Rapport d'analyse informatique

Ingénieur 2ème année

Avril 2025

Mobility

Amélioration de la librairie Python open-source Mobility

Anaïs Floch

Liam Longfier

Gabin Potel

Commanditaire :

Capucine-Marin Dubroca-Voisin (AREP)

capucine-marin.dubroca-voisin@arep.fr

Table des matières

I. Contexte du projet

A. L'entreprise AREP

Mobility est développée par l'agence AREP et l'atelier d'ingénieur Elioth du groupe Egis. L'AREP (Architecture, Recherche, Engagement Post-carbone) est une agence pluridisciplinaire filiale de la SNCF Gares et Connexions, portée sur l'architecture, l'urbanisme, le design, l'ingénierie, la programmation, le conseil et le management de projet. Ces compétences sont alliées dans le but de répondre de manière concrète au défi que représente le réchauffement climatique.

La bibliothèque Mobility est en cours de développement. Notre équipe intervient dans ce projet dans le but de réaliser un prototype d'interface graphique servant de support à l'outil. Pour l'instant, la bibliothèque Mobility est assez compliquée à prendre en main et à installer.

B. Mobility

La bibliothèque Mobility a pour but de permettre l'étude de la mobilité de différentes personnes pour divers motifs à des échelles multiples, de l'individu à la population, du bâtiment au territoire. Son objectif premier est d'être une aide à la décision pour les collectivités ou encore les bureaux d'étude. Elle peut aussi être utilisée dans le cadre de la recherche par exemple.

C. Problématique du projet

Par son objectif, cette bibliothèque peut être amenée à être utilisée par beaucoup de personnes différentes, plus ou moins à l'aise avec l'informatique. Pour l'instant, la bibliothèque Mobility requiert plusieurs étapes d'installation, dont le cheminement n'est pas des plus évidents, ce qui peut mettre en péril son accessibilité alors qu'il serait bénéfique que cet outil soit disponible au plus grand nombre.

Outre l'installation, il peut être difficile de comprendre exactement comment mettre en place une analyse poussée avec cet outil, en raison du nombre de paramètres qu'il propose.

Nous avons fait le choix de réaliser une interface graphique pour cet outil afin de le rendre plus accessible et de permettre à n'importe qui de l'utiliser plus facilement.

D. Besoins du commanditaire

L'AREP demande un prototype d'interface graphique interactif, permettant de générer des simulations de déplacements en fonction des moyens de transports utilisés, des motifs de déplacement disponibles ou encore des catégories socioprofessionnelles des usagers à partir de la librairie Python Mobility. Les fonctionnalités sont amenées à se diversifier par le futur, ainsi que les types de simulations générées. Les simulations incluraient par exemple des graphes de probabilité d'aller travailler aux alentours d'une zone de départ sélectionnée en fonction des modes de transport proposés.

Ce projet s'adresse au grand public, il est donc nécessaire de créer un prototype d'interface WEB épuré et intuitif. L'utilisateur ne doit pas être noyé sous le choix de paramètres apparents : une simulation préchargée par défaut doit apparaitre sur la page WEB, où il est par la suite invité à spécifier ses propres besoins pour adapter la simulation à sa situation. Chaque paramètre devra être expliqué afin de ne pas perdre l'utilisateur dans ses démarches.

Le multilinguisme est recherché afin d'ouvrir des perspectives internationales. L'AREP est attachée au caractère open-source dans une idée de développement des communs numériques, et se veut transparente dans les données utilisées. Il est donc nécessaire que n'importe quel utilisateur puisse accéder aux données qui ont permis de réaliser sa simulation.

E. L'équipe projet

L'équipe travaillant sur le projet se compose d'Anaïs Floch, cheffe de projet, Liam Longfier en tant que développeur WEB et Gabin Potel, WEB Designer.

La cheffe de projet apporte sa polyvalence dans la gestion des relations avec la commanditaire, des phasages et outils du projet, de la création de modèles et des tests sur des utilisateurs réels.

Le développeur utilise son adaptabilité informatique pour la compréhension de la bibliothèque Mobility, le choix d'un Framework approprié au développement de l'interface, sa réalisation et ses tests unitaires.

Le WEB Designer s'occupe principalement de l'esthétique de l'interface, la vérification des modèles sur les personas créés et la rédaction du rapport.

F. Définitions importantes

Ce projet comporte des termes techniques liés aux mobilités qui vont être définis ci-dessous. La bonne compréhension de l'analyse menée ne nécessite pas la maîtrise de ceux-ci.

Mobilité : désigne un changement de lieu accompli par une ou des personnes. On peut en distinguer 3 types :

- Les mobilités définitives ou de longue durée (ex : migration)
- Les mobilités temporaires (ex : tourisme)
- Les mobilités quotidiennes (ex : trajet domicile-travail)

Part Modale : répartition d'utilisateurs dans chaque mode de transport pour une mobilité donnée. Elle permet d'analyser les transports théoriquement privilégiés par les utilisateurs. Cela permet notamment d'identifier les infrastructures à développer ou améliorer pour que les utilisateurs se tournent vers des modes de transport moins polluants.

II. Objectifs de l’étude – Reformulation du besoin

A. Les objectifs de l’étude

Ce projet ambitieux comporte de nombreux objectifs :

• Créer un prototype d'interface WEB fonctionnel

• Proposer des options interactives aux utilisateurs selon les paramètres disponibles avec Mobility

• Expliquer chaque paramètre à l'utilisateur s'il en a besoin au moyen d’infobulles

• Pouvoir télécharger les données utilisées lors des simulations dans une optique de transparence

• Diversifier les langues pour augmenter l’accessibilité de l’outil

B. Les contraintes

Le projet est assez libre dans l’ensemble, les seules contraintes sont de :

• Générer automatiquement une simulation préchargée en arrivant sur la page WEB, afin de proposer un exemple à l’utilisateur

• Relier la librairie Mobility en Python pour le back-end, étant donné que c’est le langage utilisé par la bibliothèque

• Limiter le temps de calcul, pour permettre une plus grande fluidité dans l’utilisation de l’outil

• Produire un code open-source, compréhensible et commenté, afin de s’inscrire dans la lignée de ce qui a déjà été fait

• Définir les textes et infobulles de l’interface WEB au format JSON respectant la norme I18N

C. Le recueil du besoin – Les acteurs

Cette bibliothèque est libre d’accès, et le but en créant une interface graphique est de la rendre accessible au plus grand nombre.

Pour essayer de créer une solution aussi accessible que possible, il est prévu d’avoir des discussions avec les commanditaires, certains de leurs clients et également des personnes complètement extérieures à Mobility (ici, des élèves de l’école). De plus, afin de couvrir la majorité des cas d’utilisation de Mobility, il a été décidé de créer 4 personas : des profils types d’utilisateur pour lesquels il faut envisager les besoins, les contraintes, etc.

Ces 4 profils sont donc :

Camille, 33 ans

- chercheur.euse
- travaille sur l'empreinte carbone (en général)
- intérêt pour les CSP
- travaille sur le territoire global

Philémon, 25 ans

- étudiant curieux
- peut perdre patience
- focus sur un territoire (point nodal)

Josette, 58 ans

- travaille en collectivité
- veut anticiper ses mobilités
- travaille sur son territoire

Medhi, 42 ans

- cabinet de conseil
- besoins de **toutes** les infos **vite**
- un projet +/- précis

Ces profils ont été étoffés au cours du projet afin de les compléter, de préciser leurs besoins et leurs attentes. La version finale peut être retrouvée en annexe si nécessaire, mais leurs versions initiales (ci-dessus) suffisent à justifier les choix réalisés, notamment en termes de design de l'interface.

Dans le diagramme ci-dessous sont listées les possibilités dont un utilisateur lambda pourrait avoir besoin.

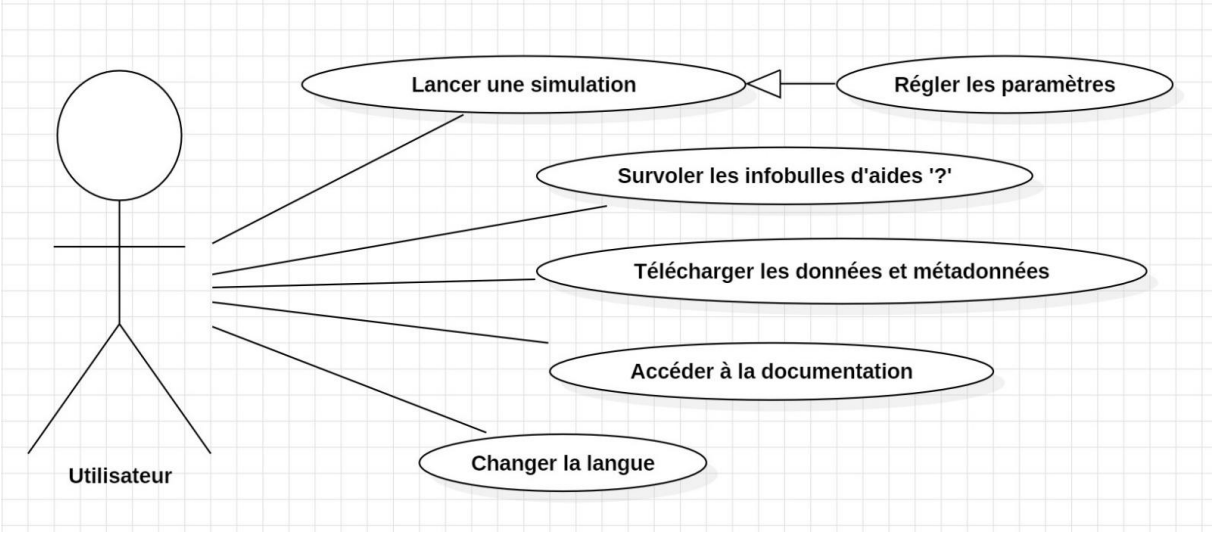

Figure 1 - Diagramme de cas d'utilisation

III. Analyse fonctionnelle – Solutions proposées

Ce projet s'inscrit dans le besoin de la part des commanditaires de développer une interface WEB pour Mobility, il est question de réaliser le tout premier prototype de cette interface. Par conséquent, le choix du design de la page WEB est assez libre. Etant donné qu'il est nécessaire que cette interface soit accessible à tous, il a été décidé de partir sur un style épuré et d'éviter toute surcharge inutile. La première maquette ci-dessous va en ce sens.

Figure 2 – Première maquette de l'interface WEB

Outre l'aspect visuel de la solution, il est nécessaire de rendre l'outil Mobility accessible sur le fond également. Pour ce faire, il est envisagé d'expliciter chacun des paramètres à l'aide de boutons « ? » qui feront apparaître une infobulle lorsque les utilisateurs les survoleront. Le texte de ces infobulles sera soumis à la validation des commanditaires, mais aussi de personnes extérieures afin de s'assurer qu'il soit compréhensible par tous.

La bibliothèque Mobility est développée en open-source, et donc attachée à la transparence des données. Pour cela, il est prévu de rendre possible le téléchargement des données utilisées pour les simulations, et des résultats (cartes, graphes) issus de ces simulations.

Au cours de la phase de conception de l'interface, les maquettes ont été amenées à évoluer ; entre autres en raison de retours de la part des commanditaires, mais également dans le cadre de l'utilisation de personas : il a fallu modifier quelques aspects du design pour convenir au mieux à nos 4 profils. Lors d'une réunion avec l'un des clients des commanditaires – utilisateur de Mobility – la maquette finale a reçu des retours très positifs, la pertinence des personas a également été soulignée.

On peut voir sur la maquette finale ci-dessous que plusieurs modifications ont été apportées. D'abord, le choix a été fait de cocher tous les paramètres possibles au chargement de l'interface, essentiellement parce que les utilisateurs ont tendance à étudier d'abord les mobilités dans leur globalité avant de se concentrer – ou non – sur des points plus précis. Il est également prévu d'ajouter une option permettant de tout décocher rapidement. De plus, le volet « Zone d'étude » a été revu entièrement car il omettait plusieurs cas de recherche, ce qui a été remarqué notamment à l'aide des personas dont les types de zones étudiées diffèrent. Enfin, on peut constater l'ajout d'autres infobulles et de la section « Autres paramètres », destinées aux utilisateurs aguerris de l'outil qui chercheraient à l'utiliser de manière plus poussée que

l'utilisateur lambda. Ces autres paramètres pourraient comprendre la possibilité de choisir une zone tampon autour des communes ou départements qui seraient inclus dans la simulation.

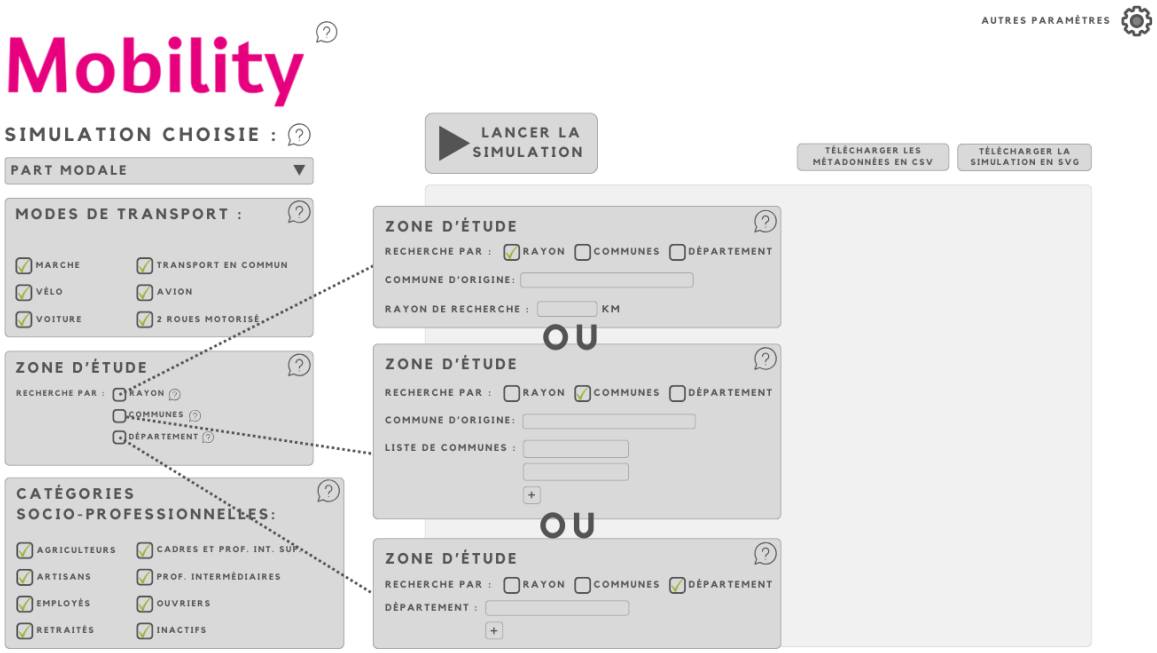

Figure 3 - Maquette finale de l'interface WEB

IV. Etude technique : Choix des logiciels et langages – Architecture

La librairie Mobility est une bibliothèque développée en Python, les commanditaires ont donc demandé à ce que ce projet reste dans le même langage.

Ce simple diagramme de composants (Figure 4) représente la liaison entre les différents éléments de l'interface WEB. Lorsque l'utilisateur démarre son navigateur internet pour se connecter à l'application WEB, il rencontre l'interface utilisateur, un script Mobility et des données précalculées. L'interface questionne Mobility pour lancer des simulations et celui-ci vérifie la présence de résultats précalculés dans la base de données, sinon il les calcule puis les renvoie à l'utilisateur.

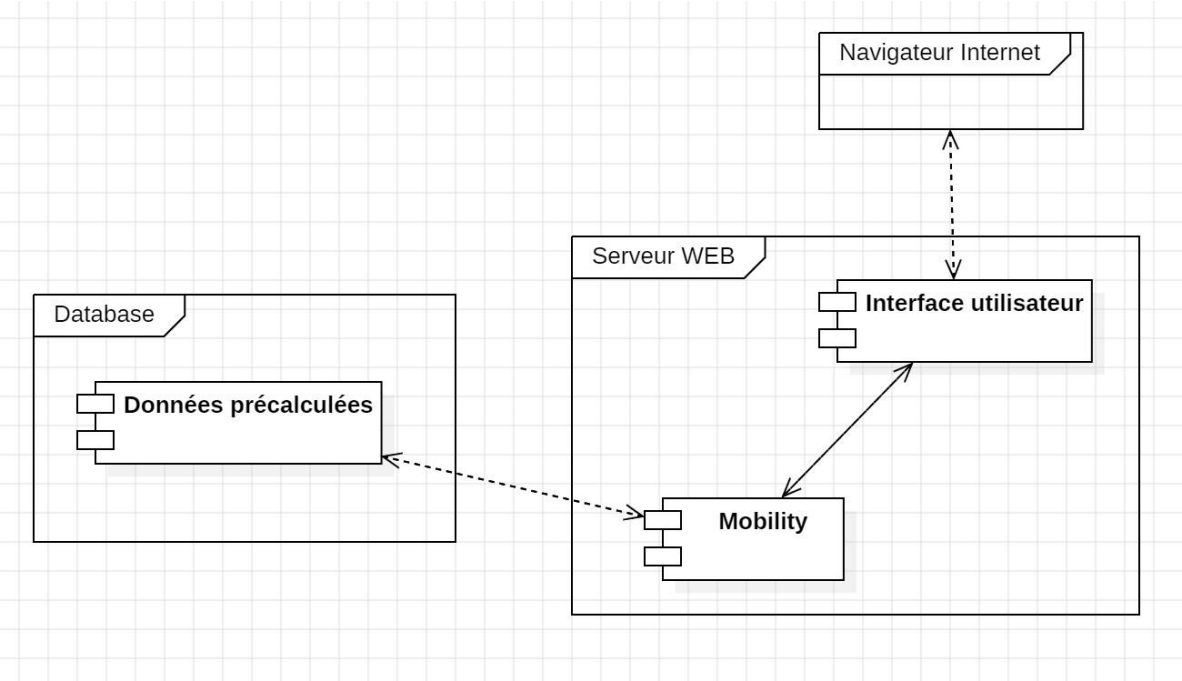

Figure 4 – Diagramme de composants

Le diagramme d'état ci-dessous illustre le cheminement d'un utilisateur lambda. En arrivant sur l'interface WEB, les cadres contenant tous les paramètres pré-cochés apparaissent, lançant le début de la simulation. Comme la simulation choisie est déjà présente dans la base de donnée, le résultat précalculé est affiché. L'utilisateur peut télécharger les données s'il le souhaite, et est invité à personnaliser ses paramètres afin de relancer une simulation. Si les résultats ne sont pas présents dans la base de données, le calcul va être réalisé par Mobility avec les paramètres choisis puis affichés.

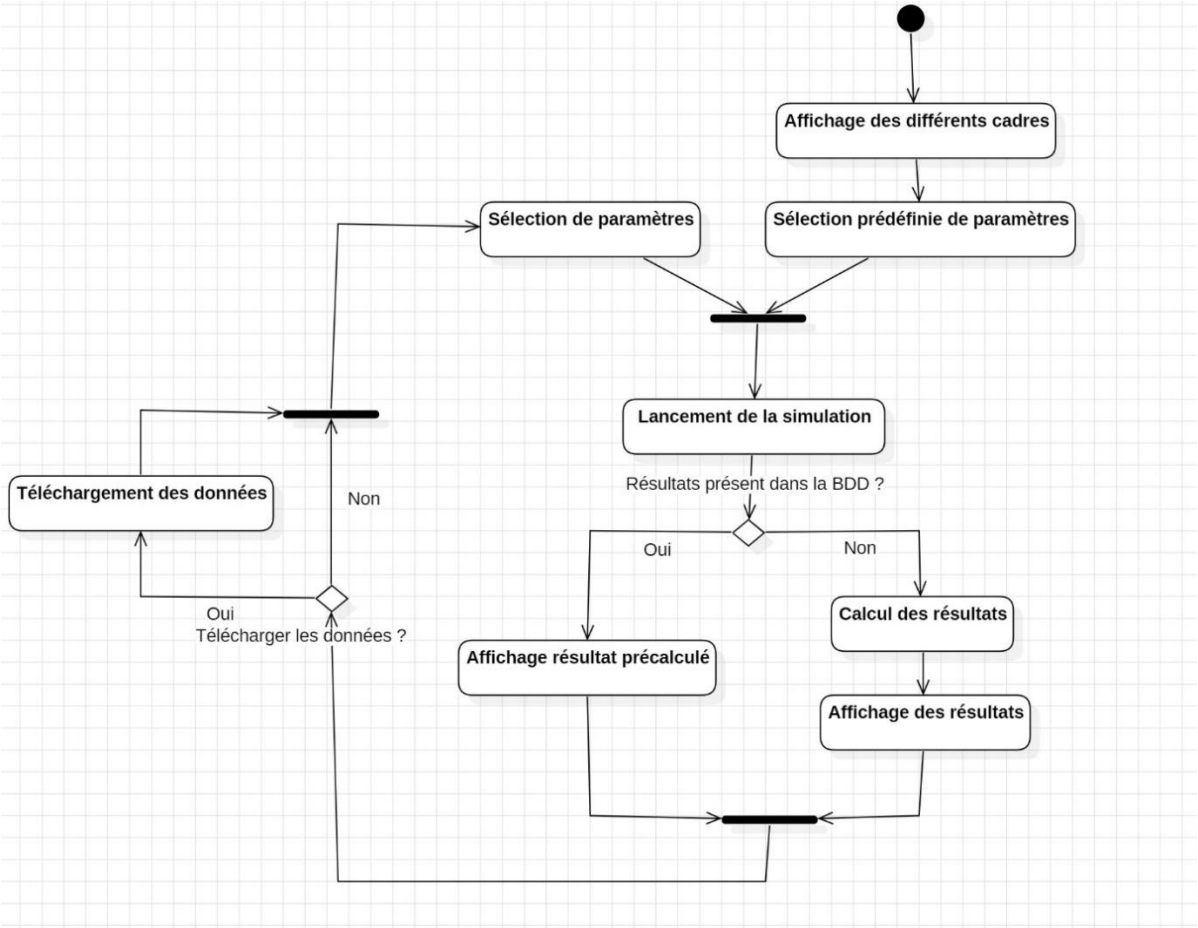

Figure 5 – Diagramme d'état

Afin de réaliser une interface WEB, il est nécessaire d'avoir recours à un Framework Python permettant de faire du développement WEB. Pour faciliter le choix de ce Framework, des comparaisons entre différentes solutions existantes ont été réalisées :

Critères	Pertinence du critère	Flask	FastAPI	CherryPy	Django	Pyramid	web2py	Dash
Facilité d'utilisation	Forte	Très facile	Facile	Très facile	Moyenne	Moyenne	Très facile	Très facile
Documentation / Information	Forte	Très bonne	Moyenne	Bonne	Très bonne	Bonne	Bonne	Bonne
Version / License	Forte	3.1.0 / BSD	0.115.8 / MIT	18.10.0 / BSD	5.1.6 / BSD	2.0.2 / BSD	2.27.1 / LGPL3	2.18.2 / MIT
Objectif	Moyenne	Création très facile d'un projet évolutif	Facile et basé sur le type hints	Facile pour utilisateurs python	Facilité des sites complexes	Entre Megaframwork et Miniframework	Pour application basée sur des databases	Affichage graphes, curseurs et liste déroulantes
Performance	Moyenne	Bonne	Excellente	Bonne	Excellente	Bonne	Moyenne	Bonne
Evolutivité	Moyenne	Forte	Excellente	Moyenne	Excellente	Forte	Moyenne	Moyenne
Github Stars	Moyenne	69k	81.8k	1.9k	82.6k	4k	2.1k	22.1k
Tâches asynchrone	Faible	Présente	Présente	Absente	Présente	Présente	Présente	Présente
Gestion de session	Faible	Présente	Présente	Présente	Présente	Présente	Présente	Non présente
Sécurité	Faible	Forte	Faible	Faible	Très forte	Faible	Forte	Faible

Figure 6 - Tableau comparatif des différents Framework disponibles

Les critères « facilité d'utilisation » et « documentation/information » étaient importants dans le cadre d'un projet aussi court : un Framework facile d'utilisation était essentiel pour permettre de respecter au mieux les échéances et de fournir un prototype complet. D'autres critères, intéressants car permettant de réduire les choix sont « objectif » et « Github Stars », respectivement pour trouver un outil adapté et pour s'assurer de la popularité du Framework, également signe de sa pertinence.

Deux possibilités semblaient ressortir : Flask et Django, suivies par Dash et CherryPy. Le choix du Framework a été réalisé de concert avec les commanditaires, qui ont révélés être familiers de Dash. Même si d'après les critères sélectionnés il n'est pas le Framework qui répond le plus à nos attentes, le choix de Dash a été fait, afin de faciliter l'appropriation du code par les commanditaires à la suite du rendu final du projet.

Le code du projet se décompose de la manière suivante :

1. Le style

Situé dans le dossier assets et en langage CSS, il est automatiquement utilisé par Dash. Comme une seule page WEB est utilisée, un seul fichier de style a été créé. Cependant il est bon de noter qu'avec plus de temps et d'attention à l'évolutivité du projet, d'autres fichiers auraient pu être créés pour décomposer la gestion du style entre un fichier de style commun à notre interface WEB et les potentielles pages futures, et un fichier de style propre à cette interface.

2. Stockage des différents contenus et identifiants de la page

Toutes les informations relatives au contenu de la page comme les classes et identifiants sont stockés dans des fichiers au format .json à la demande des commanditaires dans un effort de respect de la norme I18N et de facilité d'adaptabilité et de réutilisabilité du code. Pour faciliter l'utilisation de ces informations, une nomenclature commune a été mise en place, disponible dans un tableur sur le Git.

3. Code source de la page WEB

Le code source est constitué de plusieurs parties détaillées ci-dessous :

- Une initialisation de Dash

- Une initialisation de l'architecture HTTP en langage Dash en incluant les différents contenus, classes et identifiants récupérés des fichiers au format .json

- Différentes fonctions "callbacks" qui permettent l'interactivité de la page, notamment l'ajout de boîtes de saisie pour renseigner plusieurs communes/départements ou le lancement du script Mobility lors d'un clic sur le bouton « commencer la simulation »

V. Réalisation et suivi de projet

A. Les risques

Les risques ainsi que certaines solutions ont été étudiés dans le tableau ci-dessous.

Nature du risque	Description	Aléa	Vulnérabilité	Solution	Risques	Evolution
Humain	Ne pas finir le projet à temps	Très probable	Moyenne	Définir les priorités du projet	Important	
Logistique	Erreur dans l'import sur serveur	Peu probable	Très élevée	Prendre le temps de réaliser les tests	Important	
Humain	Ne pas satisfaire la demande du commanditaire	Peu probable	Elevée	Réunions régulières pour faire un point d'avancement	Moyen	
Scientifique / Ergonomique	Faire un site qui met beaucoup de temps à calculer	Probable	Peu élevée	Prévoir une solution pour prévenir l'utilisateur du temps d'attente	Faible	
Humain	Manque de communication dans l'équipe	Improbable	Elevée	Réunion en début et fin de journée	Faible	
Scientifique	Ne pas trouver les meilleurs outils pour réaliser le projet	Peu probable	Peu élevée	Comparer les possibilités	Très faible	

Figure 7 - Matrice des risques

Les risques auxquels notre projet a été confronté sont :

- **Ne pas finir le projet à temps :** Le projet est très ambitieux, et le temps dédié assez restreint. Des priorités ont été définies avec les commanditaires afin de limiter ce risque. Cependant, en raison d'un accès aux données tardif, le projet a pris un peu de retard du point de vue du développement WEB. Le résultat obtenu est tout de même globalement en accord avec les priorités établies.
- **Erreur de connexion entre le site Web et le serveur :** En raison d'un accès aux données tardif, il n'a pas été possible d'apporter autant de soin qu'il aurait fallu à la liaison entre l'interface et Mobility. Néanmoins, la connexion est fonctionnelle et lancer une simulation génère bel et bien un graphe de probabilité remplissant l'objectif premier de ce projet. Le passage entre une solution locale et un serveur n'a pas pu être réalisé, ce risque demeure donc tel qu'il est, faute d'avoir pu s'y confronter.
- **Ne pas satisfaire la demande du commanditaire :** Au cours du projet, les attentes des commanditaires se sont précisées et un suivi a été mis en place, avec a minima une réunion par semaine. Ce risque a donc été dégradé car bien qu'il garde une Vulnérabilité élevée, son Aléa est devenu peu probable.
- **Faire un site qui met beaucoup de temps à calculer :** La bibliothèque Mobility met en jeu de nombreuses données, ce qui rend les calculs assez longs. Le risque serait alors de repousser les utilisateurs peu patients ; mais les utilisateurs qui ont vraiment besoin de l'outil sont tout de même plus susceptibles d'attendre. Il est possible de mettre en place des systèmes pour pallier cette attente, par exemple un mail envoyé à la fin des calculs qui invite l'utilisateur à revenir sur la page. Ce risque a été dégradé car la

longueur des calculs est indépendante de l'interface, c'est la bibliothèque qui influe dessus. Le seul risque pour notre projet serait de faire une interface qui met du temps à traiter les requêtes de l'utilisateur.

- **Manque de communication dans l'équipe :** Ce risque aurait pu être dérangeant, mais a été revu à la baisse étant donné la bonne entente de l'équipe, au sein et en dehors du projet.
- **Ne pas trouver les meilleurs outils :** Un travail a été réalisé sur le choix du Framework utilisé, en en comparant plusieurs et en se concertant avec les commanditaires. Bien que le choix final puisse sembler non idéal, c'est le meilleur compromis entre les différentes comparaisons et les souhaits des commanditaires. Ici, les « meilleurs » outils sont ceux qui conviennent à chaque acteur du projet.

B. Planning prévisionnel et explications

L'organisation de l'équipe a été abordée dans la partie I.E.

Il avait été prévu de communiquer entre élèves par les moyens habituels, avec les commanditaires par mail, Microsoft Teams lors des réunions et par le Github de Mobility (branche WebApp, dossier WebAppMobility) afin de partager le code produit dans le cadre du projet. Tous ces moyens de communication ont été respectés.

Les différentes tâches à accomplir au cours du projet ont été représentées sur le diagramme ci-dessous.

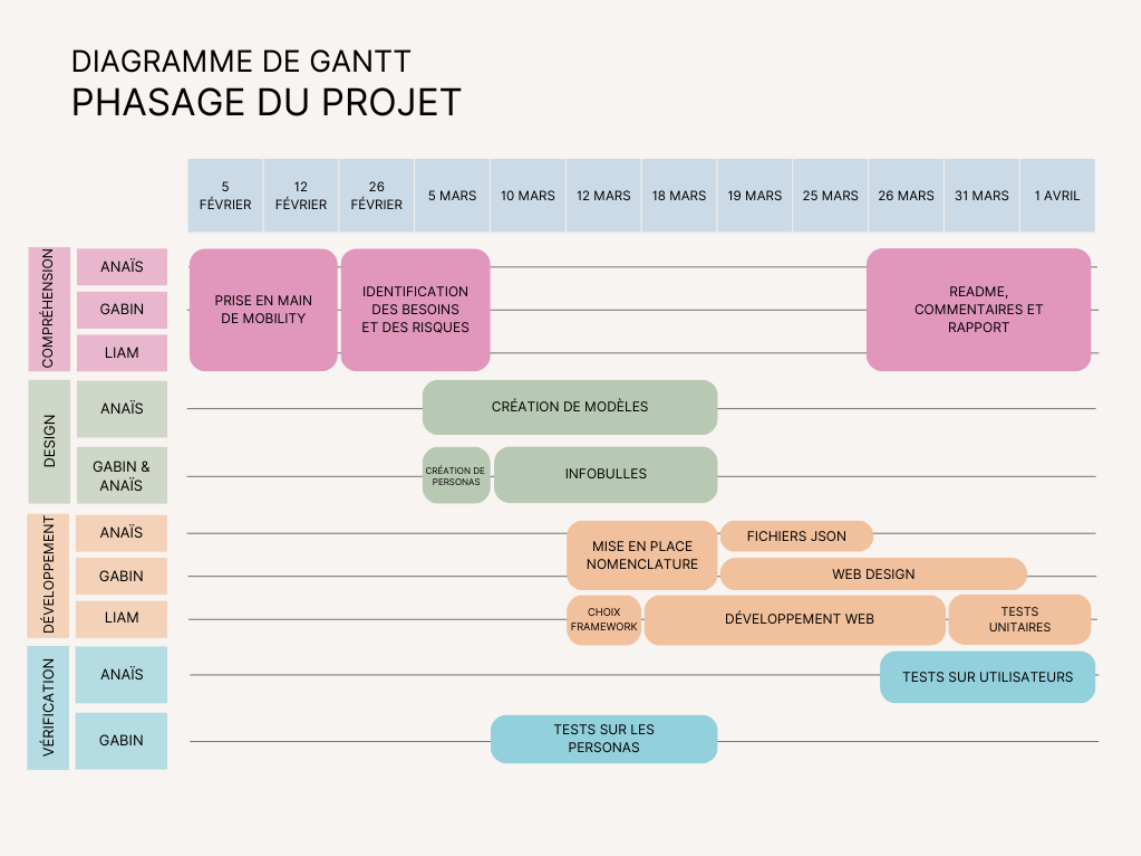

Figure 8 - Diagramme de Gantt

Ce diagramme est globalement resté le même qu'au début du projet, la phase de développement a simplement été légèrement allongée par rapport aux prévisions, réduisant de fait la durée accordée aux tests utilisateurs.

C. Les tests utilisateurs

Comme mentionné précédemment, les tests utilisateurs n'ont pu être menés dans leur intégralité, seuls une dizaine d'individus ont pu être sondés. Leur avis a cependant été complété par celui d'un client de notre commanditaire.

Ces tests ont permis de retenir plusieurs pistes d'amélioration :

- Pouvoir ajouter les listes de communes à partir d'un fichier .csv afin de traiter de plus gros volumes
- Compléter les zones d'étude pour des pays dont le découpage administratif diffère du système français
- Augmenter la visibilité des infobulles et de leur « ? » afin qu'ils ressortent mieux
- Permettre l'ajout d'un fond de carte au graphe généré
- Ajouter une boîte qui gère les leviers (zone à trafic limité, péage urbain, piste cyclable, …)

D. Réalisation finale

La figure ci-dessous illustre l'état final du prototype d'interface graphique de la librairie Mobility réalisé au cours de ce projet.

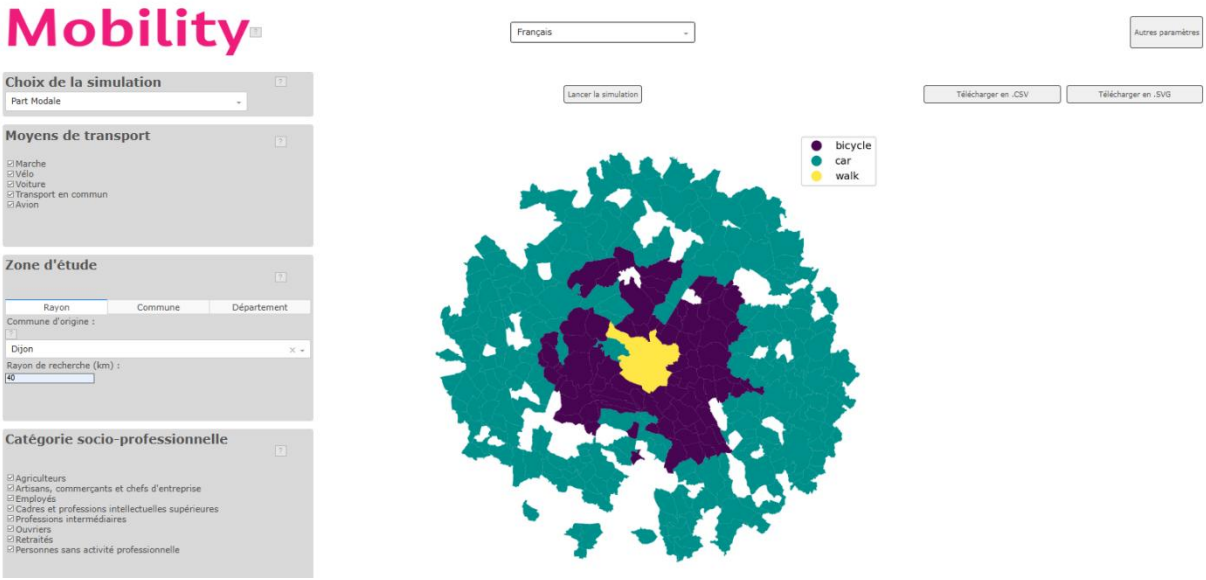

Figure 9 - Prototype final de l'interface

Le graphe affiché représente en son centre la ville de Dijon, et toutes les communes dans un rayon de 40 km aux alentours. Les différentes couleurs correspondent au moyen de transport le plus utilisé pour se rendre de Dijon à la commune colorée.

Il est bon de noter que cette simulation a été réalisée en affectant un poids très élevé à la marche, afin de distinguer 3 couleurs différentes. Elle ne représente donc pas de données réelles.

Les communes restées blanches quant à elles sont dues à la limitation du nombre de trajets envisagés entre chaque communes sélectionnées. Pour leur affecter une couleur, il suffirait d'augmenter ce nombre de trajets mais cela entrainerait un temps de calcul important. Le temps nous ayant manqué, il n'a pas été possible de le faire.

VI. Conclusion

Au terme de ce projet, un prototype d'interface graphique fonctionnel de la librairie Python Mobility a été créé. Le travail a pu être réalisé dans le temps imparti en dépit des contre-temps, en respectant assez fidèlement les phases prévues dans le diagramme de Gantt. Un peu de temps supplémentaire aurait pu permettre de peaufiner l'aspect de l'interface WEB, mais aussi de finaliser la liaison avec Mobility : à ce jour, les seuls paramètres fonctionnels sont la zone d'étude « Rayon », la commune et le rayon qui lui sont associés. Même s'il aurait été intéressant de pouvoir terminer cette liaison, la plus large partie du travail a été réalisée.

Ce projet nous a permis de mettre en œuvre les outils de gestion de projet étudiés en cours (calendrier prévisionnel, matrice de risques, recueil et interprétation d'un besoin, organisation d'un suivi avec des acteurs externes). De plus, le groupe a pu prendre en main le Framework Dash, utiliser le format JSON et mettre en place l'utilisation de personas, très pertinente dans le contexte de ce projet.

Enfin, nous souhaiterions remercier chaleureusement notre commanditaire Capucine-Marin Dubroca-Voisin pour son dynamisme, sa disponibilité, sa pédagogie ainsi que son ouverture aux propositions et idées, et Félix Pouchain pour sa réactivité à l'égard des problèmes rencontrés dans le code.

Annexe

Annexe 1 : Description détaillée des personas

Josette : A besoin étudier son territoire, c'est-à-dire sa commune (et peut-être quelques communes aux alentours) pour prévoir les déplacements pour la collectivité.
Peu à l'aise avec l'outil informatique (typiquement, ne serait pas capable d'utiliser la librairie avec Python).
Pour elle, l'interface doit être claire, intuitive et très facile d'utilisation.

Camille : Ne s'intéresse aux mobilités que du point de vue de la production de CO2. Son étude se concentre sur les Catégories Socio-Professionnelles donc le territoire étudié doit être global (au moins dans un premier temps).
A l'aise avec l'outil informatique, mais iel n'a pas le temps de se plonger dans la librairie Python étant donné son activité.
L'interface est ici un vrai gain de temps, elle doit faciliter l'utilisation de la bibliothèque.

Medhi : Ne cherche pas d'informations spécifiques, cela dépend du projet sur lequel il travaille. Dès qu'il sait ce dont il a besoin, il faut qu'il y ait accès *vite*.
Il n'a pas de temps à perdre à essayer de comprendre une librairie entière, ni même un site trop complexe : il faut que l'utilisation de l'interface soit *efficace*.

Philémon : S'intéresse aux mobilités autour d'un point spécifique (et dans un rayon défini sans doute). Il peut se plonger dans une librairie ou dans la documentation, mais il ne veut pas perdre son temps non plus.
Il faut un outil « ludique » afin de cultiver son intérêt et lui permettre d'étudier ce qu'il souhaite.

Annexe 2 : Sources utilisées pour la réalisation du tableau comparatif des Frameworks

https://en.wikipedia.org/wiki/Comparison_of_server-side_web_frameworks
https://github.com/pallets/flask
https://github.com/FastAPI/FastAPI
https://github.com/cherrypy/cherrypy
https://github.com/django/django
https://github.com/naksyn/Pyramid
https://github.com/web2py/web2py
https://github.com/plotly/dash
https://stackoverflow.com/questions
https://www.tresfacile.net/le-framework-cherrypy-python/
https://www.g2.com/products/cherrypy/reviews
https://www.geeksforgeeks.org/introduction-to-cherrypy/
https://careerfoundry.com/en/blog/web-development/django-framework-guide/#how-do-web-developers-use-python
https://www.capterra.com/p/234557/Django/reviews/
https://stackoverflow.com/questions/16301109/pyramid-is-slow-with-big-in-memory-object
https://www.tutorialspoint.com/web2py/web2py_security.htm
https://community.plotly.com/t/pros-cons-of-dash/62688
https://best-of-web.builder.io/library/plotly/dash
https://www.planeks.net/python-dashboard-development-framework/