Rapport

LIFPROJET

License informatique 2021/2022

Groupe : CGH 03/05/22

BOUDRAA Amine
BOUDRAA Abdelhakim
TALAHARI Mohamed Chihab
BOUSSAHLA Ghiles

Encadrant:

Rémy Cazabet

INTRODUCTION :

Titre

CHGA COVID_19 DATA EXPLORER

Objectif :

- Développement d'une application type DATA EXPLORER avec une interface WEB interactive.

Plan :

Notre objectif de base était de partir sur la conception d'une application nous permettant de consulter/manipuler les informations relatives au virus COVID-19 qui touche actuellement notre monde. En plus d'être un bon candidat grâce à la quantité de donnée disponible, cela nous permettait de parler d'un sujet actuel.

Nous voulions développer les fonctionnalités d'un explorateur de données classiques mais pas seulement. En effet nous souhaitions posséder différentes interactions qui puissent satisfaire le degré de curiosité de chacun. Il y'as des personnes qui ne sont pas très intéressés par les statistiques mais qui préfèrent seulement avoir les données simples tout en conservant un regard global. D'autres au contraire préfèrent avoir une approche plus scientifique et consulter un plus gros volume de données. Nous avons essayé de modéliser notre application en proposant un contenu a différents utilisateurs.

Ce rapport traduit nos choix de conception, de sélection d'outils, et de développement de notre application en accord avec tous les membres du groupe et notre encadrant.

Il est important de préciser avant la lecture de ce rapport ou l'utilisation de notre application, que toute manipulation ou interprétation de ces données ne sont et ne seront pas sous notre responsabilité.

Toute nos données sont sourcées et ne reflètent en aucun cas les préférences ou les croyances des membres du groupe ni de leur encadrant.

Organisation :

BOUDRAA Amine : développement Front end.
BOUDRAA Abdelhakim : développement Back end.
TALAHARI Mohamed Chiheb ET BOUSSAHLA Ghiles : traitements et analyse des Données.

BACK END et création d'API

1)Plan et outils

Concernant la partie backend du projet, il a été décidé de partir sur des outils spécifiques à la réalisation du projet.

La première chose à faire fut de choisir un gestionnaire de base de données, l'objectif est en premier lieu de pouvoir transférer une série de données du back end vers le front end qu'elle soit en brute où traitée à travers python. Cet outil devait nous permettre de pouvoir modifier nos données sans avoir besoin de les recharger à chaque fois, nous avons donc décidé de partir sur un gestionnaire de base de données dit no SQL.

Nous avons donc décidé de choisir MongoDB car c'est un outil simple et très flexible mais qui possède comme inconvénient d'avoir une taille de données plus élevée en contrepartie. Il nous a permis de pouvoir charger des données avec un format d'origine JSON ou en CSV dans la base.

La 2eme partie consistait à trouver un langage de programmation pour coder une API permettant de communiquer très simplement avec la base de données en premier lieu mais aussi les fichiers PYTHON ainsi que le front end.

Le choix s'est donc porté sur NODE.JS qui permet de coder une API en Java script très adapté aux bases de données NOSQL, il est rapide et excellent réel. Le gestionnaire de paquets NPM nous nous a permis d'utiliser des Frameworks pour nous faciliter la tâche, notamment express qui permettait de créer des habits en javascript de manière simple MONGOOSE ce qui permettait de faire communiquer la base de données avec l'API.

Après cela nous avions donc tous les éléments nécessaires à la réalisation d'une API stable.

2) Conception et difficulté

Le plus dur dans cette partie du projet fut la gestion et la transmission des données des fichiers PYTHON vers le front end, car après traitements ce sont des schémas de prédictions de données et non pas du texte que nous devions transférer.

Après plusieurs jours de recherche et d'essai, nous avons codé une méthode permettant de le faire à l'aide notamment à « python-Shell » ou la méthode « spawn ».

Malgré cela il a été décidé de se passer de ces deux méthodes pour se concentrer sur les fonctionnalités qu'offraient la librairie PLOTLY de python.

Pour ce qui est de l'affichage brut des données sur l'interface, nous avons respectés le plan de base qui était de faire passer les données par l'api NODE.JS pour être ensuite récupéré par l'interface.

Deux jours avant la présentation, un problème est survenu avec le serveur et la base de données MongoDB hébergé en local sur celui-ci, ce qui nous poussa à migrer nos données sur l'outil CLOUD de MongoDB nommé ATLAS (plutôt que de résoudre le problème, en raison du manque de temps et par sécurité).

Toutes ces étapes ont été faites en collaboration avec toutes l'équipe.

Traitements et analyse des données

1) Outils et langages utilisés pour le traitements et l'analyse de données :

Etant donné notre objectif qui consiste à faire communiquer, lancer et exécuter nos résultats d'analyses des données covid avec le backend (grâce à Node.js) et d'afficher leurs résultats sur le frontend, notre choix s'est donc porté directement vers python et ce depuis le début du projet, car c'est un langage de haut niveau et surtout simple à utiliser puisqu'il est un langage interprété qui exécute ligne par ligne qui simplifie donc la correction des erreurs et bugs, mais aussi car il permet d'envoyer les résultats au frontend sous différentes méthodes, sa richesse en librairie était aussi l'un des avantages qui nous a orienté à choisir **Python**.

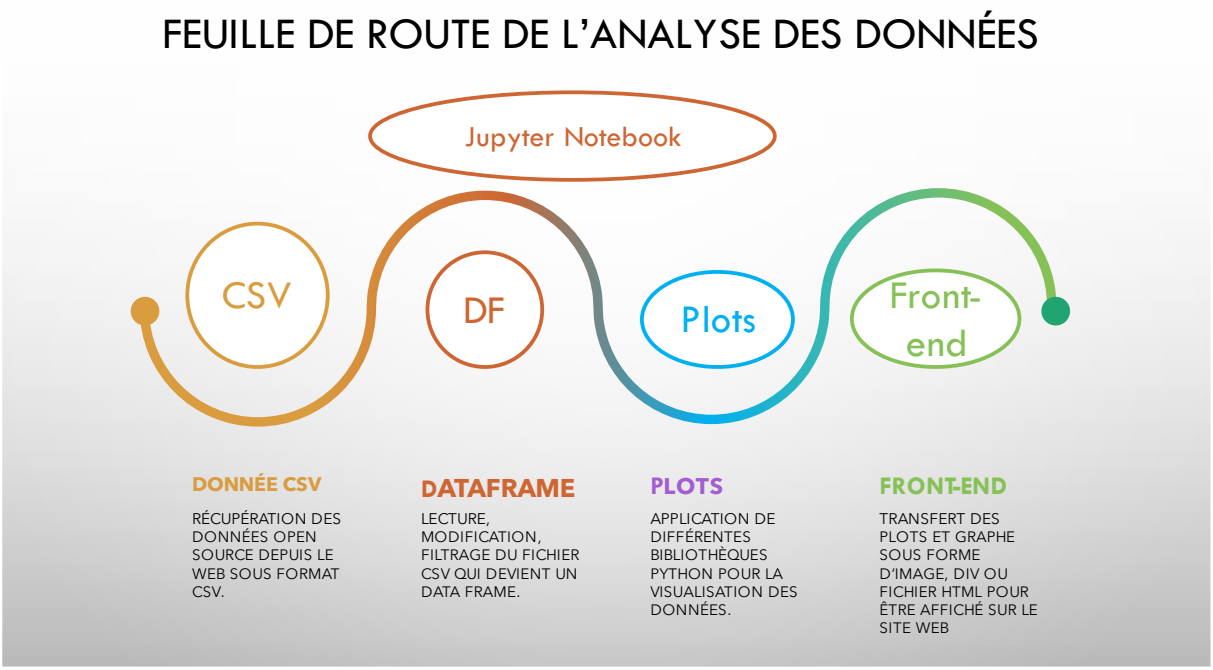
La prochaine étape après le choix du langage était la recherche des données Covid qui couvrent tous les pays du monde, où on a basé notre choix sur la recherche de données open source sous format JSON ou CSV à partir de différents sites web tel que : « Our World In Data » Wordlometer...

Après la récupération des données, venait l'étape de leur nettoyage, révision et épuration, qui consiste aux changements des index selon le besoin, changement des noms des colonnes pour être plus reconnaissable et identifiable, suppression ou ajout de colonnes selon le besoin, changement du type d'une colonne pour appliquer des fonctionnalités tel que la récupération de la valeur « max() » d'une colonne etc…

Pour réaliser les tâches précédentes, on a dû se former à l'utilisation de différentes librairies telles que Python Pandas et Numpy. Pour l'utilisation de ces bibliothèques, on a choisi l'outil Jupyter Notebook car c'est un excellent outil de développement qui permet de mettre en valeur notre travail. On peut voir le code et les résultats d'exécution en même temps, puisqu'il offre la possibilité d'exécuter cellule par cellule pour mieux comprendre ce que fait le code et ceci facilite la détection d'erreurs.

Parmi les bibliothèques Python utilisées : *Pandas* : La bibliothèque open-source spécifiquement conçue pour la manipulation et l'analyse de données en langage Python, qui nous a permis la lecture de nos données depuis des fichiers csv, leur nettoyage, la jointure avec d'autres « dataframe » tel dans la création de la carte interactive avec folium. *Seaborn* Permet de créer des graphiques statistiques en Python, elle est basée sur Matplotlib et s'intègre avec les structures Pandas. *Matplotlib* est destinée à tracer et visualiser des données sous forme de graphiques en 2D. *Scikit-learn* qui est largement utilisée par diverses organisations pour prédire et aussi la création de cluster suivant plusieurs paramètres tels que le clustering en « k-means » qui prend la moyenne, l'agglomerative clustering en suivant la distance euclidienne, distance de Manhattan etc... *Plotly* qui prend en charge divers graphiques tels que le nuage de points, facilite aussi l'exportation des graphes sous format image, div ou HTML. *Folium* est une puissante bibliothèque Python qui offre la possibilité de créer plusieurs types de cartes Leaflet interactives. Vous pouvez également créer des cartes Jupyter en ligne dans Folium. *Prophet* Outil développé

par Facebook qui permet d'effectuer des prédictions suivant 4 éléments : la tendance, la saisonnalité et l'effet événements (par exemple : concert, match de football, tout regroupement de plusieurs personnes dans un endroit ouvert ou fermé) et l'effet de bruit (appelé aussi le terme d'erreur).



2) Difficultés rencontrées :

A la fois un avantage et un inconvénients la richesse de python en bibliothèques, le temps de formation qui était nécessaire pour maitriser ces dernières était important.

L'utilisation de MongoDB pour récupérer nos données et les convertir en CSV nous a rendu certaines fonctionnalités inutilisables tel que la récupération de la date la plus récente pour un certain pays, filtrer uniquement les lignes où la France figure, en utilisant « to_csv() » on a donc utilisé le même fichier en version csv pour l'analyse des données.

L'exécution des scripts python finaux depuis le backend nous a posé problèmes, ces derniers n'affichant pas les graphes et les plots réalisés, nous a forcé à se tourner vers une autre méthode qui est de sauvegarder les graphes directement sur Jupyter Notebook, cette méthode même nous a confrontés à des problèmes parmi lesquels : l'impossibilités d'appeler certaines fonctions sur des types de graphes différents (exemple de la carte interactive de type folium map à qui « savefig() » ne peut s'appliquer, ainsi que tous graphes interactifs), la solution était de l'enregistrer en fichier html grâce à « save('nom du fichier.html') » pour d'autres graphe en utilisant plotly via l'appel à « plotly.offline.plot(output_type='div') »).

Développement du site web

1)Choix de conception

La conception du site Web s'est fondée sur trois problématiques : La récupération des données, l'affichage de ces dernières et les interactions possibles.

L'on devait s'organiser de tel sorte à pouvoir utiliser les données transmises par notre serveur et nos scripts python tout en permettant certaines interactions qui puissent satisfaire le degré de curiosité de différents types d'utilisateurs.

2)Première interface

La solution d'une carte interactive pour des utilisateurs qui ne sont pas forcément très intéressés par la consultation simultanée de multiples statistiques était une solution évidente. Donc comment représenter dans une carte les statistiques les plus importantes ? On s'est basé sur un système que l'on voulait « tape a l'œil « pour attirer l'attention de l'utilisateur : une carte customisable affichant différents modèles géométriques qui pourrait servir d'indicateur.

En effet, coder une carte Leaflet ayant comme fonctionnalité de pouvoir choisir quel '' mode d'affichage '' et surtout sur quelle région nous **pouvions** appliquer nos fonctionnalités (interagir avec le curseur, affichage des statistiques intéressantes tout en actualisant le code couleur de chaque région, définir des échelles en fonction de chacune d'entre elles…) n'était pas chose aisée, mais le résultat final nous a semblé satisfaisant.

La plus grande difficulté aura surtout de codé les fonctionnalités en fonction de la manière dont nous récupérions les données (importation d'un JSON, utilisation des champs qui nous intéressent…).

2)deuxième et troisième interface

Une fois l'étape de la carte achevée, l'on devait réfléchir à la manière dont nos utilisateurs qui sont intéressés par des statistiques plus complexes ainsi qu'à leur manipulation peuvent les consulter. Nous avons donc choisi de dédier une partie propre au graphes et diagrammes qui seront générés par des scripts Python (qui seront affichés par des importations d'aimes classiques ou des générations de Div par Plotly (consulter plus haut dans le rapport).

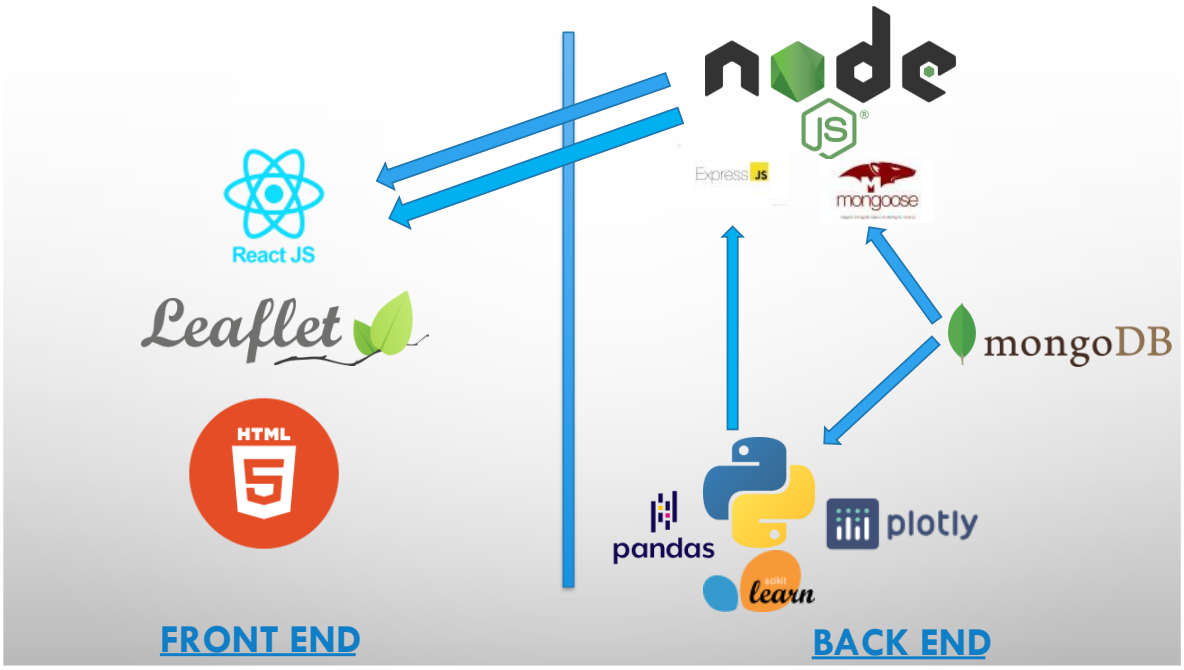
Maintenant, pour la consultation des données BRUTES (avant toute manipulation), nous avions choisis un moniteur de données avec barre de recherche. Le problème était double, car autant le serveur arrivait à diffuser sur un port des données JSON grâce à NODEJS, autant les récupérer était une histoire totalement différente.

L'autre problème était le temps de réponse entre chaque demande/recherche si elles étaient successives, en effet si l'on manipule un grand volume de données, la génération de données puis l'affichages de ces dernières risquaient de prendre un certain temp. Pour remédier à ce problème, l'utilisation de React (bibliothèque JS) nous semblait approprier.

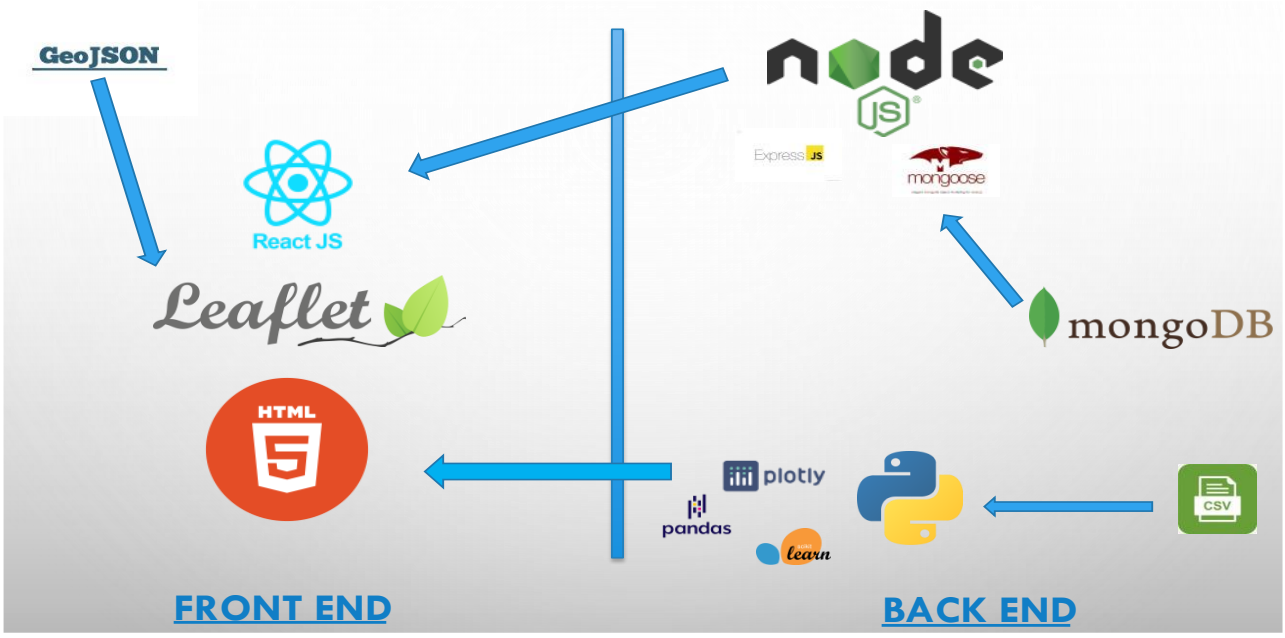
Malgré la difficulté d'apprentissage (React utilises sa propre façon de traiter/manipuler les données, elle est unique), nous avons finalement réussis à concevoir un explorateur qui permet de générer une seule fois les données, et de les manipuler par la suite avec un temp d'attente quasi nul.
Le rendu final du site WEB et la finition seront intégralement faits avec du BOOTSTRAP CSS.

Bilan :

Voici en premier lieu un schéma représentant comment l'application devait fonctionner avec les différents outils utilisés (les flèches représentent le sens dans lesquelles les données sont acheminés) :



Mais au fur et à mesure que nous avancions et selon les difficultés, le schéma a évolué à plusieurs reprises pour finalement arriver à ce résultat final :



Ce qui a marché :

Dans la globalité malgré plusieurs changements de plan en cours de routes, nous avons atteints tous nos objectifs (et plus encore) et avons appris à utiliser de nouveaux outils et langages.

Nous étions bien encadrés, ce qui nous a permis de relativement bien avancer sur plusieurs points du projet.

Ce qui a moins marché :

Comme absolument tout ce que l'on a fait au cours du projet était nouveaux pour nous, on a dû apprendre à faire plusieurs choses, On a donc atteint nos Objectifs mais pas de la meilleure des manières possibles ni la plus optimisés, accentuées par le fait que nous étions en autoformation.

Le schéma de base a été conçus sans savoir réellement si nous étions capables de faire tout ce que l'on n'avait prévu ni si les outils utilisés pouvaient le permettre, ce qui a contribué aussi d'une certaine manière à peut-être fausser nos jugements et a modifier le plan en cours de routes.