il est également extensible et peut être utilisé avec d'autres bibliothèques Python populaires pour les visualisations de données, l'apprentissage automatique et la manipulation de données.
* **Pandas** : Pandas est une bibliothèque Python qui nous a servi à bien formater, structurer, nettoyer et transformer nos données récupérer dans des formats différents ainsi que des sources multiples.
* **Pymongo** : Pymongo est une bibliothèque Python qui permet de facilement intéragir avec les bases de données MongoDB. Elle offre des outils pour insérer, récupérer, mettre à jour et supprimer des documents, ainsi que pour gérer les index et les utilisateurs.
* **dateTime** : datetime est un module Python qui fournit des classes pour manipuler les dates et les heures. Il permet de créer des objets datetime pour stocker des dates et des heures, ainsi que des objets timedelta pour stocker des durées, nous l’avons utiliser pour stocker la date et l’heure exacte quand un utilisateur rempli le formulaire de Create ou Update.
* **Pytz** : Pytz est une bibliothèque Python qui fournit des informations de fuseau horaire pour les zones horaires à travers le monde. Il permet de créer des objets datetime conscients du temps à partir de dates et d'heures locales et de les convertir en heures UTC 🡺 nous l’avons utilisé pour stocker automatiquement les dates et les heures au bon format selon le fuseau horaire de la ville sélectionnée.
* **Requests** : Requests est une bibliothèque Python pour envoyer des requêtes HTTP. Elle permet de faire des requêtes GET, POST, PUT, DELETE et d'autres types de requêtes couramment utilisées, ainsi que de gérer les redirections, les cookies et les authentifications 🡺 nous l’avons utilisé pour nous connecter aux différentes API, et récupérer les données dont nous avons eu besoin.
* **Json** : Bibliothèque python xxxxx 🡺 Nous l’avons utilisée pour formater les données retournées par nos API en format JSON, pour les stockées dans Mongo ainsi que pour avoir nos données historiques dans un seul fichier pour faciliter la transmission des données entre nous.

# Données :

1. **Vélib - Vélos et bornes - Disponibilité temps réel :** Les données mises à disposition sont des données de type dynamique permettant de suivre l’évolution du service en temps réel. Le moment de la dernière mise à jour est renseigné dans chaque base.

Ces données nous permette de connaître en temps réel le nombre de vélos mécaniques/électriques à chaque station ainsi que le nombre de bornes libres.

1. **WeatherStack :** WeatherStack est une API de météo en temps réel qui nous a permis de récupérer des informations météorologiques pour des emplacements spécifiques à travers le monde dans notre cas nous nous sommes focalisés sur la ville de Paris. Elle nous a fourni des données en temps réel telles que la température actuelle, les prévisions météorologiques, les conditions météorologiques, les vents, les nuages, la pression atmosphérique, l'humidité, etc. Il prend en charge des requêtes par adresse, par nom de ville, par code postal et par coordonnées GPS. Elle permet également de récupérer des données historiques de la météo pour des dates spécifiques mais cette fonctionnalité est payante d’où la création de notre programme qui simule l’historique. Elle est accessible via une clé qui doit être utilisée pour toutes les requêtes, et il existe des limites d'utilisation pour les différents niveaux de prix, dans notre cas nous avons utilisé l’offre basique et gratuite qui a une limite de 250 appels **max**.

# Fichiers :

1. Météo\_api\_historique.ipynb : Fichier dans lequel nous nous connectons à l’API WeatherStack pour récupérer les données relatives à la météo pour différentes villes en temps réel, nous utilisons aussi ce programme pour stocker nos données historiques.
2. Météo.py : Fichier dans lequel nous instancions notre interface web Streamlit, et que nous simulons toutes les opération **CRUD**
3. Vélib.py : Fichier dans lequel nous nous connectons à l’API Velib pour récupérer les données relatives à la station pour différentes villes en temps réel.
4. Visualisation.py : Fichier dans lequel nous affichons différents graphiques à savoir :
   1. **Répartition des températures par ville**: Ce graphique montrera la variation de température par ville. On récupère les données de notre base de données, les trie par nom de ville et affiche les variations de température dans un graphique de ligne avec le nom de la ville sur l'axe des X et la température sur l'axe des Y.
   2. **Nombre de vélos disponibles par borne** : Ce graphique montrera le nombre des vélos disponible sur chaque borne. Chaque bar représente le nombre des vélos par bornes.
   3. **Top 10 des stations qui contiennent le plus de vélos disponibles + Comparaison entre Vélos électriques et mécaniques**
   4. **Répartition des stations de vélib par adresses : graphique en carte**

# Problématiques rencontrées :

1. **Historiques manquants :** Ceci est le premier souci que nous avons rencontré, en effet, les API que nous avons trouvée ne fournissait que les données en temps réel :
   1. **Exemple :** API Météo : à chaque appel de cette API nous récupérons seulement les données de la météo relatives à la date et l’heure de l’appel de l’API
   2. **Solution :** Nous avons créé un programme qui permet de stocker nos données en live de manière différée dans le temps ce qui nous a permis d’avoir un historique de la météo pour différentes villes 🡺 pour la ville de Paris par exemple nous avons la possibilité de retournée la température de différentes dates antérieure à la date actuelle.
2. **Qualité de la donnée :** Nous avons vite rencontré des problématiques concernant le formatagedes données par exemple pour les données de la météo, la date était stockée en chaine de caractère, ce qui rendait nos aggregate compliqué à coder
   1. **Solution :** utilisation de la bibliothèque **Datetime** et **Pytz** reformater la donnée et pouvoir l’utiliser dans nos programmes.

# Conclusion :

Nous avons affichés nos conclusion dans la partie dataViz de Streamlit, malheureusement, nous n’avons pas assez de données pour donner des conclusions concrètes.