Rapport de Projet
« Groupe B »

SUROTH Saied

KACI Yannis

ANZID Ayoub

MARIE CHRISTY Honorine

CARUSO Enzo

BAILLARGEAU Kelyan

GERARD Pierre
HEBERT David

IUT de Villetaneuse – Université Paris 13

Sommaire

Introduction

Description du projet

Analyse des fonctionnalités à mettre en place

Solutions mises en œuvre

Problèmes rencontrés et leurs résolutions

Algorithmes

Tests

Reste à faire et future de Gari

Conclusion

Glossaire

Références bibliographiques

Annexes

Introduction

Le fondateur AMMOUCHE Abdeldjalil de l'entreprise Gari nous a proposé un projet pour faire une visualisation de sa base de données en forme de map.
L'équipe de projet est constituée de 6 membres qui sont Saied, Yannis, Ayoub, Honorine, Enzo et Kelyan.

En quelques lignes, Gari c'est : une application communautaire d'échange de places de stationnement entre particuliers. Un automobiliste quitte sa place, il la signale sur Gari. Un autre automobiliste cherche une place, il se signale sur Gari.
Gari les met en relation et guide l'automobiliste chercheur de place jusqu'à la place qui va se libérer, car l'automobiliste quittant sa place a l'obligation d'attendre l'arrivée de l'autre automobiliste.
Un système de points permet de régir les demandes et permet de créer un cercle vertueux. Ainsi, pour demander une place il faut disposer de crédit suffisant pour obtenir le crédit suffisant il faut d'abord donner une place.
Lorsque l'on cherche une place, la demande est active durant 5 minutes. Si aucun automobiliste n'a été trouvé, on est notifié et on nous propose de renouveler ou pas notre recherche.
Lorsque l'on donne sa place, on est invité à garder la place 3 minutes. Au-delà, on est libre de partir et on aura gagné les points bonus.

Aujourd'hui ce système fonctionne très bien en Algérie et sera implémenté à Paris. Le but du projet est de faire un data visualisation qui sera une sorte de pub pour Gari.

Notre tâche est de projeter les comportements des automobilistes sur une map, afin de visualiser les départs et les demandes de stationnement en 30 secondes.
Nous n'avons pas de contrainte pour le choix des technologies.

Pour réaliser le projet, nous avons finalement choisi d'utiliser le langage Python. Pour la gestion de projet, nous avons décidé d'utiliser les méthodes agiles (GitHub / Tableau d'avancement / Diagramme de Gantt / Diagramme de PERT) → Voir annexe
Ce projet sera pour nous une expérience enrichissante, car nous allons découvrir plusieurs fonctionnalités du langage python que nous n'avions pas étudié en S1.

Description du projet

Dans ce data visualisation, nous devrons montrer sur une carte de Paris tous les échanges qu'il y aura eu entre un donneur de place et un chercheur de place en faisant apparaître les caractéristiques des voitures.
Le donneur est appelé Garitoo : en donnant une place il gagne des points bonus et un crédit parking. Celui qui cherche une place est appelé Garini : il doit avoir au minimum 1 crédit parking. Si le Garini ne trouve pas de Garitoo, l'intelligence artificielle de Gari prend le relais et en trouve un autre. L'entreprise Gari s'occupe de trouver le Garitoo le plus proche et guide jusqu'à cette place le Garini concerné. → Voir annexe (maquette)

Analyse des fonctionnalités à mettre en place

Dans le premier rendez-vous que nous avons eu avec le fondateur, il nous était dit qu'il n'y aurait pas d'interaction avec la map. Puis le lendemain nous avons reçu le cahier des charges, et nous avons pu observer d'autres besoins établis par le client et qui n'était pas dans notre compte rendu du premier rendez-vous.

Voici les besoins qui prouvent l'interaction avec la map :

- Si l'utilisateur peut cliquer sur un bouton pour afficher seulement les Garini, cela reste une interaction (modification de la map).
- Si l'utilisateur peut cliquer sur un bouton pour afficher seulement les Garitoo, cela reste une interaction (modification de la map).
- Si l'utilisateur peut zoomer sur la map, cela reste une interaction avec la map, même si l'action est minime.

Vu que nous avions carte blanche, nous avons décidé de continuer le projet malgré le manque d'information importante et d'improviser ce qui nous manquerait.

D'après notre compréhension avec le fondateur :

Nous devons donc initialiser une carte de Paris sur laquelle nous devons placer des points verts qui représentent les Garitoo, et des points rouges pour les Garini. Puis nous devons les relier pour représenter un « match » entre deux voitures. Lorsque nous passons sur les points avec la souris, nous devons voir les caractéristiques de la voiture.
Cette carte doit aussi être connectée à une base de données. Nous allons récolter les données, les analyser puis les utiliser pour afficher les informations nécessaires.
Nous devons aussi pouvoir zoomer sur la carte.
Cela sera une sorte d'animation de quelques secondes qui va montrer tous les échanges qu'il y a eu dans une période, par exemple un data visualisation avec les échanges qu'il y a eu la semaine du 14 octobre 2019.

Voici la liste des user-stories que nous avons établi selon les besoins du client :

1) En tant qu'utilisateur, je veux pouvoir visualiser les activités des clients afin d'avoir un aperçu du trafic interne et du fonctionnement de l'application.

2) En tant que client, je veux que la map ai les 4 points cardinaux afin de se repérer sur la map.

3) En tant qu'utilisateur, je veux pouvoir visualiser les véhicules qui sortent et les véhicules qui trouvent une place afin de voir les échanges entre les véhicules.

4) En tant qu'utilisateur, je veux pouvoir cliquer sur un bouton afin d'afficher seulement les personnes qui quittent leur place.

5) En tant qu'utilisateur je veux pouvoir cliquer sur un bouton afin d'afficher seulement les personnes qui trouvent une place.

6) En tant qu'utilisateur, je veux pouvoir zoomer (de sorte à voir la rue) et dézoomer afin d'avoir une visualisation en détails des échanges.

7) En tant qu'utilisateur, je veux pouvoir afficher le modèle de la voiture, son immatriculation et la date afin de mieux comprendre le processus du trafic.

À partir des besoins, nous avons fait un diagramme des cas d'utilisation (voir annexe) et un tableau d'avancement pour se répartir les tâches et mieux contrôler l'avancement du projet.

Solutions mises en œuvre

Pour communiquer entre les membres du groupe du projet, nous avons créé un discord, un groupe WhatsApp, une adresse mail commun, et après le cours sur Git un Github. De plus, pour communiquer avec le client nous avons seulement utilisé les mails.
Nous avons fait des recherches sur plusieurs technologies et librairies comme par exemple :
Les librairies en Python → Plotly / MapBox / Matplotlib / Folium / BaseMap / Pandas
Librairie en JavaScript → D3.js

Nous avons décidé d'utiliser Python car nous avions déjà étudié ce langage en S1. Nous avons également choisi ce langage car en choisissant JavaScript, nous aurions dû nous former. Certes, avoir choisi le JavaScript nous aurait apporté de nouvelles compétences et connaissances, mais avec le retard que l'on a eu à avoir les projets et le retard des réponses de notre client, nous avons décidé d'abandonner ce choix. Pour initialiser une carte, nous avons utilisé la bibliothèque Folium et pour nous connecter à la base de données, nous avons utilisé MySQL. Nous avons utilisé la librairie mysql.connector, et pour la période de temps avec le bouton play, nous avons utilisé les librairies datetime / time et les plugins de la librairie folium.

Problèmes rencontrés et leurs résolutions

Nous aurions eu besoin que le directeur technique nous réponde concernant nos questions techniques. Mais à la place c'était le fondateur qui y répondait sans comprendre réellement nos attentes.
De ce fait, nous avons perdu énormément de temps pour la compréhension du projet, et rajouter à cela la réactivité inexistante du client pour répondre aux mails (plus d'un mois pour une réponse partielle). Nous avons tenté d'établir un rendez-vous pour valider le cahier des charges et la maquette (prototype de la map), mais le directeur technique n'était pas en France. Il nous a proposé un appel sur Viber, mais nous avons refusé car cela ne nous aurait servi à rien.

Ensuite nous avions besoin du format de la base de données, connaître l'utilité et le format de chaque attribut de la base de données.
Puis nous avons demandé un extrait de la base de données pour travailler et faire les tests avec de vraies données.
D'autant plus que nous voulions savoir comment reconnaître un Garini / Garitoo / match / don et une recherche dans la base de données.
La seule réponse que le directeur technique nous a donné est la base de données, sans répondre à nos questions. Après analyse de la base de données, nous avons constaté que les données n'étaient pas correctes.

La base de données était incompréhensible sans explications.
Exemple concret : Nous nous sommes aperçus que pour la table 'TROUVER' que l'adresse de départ et l'adresse d'arrivée est la même. Sachant que la voiture se déplace d'un point A et va vers un point B.

Pour ne pas perdre davantage de temps, nous avons donc créer une base de données exploitable juste pour faire nos tests → Voir annexe (base de données). Nous avions trouvé plusieurs manières d'initialiser des maps, comme par exemple avec MapBox. Le problème de cette librairie c'est que l'on ne pouvait pas faire d'animation, et le plus souvent ce n'étaient que des maps statiques paramétrables qu'une seule fois (donc pas de mise à jour).
Ensuite nous voulions tenter d'initialiser une map avec BaseMap et de faire une animation avec Matplotlib et Numpy, mais le problème c'est que l'animation avec Matplotlib se fait dans une figure. Beaucoup d'inconvénients : pas de map dynamique / pas d'interaction. → Voir annexe
Nous avons finalement choisi d'utiliser la librairie Folium en Python car l'animation était possible en combinant un plugin de la librairie folium : 'plugins.TimestampedGeoJson'.

Algorithmes

Pour l'initialisation de la Map, nous avons récupéré une map de folium générée par folium :

m=folium.Map(location=[…….,……],titles='cartodbpositron', zoom_start=12 ,width=1400, height=800, zoom_control=True).

Nous avons mis des marqueurs pour les points qui représentent les garinis et les garitoos :

folium.Marker(location=data['cordonnee']
[i],popup=caraStr(car[nb]),icon=folium.map.Icon(color='…..',
icon= ' ...')).add_to(garini/garitoo)

Le popup sert à afficher les caractéristiques de la voiture lorsque l'on appuie sur un point.

Pour récupérer les données dans la base de données et pour les rendre exploitable, nous avons écrit plusieurs fonctions.

Et pour finir, nous avons dessiner un trait qui relie le Garini à son Garitoo, donc un trait pour représenter le match :

folium.PolyLine([data['match'][i], data['match'][i+1]],color= ' blue', weight=2.5 , opacity=1).add_to(m)

Nous avons ajouté une image de boussole en bas à droite :
FloatImage(url,bottom=5,left=80).add_to(m) avec un url qui contient une image de boussole.

Pour l'implémentation des boutons pour afficher seulement les Garitoo ou les Garini, ainsi que leur nombre, affin d'alléger l'affichage pour que l'utilisateur puisse voir seulement ce qui l'intéresse : par exemple :

garitoo = plugins.FeatureGroupSubGroup(groupe, 'Garitoo') #un bouton pour afficher les garitoos
m.add_child(garitoo) #ajout a la map

folium.Marker(location=data["coordonnee"][i],popup=
caraStr(cara[nb]),icon=folium.map.Icon(color='red',icon='')).add_to(garini)

Pour afficher la map en plein écran :

plugins.Fullscreen(#un bouton pour mettre en plein écran position='topright', title='Agrandir', title_cancel='Quitter le mode pleine écran', force_separate_button=True).add_to(m)

Tests

Pour les tests, nous avons vérifié si l'on était bien connectés à la base de données avant de généré la page HTML. Nous avons fait un test pour voir si les marqueurs étaient à la bonne place sur la Map.
Nous avons réussi à faire une animation à l'aide du plugin 'plugins.TimestampedGeoJson'.
→ Voir annexe (animation avec folium). Malheureusement, on ne pouvait pas mettre de conditions dans le code, de ce fait le contrôle de l'animation était impossible. Nous avons donc abandonné cette idée. Avec le temps qui nous restait, nous avons décidé d'utiliser une map statique.
→ Voir annexe (map finale)

Reste à faire et future de Gari

Nous avons pensé à quelques éléments que nous pourrions rajouter pour améliorer notre data visualisation, éléments non présents par manque de temps. Par exemple, nous voulions faire une animation : voir sur le data visualisation les Garitoo quitter leurs places et les Garini récupérer les places des Garitoo, voir les déplacements de chacun en somme. Nous avions aussi pensé à intégrer un « timer » pour contrôler le temps, faire défiler plus ou moins vite les points, sur telle ou telle période de temps, mais il se trouve que lorsque nous avons réussi à réaliser l'animation, nous ne contrôlions pas l'affichage, donc nous avons préféré rester sur un affichage statique.

En ce qui concerne le futur de la start-up, nous trouvons que leur idée est innovante et très pratique pour leurs utilisateurs, mais Gari reste méconnu (aucun de nous six ne les connaissait avant le projet). Par exemple, on ne trouve pas leur site en cherchant « Gari » sur Google, ils sont mal positionnés au niveau du référencement. Or, selon nous la réussite de cette application repose sur le nombre d'utilisateurs qui doit être relativement élevé : plus il y aura d'utilisateurs, plus l'application sera efficace. Par conséquent, nous pensons que Gari devrait faire plus de publicité pour se faire connaître et ainsi attirer d'avantage d'utilisateurs.

Conclusion

Pour résumer, ce projet était intéressant pour nous car il nous a permis dans un premier temps de travailler avec un vrai client, et donc de voir la réalité du travail pour nous, étudiants.
Ce projet nous a également permis de découvrir des fonctionnalités de Python que nous ne connaissions pas, de voir que Python est un langage puissant, et de nous rendre compte qu'il n'y a pas de langage de programmation parfait : certains ont des avantages que d'autres n'ont pas, et que cela implique de faire un choix pour être le plus efficace possible. Nous pensons aussi que nous n'avons pas eu des conditions optimales pour pouvoir faire notre projet à bien. Avec de la communication et de l'aide de la part de notre client, nous aurions pu améliorer le visuel et les fonctionnalités, et avec une base solide, nous aurions aussi pu améliorer la map. Cela étant dit, les conditions de travail en entreprise ne sont peut-être pas nécessairement meilleures, nous voyons donc le côté positif et formateur que ces contraintes nous ont imposé. Ce projet nous a aussi amené à voir le travail à fournir, non seulement en termes de programmation pure, mais également en termes de gestion de projet, qui est une part très importante du travail d'un informaticien. Ces connaissances acquises durant le projet nous fournissent un bagage qui nous sera utile pour notre stage et notre futur métier.

Glossaire

Garini : Celui qui cherche une place.

Garitoo : Celui qui donne sa place.

Match : Quand un Garini trouve un Garitoo

Data visualisation : Représentation graphique de données statistiques.

Base de Données : Permet de stocker et de retrouver l'intégralité de données brutes ou d'informations en rapport avec un thème ou une activité.

GANTT : Outil utilisé en ordonnancement et en gestion de projet permettant de visualiser dans le temps les diverses tâches composant un projet.

PERT : Une méthode et des moyens pratiques pour décrire, représenter, analyser et suivre de manière logique les tâches et le réseau des tâches à réaliser.

Tableau d'avancement : Tableau regroupant les tâches à effectuer ou déjà effectuées, avec les dates prévues/réelles et les acteurs de celles-ci ; permet de suivre l'avancement du projet.

Référencement : Moyen d'accroître la visibilité d'un site web dans les résultats naturels d'un moteur de recherche.

User-stories : Phrase simple permettant de décrire avec suffisamment de précision le contenu d'une fonctionnalité à développer.

GitHub : C'est un service web d'hébergement et de gestion de développement de logiciels.

Plotly : Développe des outils d'analyse et de visualisation de données.

MapBox : Développe un ensemble de technologies et d'outils cartographiques. Ces projets reposent en très grande partie sur le logiciel libre et sur les données d'OpenStreetMap.

Matplotlib : C'est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques

Folium : Le module folium permet de créer une carte OpenStreetMap et d'y dessiner des marqueurs, des lignes brisées, des polygones, etc.

Pandas : Pandas fait partie des librairies de base pour la data science en Python. Pandas fournit des structures de données puissantes et simples à utiliser.

Références bibliographiques

Les recherches pour la technologie à utiliser :

-https://observablehq.com/@d3/world-tour
-http://zevross.com/blog/2014/09/30/use-the-amazing-d3-library-to-animate-a-path-on-a-leaflet-map/
-https://plot.ly/python/bubble-maps/
-https://plot.ly/python/
-https://www.sqlalchemy.org/
-https://towardsdatascience.com/combining-python-and-d3-js-to-create-dynamic-visualization-applications-73c87a494396
-https://www.datavis.fr/index.php?page=leaflet-control
-https://disciplines.ac-toulouse.fr/informatique/sites/informatique/files/fichiers/localisation/activite_creer_une_carte_personnalisee_avec_python.pdf
-http://www.xavierdupre.fr/app/ensae_teaching_cs/helpsphinx/notebooks/td1a_cenonce_session_12_carte.html
-https://python-graph-gallery.com/313-bubble-map-with-folium/
-https://www.geeksforgeeks.org/python-plotting-google-map-using-gmplot-package/
-https://towardsdatascience.com/easy-steps-to-plot-geographic-data-on-a-map-python-11217859a2db
-https://python-graph-gallery.com/300-draw-a-connection-line/
-https://www.fullstackpython.com/blog/maps-django-web-applications-projects-mapbox.html
-ttps://towardsdatascience.com/geopandas-101-plot-any-data-with-a-latitude-and-longitude-on-a-map-98e01944b972

Initialisation de la map :

-https://github.com/IvanSanchez/Leaflet.Polyline.SnakeAnim
-https://fabdev.fr/articles/iss-python/
-https://towardsdatascience.com/visualizing-air-pollution-with-folium-maps-4ce1a1880677
-https://deparkes.co.uk/2019/02/27/folium-lines-and-markers/
-https://programtalk.com/python-examples/folium.plugins.TimestampedGeoJson/
-https://nbviewer.jupyter.org/github/python-visualization/folium/blob/master/examples/Plugins.ipynb

Animation :

-https://nbviewer.jupyter.org/github/python-visualization/folium/blob/master/examples/Plugins.ipynb#Timestamped-GeoJSON

Connexion à la base de données :

-https://www.youtube.com/watch?v=x7SwgcpACng

Annexes

Notre GitHub du projet : https://github.com/SaiedS/Gari

Tableau d'avancement (Cliquez pour zoomer ctrl + clic) :

Code	Tâche	Début	Fin	Effort 155	Début	Fin	Effort 170	
A	Définition des besoins	01/10	30/10	6	01/10	31/10	8	Saied, Honorine, Yannis, Kelyan, Enzo, Ayoub
B	Mise en place des outils de communication (Gmail,Discord,Whatsapp,GitHub)	01/10	01/10	2	01/10	01/10	2	Saied, Honorine, Yannis, Kelyan, Enzo, Ayoub
C	Mise en place du tableau d'avancement	31/10	03/01	4	12/11	08/01	5	Saied, Enzo
D	Mise en place d'une maquette de la map	31/10	31/10	3	11/11	11/11	2	Honorine, Yannis
E	Etablir un diagrame de cas d'utilisation	31/10	31/10	1	11/11	11/11	1	Saied, Kelyan, Enzo, Ayoub
F	Recherche des technologies possibles	31/10	09/12	25	11/11	20/12	30	Saied, Honorine, Yannis, Kelyan, Enzo, Ayoub
G	Tests avec Plotly	10/12	15/12	6	15/11	20/11	6	Saied, Honorine, Yannis, Ayoub
H	Tests avec Mapbox	10/12	15/12	6	23/11	25/11	7	Saied, Honorine, Enzo, Ayoub
I	Tests avec Matplotlib	10/12	21/12	10	09/12	20/12	10	Saied, Honorine, Yannis,Ayoub
J	Tests avec Folium	10/12	20/12	10	10/12	20/12	12	Saied, Honorine, Yannis, Kelyan, Enzo, Ayoub
K	Tests avec Panda	10/12	15/12	7	15/11	20/11	10	Saied, Kelyan, Ayoub
L	Implémentation base de données test	01/11	07/12	10	01/12	07/12	12	Saied, Honorine, Yannis, Kelyan, Enzo, Ayoub
M	Initialisation de la Map + Animatation Test	21/12	24/12	10	19/12	22/12	12	Saied, Honorine, Yannis, Kelyan, Enzo, Ayoub
N	Implémentation code 4 points cardinaux Map	25/12	25/12	2	23/12	23/12	2	Honorine, Ayoub
O	Implémentation code bouton Match	25/12	27/12	2	28/12	28/12	1	Saied, Kelyan, Enzo
P	Implémentation code bouton Garito	25/12	27/12	2	28/12	28/12	1	Saied, Kelyan, Enzo
Q	Implémentation code bouton Garini	25/12	27/12	2	28/12	28/12	1	Saied, Kelyan, Enzo
R	Implémentation code caractéristique voiture	08/12	05/01	6	08/12	05/01	9	Honorine, Yannis, Ayoub
S	Tests méthodes implémentées	22/12	10/01	15	17/12	09/01	10	Saied, Honorine, Yannis, Kelyan, Enzo, Ayoub
T	Rédaction de la synthèse finale	28/12	10/01	10	28/12	10/01	10	Saied, Honorine, Yannis, Kelyan, Enzo, Ayoub
U	Mise en place des logiciels et librairies	15/11	15/12	12	16/11	14/12	15	Saied, Honorine, Yannis, Kelyan, Enzo, Ayoub
V	Création du diagramme de GANTT	31/10	03/01	2	15/11	08/01	2	Enzo, Ayoub, Yannis
W	Création du diagramme de PERT	31/10	03/01	2	15/11	08/01	2	Enzo, Ayoub, Yannis

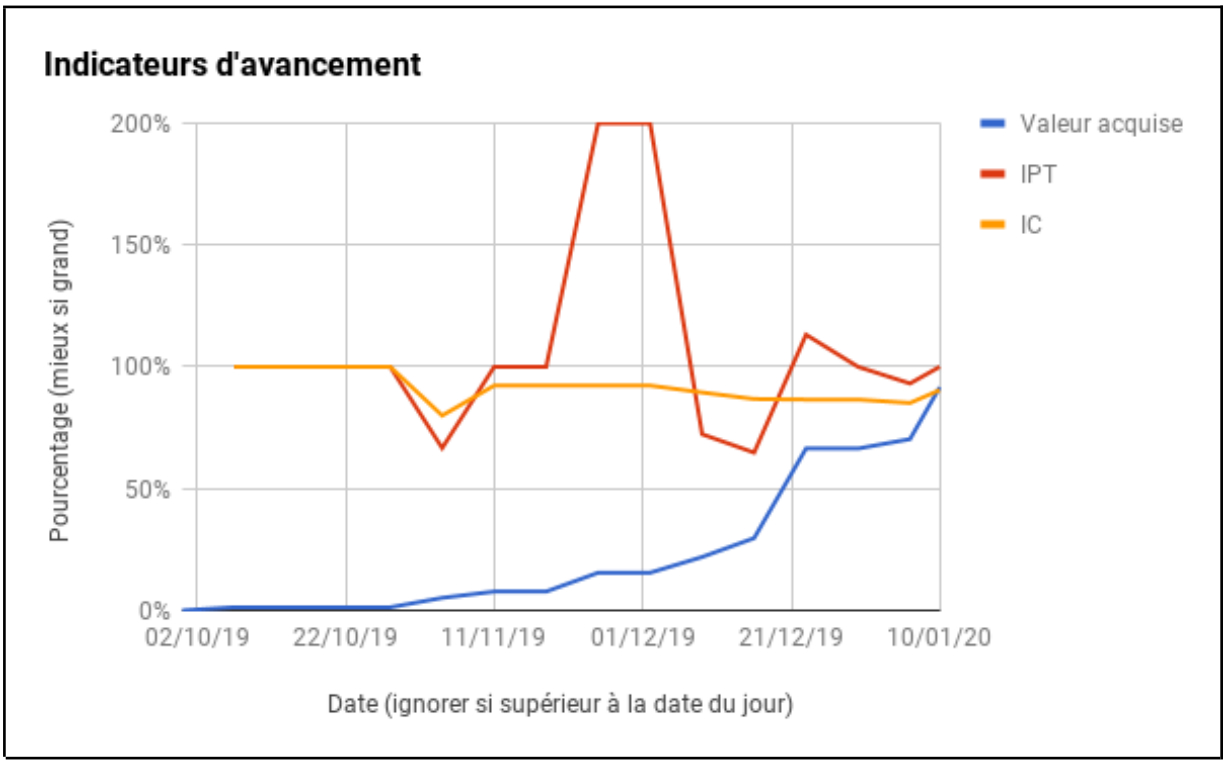


Diagramme de GANTT (Cliquez pour zoomer ctrl + clic) :

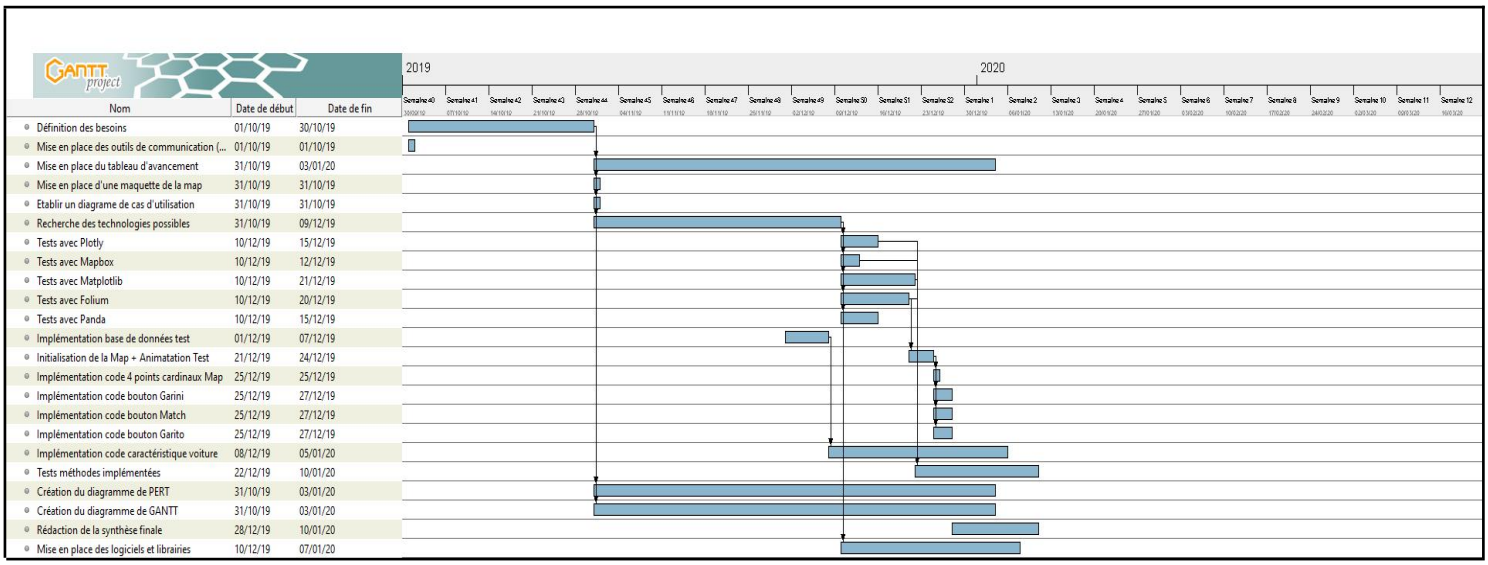
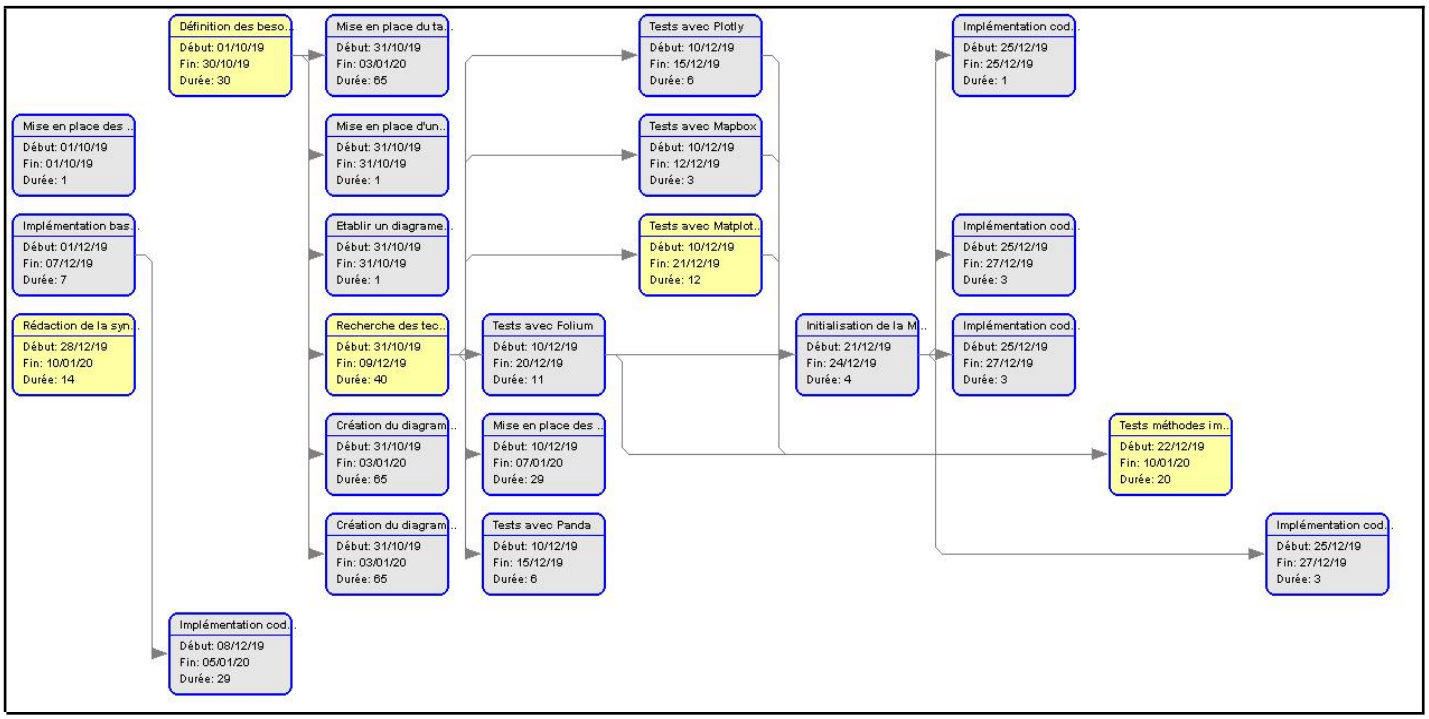


Diagramme de PERT (Cliquez pour zoomer ctrl + clic) :



Maquette de la map (Cliquez pour zoomer ctrl + clic) :

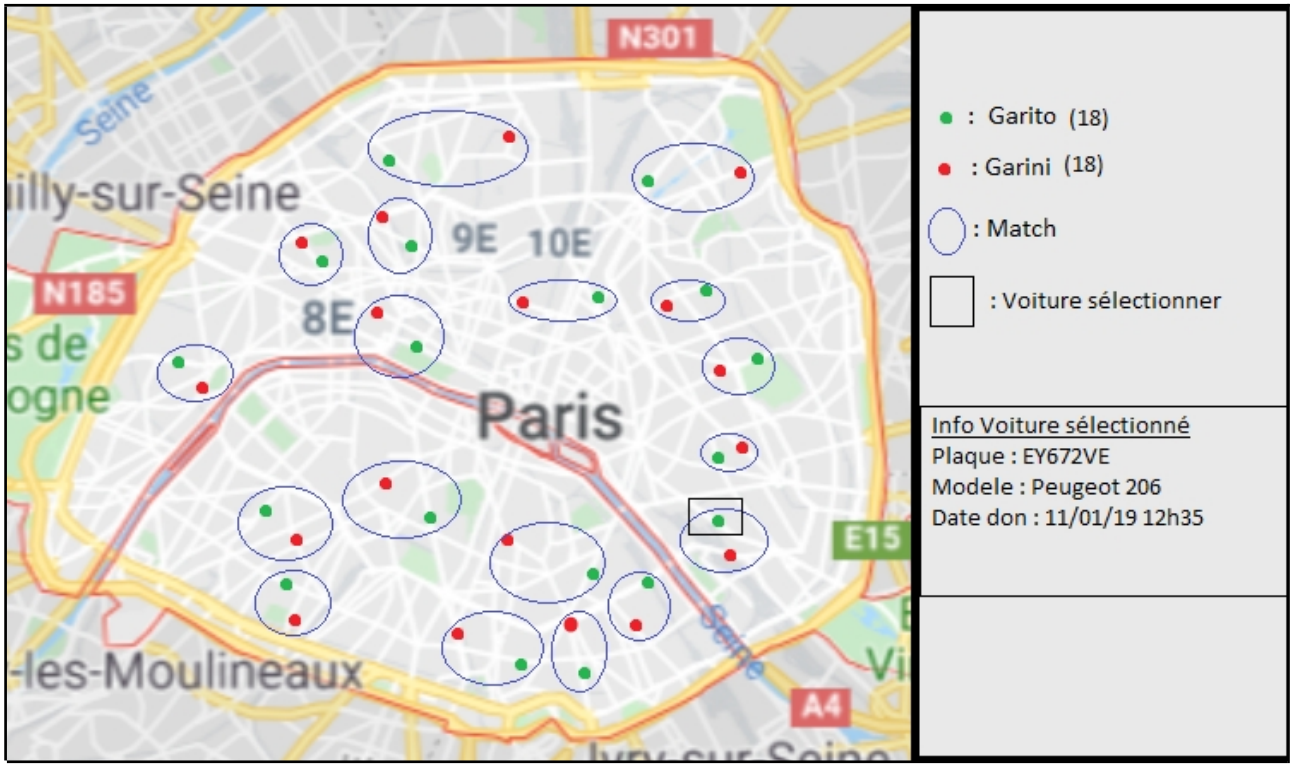
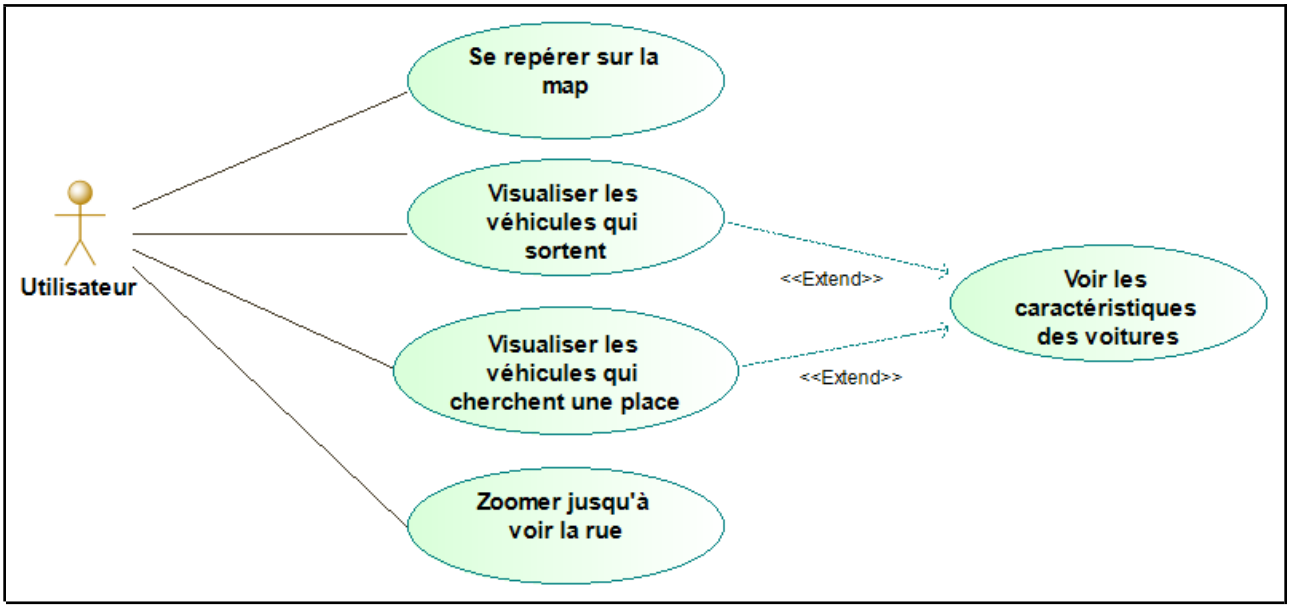


Diagramme des cas d'utilisation :



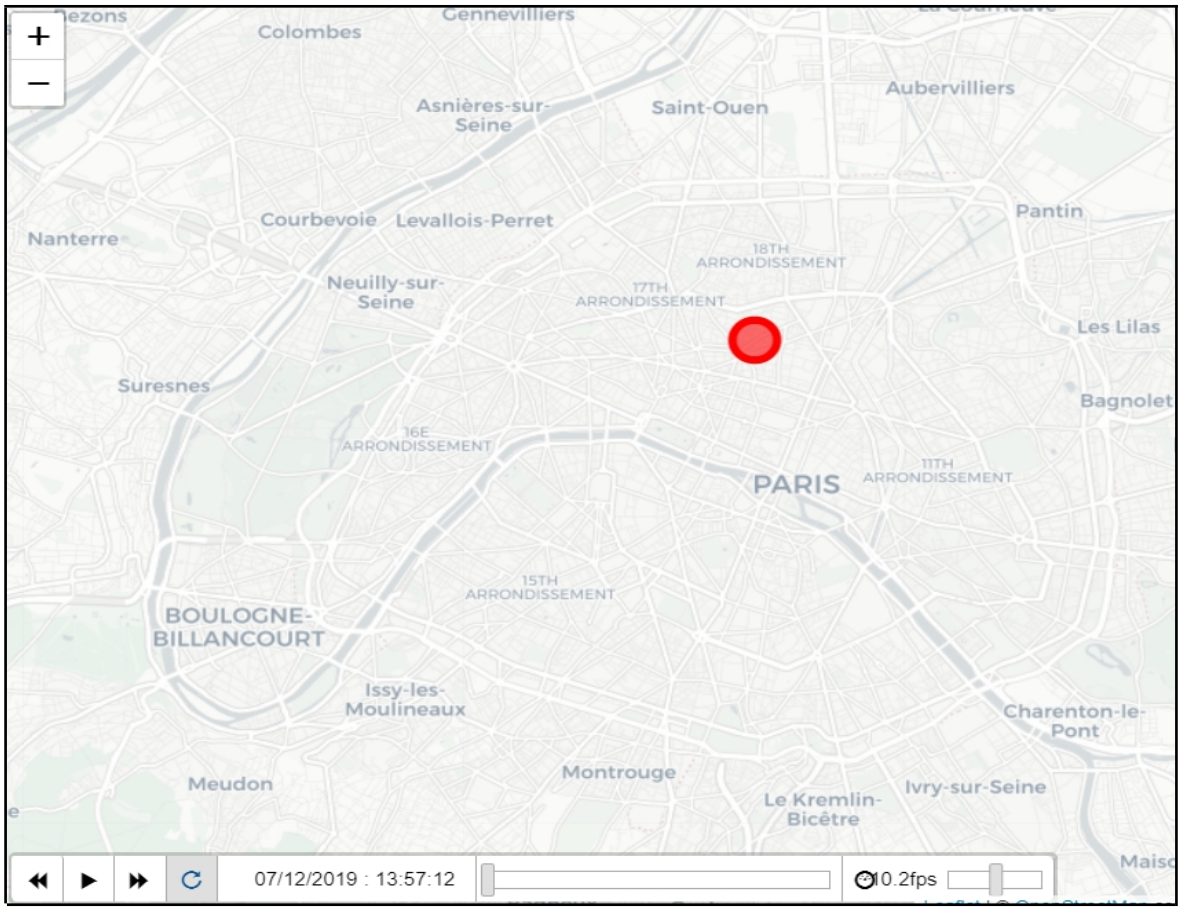
Extrait de notre base de données de test (Cliquez pour zoomer ctrl + clic) :

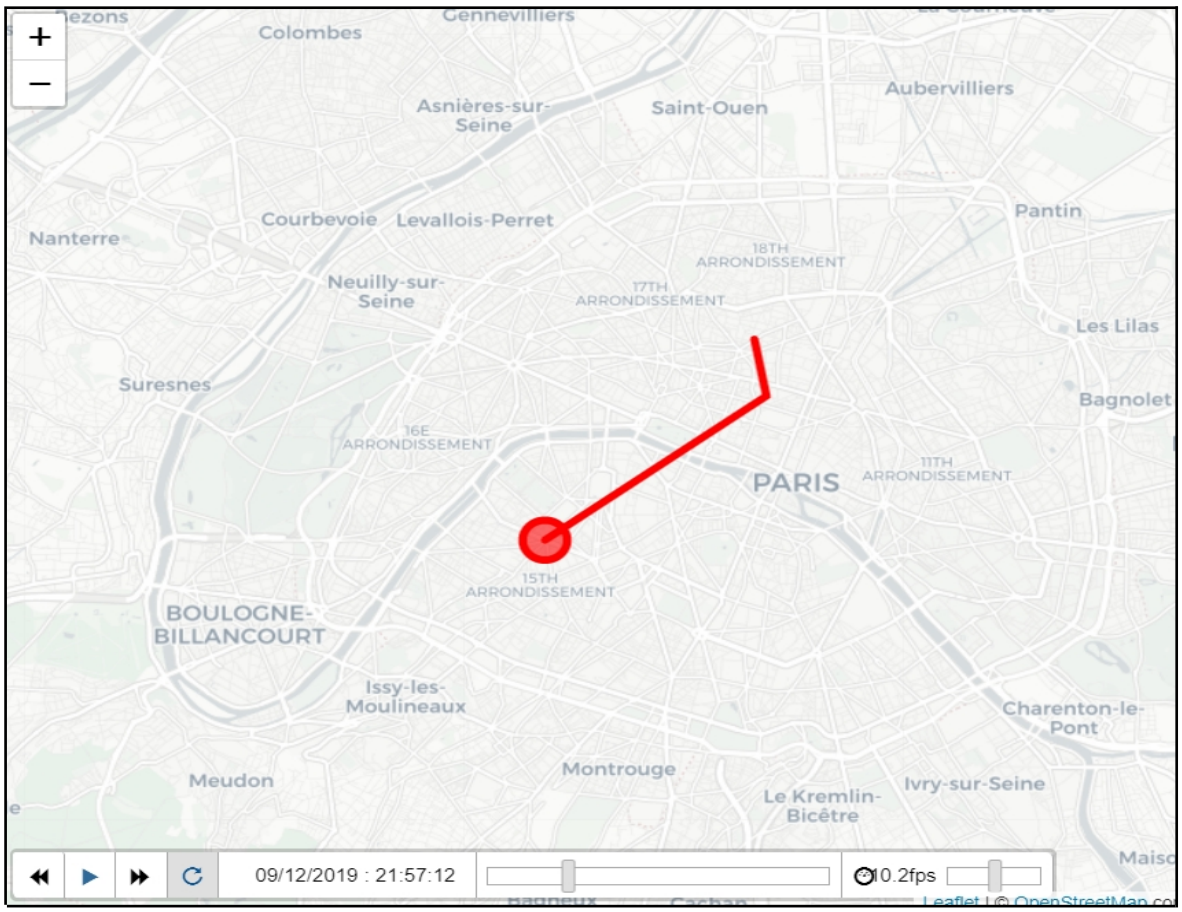
```sql
CREATE TABLE `donner` (
  `id` bigint(64) UNSIGNED NOT NULL,
  `lat` double NOT NULL,
  `lng` double NOT NULL,
  `adresse` varchar(255) NOT NULL,
  `date_heur` datetime NOT NULL,
  `heur_liberer` datetime NOT NULL,
  `id_user` bigint(64) UNSIGNED NOT NULL,
  `etat` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `donner` (`id`, `lat`, `lng`, `adresse`, `date_heur`, `heur_liberer`, `id_user`, `etat`) VALUES
(1, 48.869240, 2.345665, '17 Rue des Jeuneurs, 75002 Paris', '2019-12-08 10:55:12', '2019-12-08 10:55:12', 2, 1),
(2, 48.871928, 2.348330, '46 Rue de l\'Échiquier, 75010 Paris', '2019-12-09 18:38:41', '2019-12-09 18:38:41', 3, 2),
(3, 48.866348, 2.334195, '24 Avenue de l\'Opéra, 75001 Paris', '2019-12-10 06:19:09', '2019-12-10 06:19:09', 4, 1),
(4, 48.844017, 2.382108, '133 Avenue Daumesnil, 75012 Paris', '2019-12-11 09:40:28', '2019-12-11 09:40:28', 5, 2),
(5, 48.836328, 2.326721, '47 Rue Froidevaux, 75014 Paris', '2019-12-12 15:30:35', '2019-12-12 15:30:35', 6, 1),
(6, 48.857998, 2.300115, '52 Avenue Rapp, 75007 Paris', '2019-12-13 12:12:12', '2019-12-13 12:12:12', 7, 2),
(7, 48.858122, 2.381906, '119 Rue de la Roquette, 75011 Paris', '2019-12-14 05:29:31', '2019-12-19 05:29:31', 101, 1),
(8, 48.839173, 2.359177, '5 Boulevard Saint-Marcel, 75005 Paris', '2019-12-15 06:30:31', '2019-12-20 06:30:31', 102, 1),
(9, 48.830906, 2.312426, '151 Rue Raymond Losserand, 75014 Paris', '2019-12-16 07:40:31', '2019-12-21 07:40:31', 103, 1),
(10, 48.849368, 2.292184, '3 Rue Fallempin, 75015 Paris', '2019-12-17 08:25:31', '2019-12-22 08:25:31', 104, 1),
(11, 48.882328, 2.304256, '51 Rue de Prony, 75017 Paris', '2019-12-18 09:55:31', '2019-12-23 09:55:31', 105, 1);
```

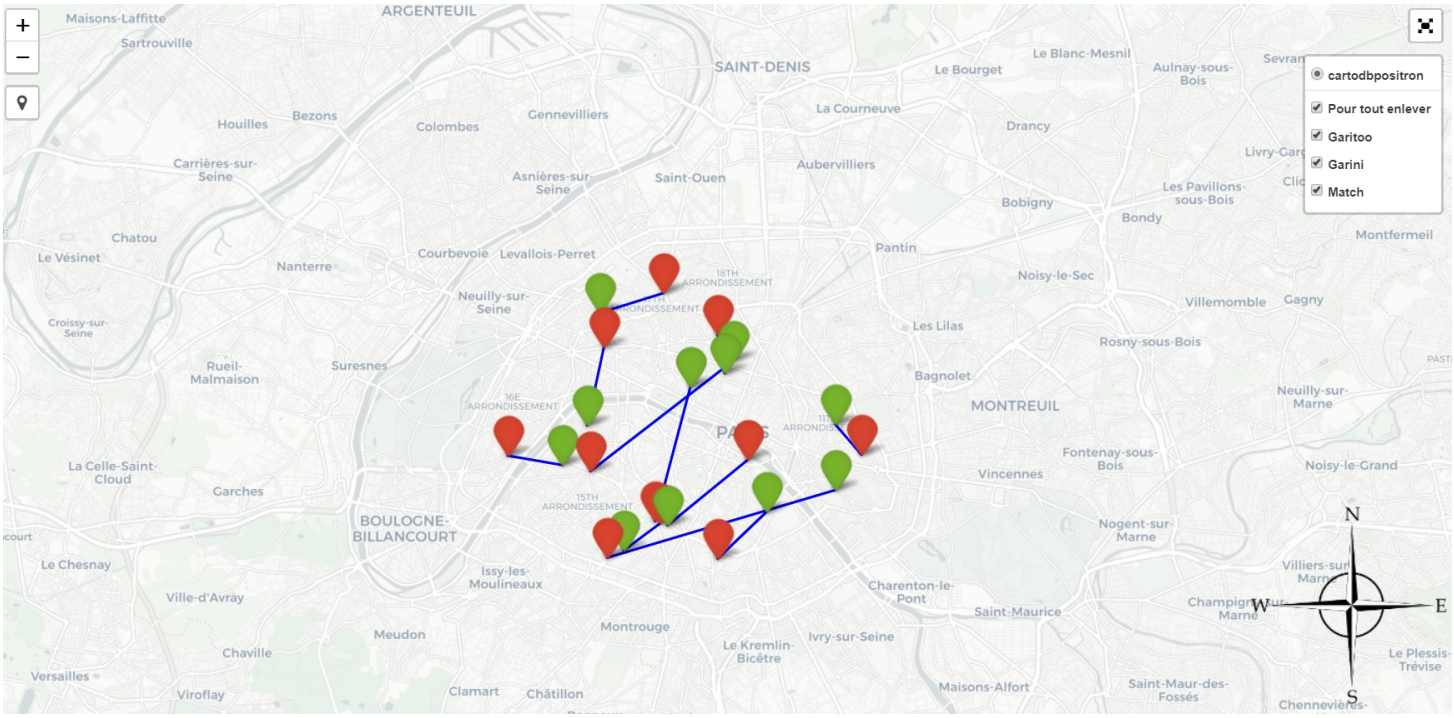
Test animation avec Matplotlib et BaseMap :

Animation avec Folium + plugins.TimestampedGeoJson → au début puis 3 secondes après :





Finalement la map finale (Cliquez pour zoomer ctrl + clic) :



Groupe B - Gari