Rapport d'analyse final
Projet "MapFlow"
Cartographie augmentée du logement étudiant français

MENGIN Alfred, GENET Arthur, JALLOUF Younes, SAMBOUN Christophe

Mai 2020

Table des matières

.

1 Contexte du projet

1.1 Les commanditaires du projets

Notre projet de développement informatique intitulé "Cartographie augmentée du logement étudiant français" a été proposé par l'entreprise *1630 Conseil* [1]. *1630 Conseil* a été créée en 2004 par Bertrand Moineau et Christine Silbermann. Il s'agit d'une entreprise de conseil oeuvrant dans trois sous-domaines : le conseil de direction, le conseil opérationnel et les études prospectives. Elle intervient dans les domaines publics comme privés et réalise plus particulièrement des études socioéconomiques. Chez *1630 Conseil* notre projet est suivi par Bertrand Moineau et Charles-Henri Arnould, associés.

1.2 Les origines du projet

L'un des thèmes important abordé par *1630 Conseil* est la gestion des logements étudiants. En effet, l'État et les collectivités territoriales cherchent à augmenter l'offre en logements étudiants en adéquation avec la réalité terrain. Or, les critères de décision ne sont actuellement basés que sur un simple croisement d'offres et de demandes. L'objectif de *1630 Conseil* est de fournir un outil de décision à ces acteurs publics, afin de mieux comprendre le monde étudiant et les différents flux qui l'animent quotidiennement. Afin de mener à bien cette mission de conseil, *1630 Conseil* a proposé à l'IGN de créer un démonstrateur cartographique permettant de visualiser les dynamiques de flux spatiaux-temporels des étudiants français.

1.3 Les enjeux du projet

Les enjeux de ce projet sont multiples. Premièrement, la carte dynamique permettra aux institutions publiques de promouvoir une politique d'aménagement des logements plus efficace. En comprenant davantage les migrations pendulaires des étudiants, l'État pourra motiver et argumenter ses orientations stratégiques. Deuxièmement, si le rendu du projet est suffisamment réussi, une commande publique pourrait être passée à l'IGN qui assurerait un déploiement industriel du projet.

2 Objectifs de l'étude

2.1 Les objectifs de l'étude

L'application finale devra contenir un certain nombre de fonctionnalités indispensables. L'application devra visualiser les facteurs sociologiques des étudiants (boursiers, situations familiales...), les facteurs académiques (disciplines, niveau d'études...), les localisations des étudiants (logement, école...) et les flux pendulaires reliant les points d'intérêts des étudiants. L'application devra être dynamique puisqu'elle représentera les flux sur un intervalle de temps donné. Enfin, elle doit présenter une vue paramétrable qui permettra de sélectionner l'emprise de la carte ou de sélectionner une certaine catégorie d'étudiants. Nos commanditaires souhaitent également que nous rédigions des scénario, afin de pouvoir présenter nos résultats dans un contexte précis.

2.2 Les contraintes

Nos commanditaires nous ont donnés carte blanche pour les développements à effectuer. Selon eux, seuls les résultats importent. Ils souhaitent cependant que l'application soit la plus esthétique et la plus efficace possible. Puisque nous devons gérer deux millions d'enregistrements dans la base de données, le système de gestion de bases de données se doit d'être suffisamment performant. De même, le logiciel de visualisation doit être suffisamment performant pour pouvoir afficher jusqu'à 2 millions de points dynamiques sur la carte. Pour réaliser toutes les étapes de ce projet, nous disposons d'un délai de 3 mois (analyse, développement).

2.3 Le recueil du besoin - Les acteurs

2.3.1 Identification des acteurs

Le commanditaire du projet est le cabinet d'étude *1630 Conseil*. Nos commanditaires souhaitent obtenir cette application afin de pouvoir décrire dans les grandes lignes les déplacements et la répartition des étudiants. Le projet doit pouvoir être présenté par nos commanditaires à deux ministères : le Ministère de la Cohésion des territoires et des Relations avec les collectivités territoriales, qui a en charge les politiques logement, et le Ministère de l'Enseignement supérieur et de la Recherche.

Il est donc important de remplir les attentes de *1630 Conseil* pour que le projet aboutisse. Il faut cependant aussi garder à l'esprit que leur objectif est de présenter les résultats aux instances publiques qui pourront être considérées comme des acteurs indirects.

Le cabinet *1630 Conseil* souhaite avoir un site WEB esthétique à présenter aux Ministères pour les faire adhérer au projet. Pour *1630 Conseil*, un site esthétique est un site qui propose un rendu optimal des différentes données. Les choix des couleurs, des types de représentation seront donc primordiaux pour afficher des données, puisque la couleur est porteuse d'information.

Puisque l'attente principale de nos commanditaires était esthétique, nous avons décidé d'accorder une part très importante sur la partie graphique du site WEB. L'exactitude de l'ensemble des données n'est pas primordiale même s'il faut des données statistiquement fidèles à la réalité.

2.3.2 Diagrammes des cas d'utilisation

Pour le premier diagramme des cas d'utilisation, nous allons nous baser sur le contexte de présentation envisagé par *1630 Conseil*. L'utilisateur du diagramme correspond donc à un conseiller de 1630 présentant l'application aux Ministères. Nos commanditaires devront pouvoir faire une démonstration interactive de l'application. Le système dispose également d'acteurs secondaires qui sont les fichiers de données. En effet, nous allons filtrer des fichiers de données pour afficher plus ou moins de données à l'écran. Voici les trois fonctionnalités principales de notre application.

- Nos commanditaires pourront présenter des informations statiques sur la carte. Cela inclura notamment la visualisation des différents lieux. Il sera possible de visualiser l'emplacement des lieux d'études des étudiants sur une période décidée à l'avance. Enfin, l'utilisateur sera en mesure de représenter tous les flux de manière statique pendant une période fixée.
- Nos commanditaires devront être capables de modifier les informations visibles à l'écran, via des cases à cocher, des boutons ou des curseurs. Ceci leur permettra d'afficher une catégorie d'étudiants selon des valeurs discrètes (boursier, filière...), selon des valeurs numériques (revenu fiscal du foyer de rattachement) ou encore selon l'intervalle de temps choisi.
- La dernière fonctionnalité majeure est la représentation d'informations dynamiques à l'écran. Contrairement aux informations statiques, l'objectif est de faire varier les informations que l'on voit à l'écran au cours du temps. Par exemple, il sera possible de visualiser le déplacement d'un étudiant entre les différents lieux qu'il fréquente au fil du temps.

Le deuxième diagramme a pour objectif de montrer les fonctionnalités octroyées à l'administrateur, une fois l'application développée. Ce diagramme explique donc la maintenance de l'application. L'administrateur disposera donc de 2 fonctionnalités majeures :

- L'administrateur pourra générer les données autant de fois qu'il le souhaite. Pour cela, il lui suffira de modifier les paramètres qu'il souhaite dans le code source ou la manière de générer certaines informations.

- L'administrateur pourra également découper les fichiers de données selon des mailles territoriales (selon les limites administratives des régions, ou selon un quadrillage).

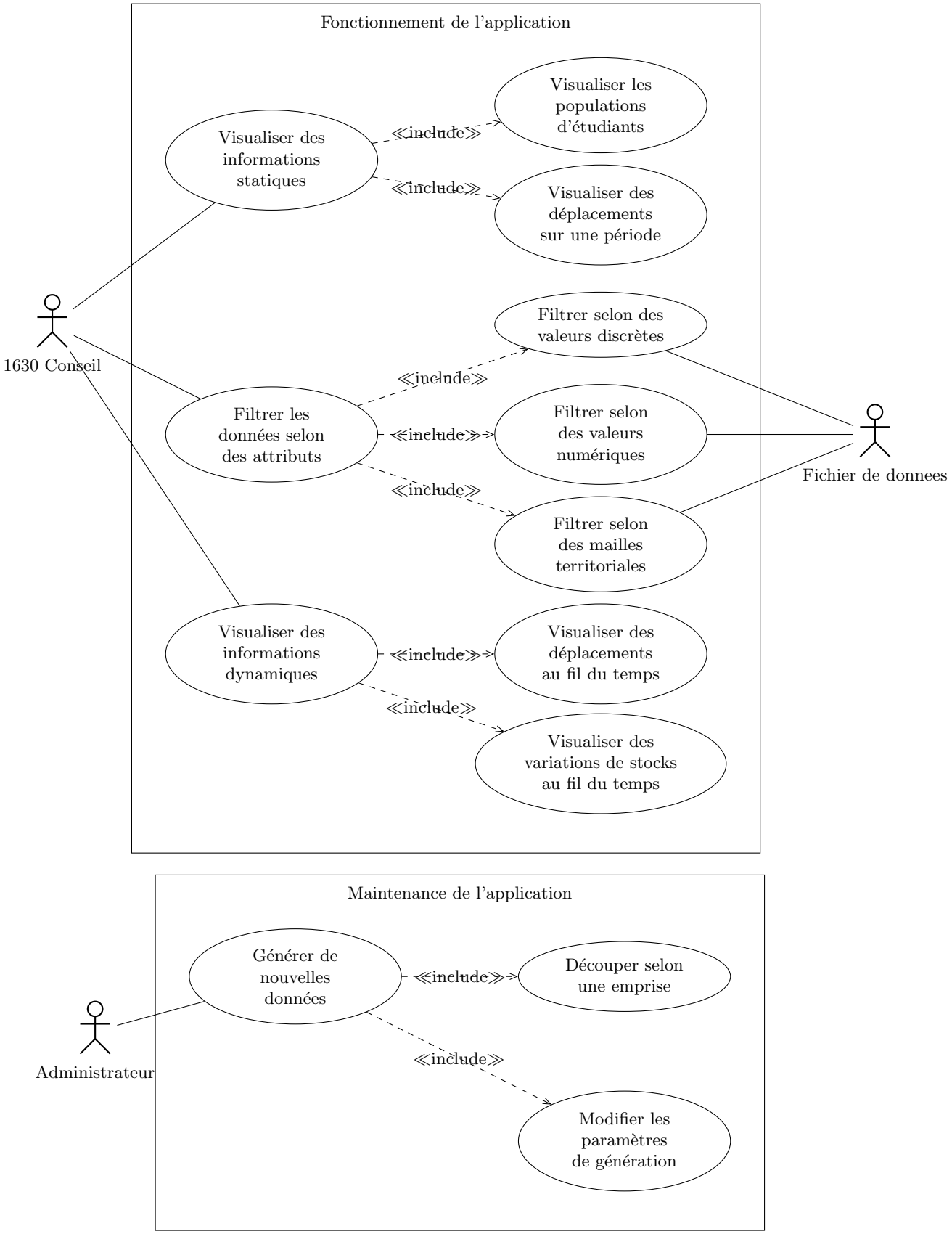

Figure 1: Diagrammes des cas d'utilisation

2.4 Aspects financiers et sociaux

Comme nous avons pu le voir dans le diagramme de Gantt d'origine, nous avions prévu 8 séances (1 séance correspondant à 1 demi-journée) d'analyse et 12 séances de développement. En réalité, toutes ces durées ne comprenaient que les séances officielles allouées à ce projet dans notre emploi du temps. Finalement, afin de nous assurer que nous puissions finir le projet dans les temps, nous avons décidé de consacrer plusieurs demi-journées de notre temps libre pour réaliser le projet. Au final, nous avons passé 56 demi-journées sur ce projet. Les 8 premières étant destinées à l'analyse et les 6 dernières aux présentations finales.

Le projet en soit est totalement gratuit. Cependant, si on suppose qu'il coûte 3,9 € de l'heure [2] pour chaque étudiant (paie minimale pour un stagiaire français), il coûterait 46,8 € par demi-journée pour tous les étudiants. Finalement, le projet complet coûterait 2620,8 €.

Concernant les aspects sociaux, chacun apporte ses qualités techniques de par le choix de ses précédents projets. Nous nous sommes naturellement séparés en 2 équipes pour la génération des données et pour leur visualisation. La phase d'analyse n'a pas posé de problème puisque nous pouvions nous voir physiquement. La phase de développement s'est retrouvée confondue avec la période du confinement, il a donc fallu utiliser de nouvelles techniques de communication que nous décrirons plus tard.

Même si aucune personne n'avait de rôle attitré initialement, il nous est apparu nécessaire de définir des rôles distincts. Ainsi, A.M. a été désigné pour assurer le rôle d'interlocuteur principal entre notre groupe et les commanditaires. Il organisait les réunions Teams avec les commanditaires et échangeait la plupart des messages avec eux. Y.J. a été désigné comme étant l'administrateur de notre site WEB et a été chargé de son déploiement sur la plate-forme Heroku. A.G. a eu pour rôle de s'occuper de la génération finale des données. Enfin, C.S. a assuré la création de la page accueil de l'application, ainsi que les différents liens inter-pages.

3 Analyse fonctionnelle

3.1 Introduction

Lors de la partie précédente, nous avons vu que nos commanditaires souhaitaient recevoir une application représentant les flux des étudiants entre les différents lieux qu'ils fréquentent. Ils souhaitaient également connaître la répartition des étudiants selon le type de lieu (domicile étudiant, lieu de travail...). Cette application doit être la plus esthétique et doit représenter des informations de la manière la plus intelligible possible.

L'objectif de cette partie est de décrire les grandes fonctionnalités que géreront notre application. Initialement, nous avions créé un diagramme de classe complexe lorsque nous pensions repartir de zéro pour coder le démonstrateur cartographique. En effet, nous avons finalement choisi d'utiliser un outil nous permettant de présenter à l'écran de larges jeux de données, ce qui correspondait parfaitement à nos attentes. Il s'agit de l'API Kepler.gl. Nous justifierons plus précisément les raisons qui nous ont poussé à nous pencher vers cette solution en cours de projet dans la partie suivante. La construction d'un diagramme de classes n'était donc plus approprié

3.2 Règles de génération des données

Pour la partie génération des données, une étape primordiale a été de définir les règles nous permettant de produire les différents attributs.

3.2.1 Informations générales des étudiants

Nous avons défini les premiers attributs selon des valeurs discrètes. Par exemple, il nous fallait décider le sexe de l'étudiant, s'il était boursier ou non, ou enfin s'il vivait en couple ou seul. Nous avons donc dû choisir des proportions d'étudiants au niveau national.

Nous avons supposé que les hommes et les femmes étaient répartis de manière homogène entre les différentes filières de l'enseignement supérieur. À partir d'un calcul réalisé à l'aide de sites internets présents en annexe [3] [4]. Le pourcentage de femmes est d'environ 54%. Voici les chiffres que nous utiliserons pour la génération :

- Femmes : 54%
- Boursiers : 38% [5]
- Mariés : 7%
- Étudiants avec un job : 46% [6]

3.2.2 Génération des adresses

Dans un second temps, il était nécessaire de générer les adresses des différents lieux fréquentés par les étudiants sous la forme d'un fichier geojson. Pour la génération d'un tel fichier, l'idée était de se limiter dans un premier temps aux étudiants qui sont dans des résidences universitaires dont le nombre est 366 000 puis d'étendre cette méthode à tout type de domicile. La création de ce fichier a commencé par la récupération d'un fichier json qui contient les résidences universitaires et les CROUS en France [7], après nous avons fait en sorte de tirer une information sur la capacité de chaque résidence à partir de la description associée en se servant des expressions régulières.

La deuxième étape consistait à tirer des informations du fichier json des établissements et universités, puis à en créer un nouveau qui va servir pour les implanter pour y attribuer des étudiants.

Maintenant que ces deux fichiers sont créés, nous sommes passés à la création du geojson des étudiants avec tout ses attributs sous les contraintes suivantes :
Chaque étudiant possède un identifiant unique généré à partir d'une bibliothèque uuid4. Puisque ces étudiants sont en résidences universitaires, nous leur avons attribué une adresse du domicile parental aléatoire en France ou outre-mer, une adresse pour un lieu d'étude et une adresse pour un emploi (s'il dispose d'un job) qui est générée dans un rayon de 7km maximum de son adresse de résidence.
D'autres étudiants ne résidant pas dans une résidence ont aussi étaient générés aux alentours de nos lieux d'étude.

Pour en savoir plus sur la génération des lieux, les figures 12 à 15 offrent des informations plus détaillées.

3.2.3 Détermination du revenu fiscal du foyer de rattachement

Un autre attribut important correspond au revenu fiscal du foyer de rattachement de l'étudiant. Il s'agit des revenus nets imposables gagnés par chaque ménage en une année.

Le site suivant [8] nous permet d'interpréter correctement le revenu fiscal du foyer de rattachement.

Nous allons dans un premier temps distinguer les boursiers des non boursiers. Nous allons supposer que le revenu fiscal moyen des foyers boursiers est de 17000 €. Pour les générer nous

allons utiliser une loi normale de moyenne 17000 € et d'écart-type 2000 €.

Dans le cas des non boursiers, nous partons de l'hypothèse que plus un étudiant part étudier loin du domicile parental, plus il y a des chances que le revenu fiscal du foyer de rattachement soit élevé. Nous allons supposer qu'en moyenne le revenu moyen fiscal des foyers familiaux qui se situent pile à côté de l'école est de 23000 € (soit un salaire à la limite inférieure de la classe moyenne). Au delà de 500 km, la distance se fait de moins en moins sentir sur le budget (les étudiants ont tendance à ne pas rentrer au domicile parental le week-end). La fonction qui associe le revenu fiscal moyen en fonction de la distance est ainsi une fonction exponentielle, puisque l'augmentation est forte au début, quasi linéaire, puis diminue, jusqu'à atteindre une valeur limite. Vous trouverez en annexe, la courbe du revenu fiscal moyen en fonction de la distance.

Nous n'affecterons pas le même revenu à tous les foyers se trouvant à une même distance de leur école car cela ne serait pas représentatif. À nouveau nous les générerons par une loi normale de moyenne la moyenne des revenus fiscaux pour la distance considérée et d'écart-type 4000 €. Cet écart-type permet de prendre en compte l'incertitude de nos hypothèses.

Enfin dans les deux cas, on a décidé de fixer une valeur minimale et maximale au revenu fiscal du foyer de rattachement, pour éviter d'obtenir des valeurs aberrantes (voir annexes pour plus de précisions).

3.2.4 Emploi du temps

Pour générer les emplois du temps, nous nous sommes basés également sur des hypothèses fortes. Vous trouverez en annexe des graphiques décrivant avec précision les différentes affectations des emplois du temps.

Pour les jours de la semaine, nous avons considéré que tous les étudiants se rendaient sur le lieu d'étude. Toutes les demi-heures, une part des étudiants se rend sur son lieu d'étude et ce de 8 h à 12 h 30. Les étudiants travaillent au moins 5 heures et quittent leur lieu d'étude au plus tard à 20 h. Un chiffre aléatoire a été ajouté afin que les étudiants n'arrivent pas sur un lieu en même temps mais peuvent aussi arriver en avance ou un peu en retard.

Parmi ceux qui réalisent un job étudiant, une partie travaille le soir en semaine et l'autre le week-end pendant la journée. Un étudiant peut uniquement travailler le soir si il est possible pour lui de se rendre sur son lieu de travail avant 20h. Il peut travailler le week-end si il ne rentre pas chez ses parents.

Le temps de trajet est calculé lui en fonction de la distance séparant les différents lieux fréquentés. Plus la distance est grande, plus le moyen de transport utilisé permet à l'étudiant de se déplacer rapidement. Enfin, pour simuler les aléas des transports, nous multiplions par le quotient (distance/vitesse) par un facteur aléatoire compris entre 0,8 et 1,2.

Pour en apprendre davantage sur la façon dont sont générés les trajets, les figures 16 à 19 offrent plus de plus amples informations.

3.3 Différents choix concernant la visualisation

3.3.1 Rédaction de scénarios

Afin de nous rendre mieux compte de la mise en contexte des données, nos commanditaires nous ont fait écrire plusieurs scénarios. Nous avons donc écrit quatre scénarios, dans lesquels nous avions pour but de raconter une histoire. Le premier suit un étudiant étranger qui souhaite faire des études de commerce sur Paris, le deuxième suit une fratrie qui réalise ses études dans

plusieurs zones de la France, le troisième décrit un grand groupe d'écoles de commerce cherchant à implémentant une nouvelle école dans un emplacement optimal et enfin le dernier et enfin le dernier se concentre sur un investisseur qui souhaite investir dans les logements étudiants.

Finalement, nos commanditaires nous ont envoyé leur propre scénario découpé en quatre parties. Ce scénario correspond aux informations qu'ils souhaitent montrer devant les deux ministères. Pour chaque sous-partie, nous avons choisi la représentation la plus adaptée. Nous allons expliquer ces choix dans le paragraphe suivant.

3.3.2 Les différentes représentations proposées par Kepler.gl

L'outil cartographique Kepler.gl a pour avantage de proposer différents types de flux et de stocks. Représenter des stocks peut être intéressant lorsque l'on souhaite visualiser la localisation de tous les logements étudiants ou tous les lieux d'étude. Nous avons identifié quatre représentations qui sortaient du lot :

- Tout d'abord, les arcs représentent des flux statiques, c'est-à-dire un lien fixe entre deux lieux. Cette représentation était parfaitement adaptée pour montrer le lien entre le domicile parental et le lieu d'étude des étudiants. Ce lien était d'ailleurs l'enjeu de la première partie du scénario.

Figure 2: Trajets entre les domiciles parentaux (côté bleu) et les lieux d'étude (côté rose)

- Les trips représentent au contraire des flux dynamiques. Un trips correspond à une ligne plus ou moins fine qui se dépasse entre deux points et à deux dates définies. Ce mode de représentation est parfaitement adapté pour montrer les déplacements des étudiants au cours de la journée entre les différents lieux qu'ils fréquentent. Il s'agit de l'enjeu de la troisième partie du scénario.

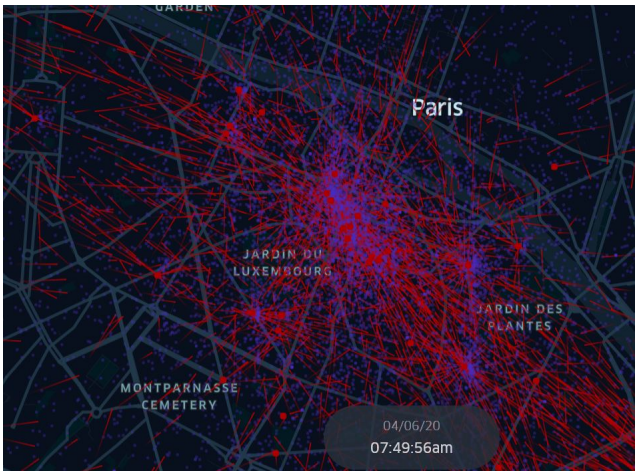

Figure 3: Déplacements des étudiants (lignes rouges) entre leur domicile étudiant (points violets) et leur lieu d'étude (points rouges)

- Les points et les hexbins représentent bien des stocks dans un contexte statique comme dynamique. Un hexbin échantillonne le plan en hexagones de tailles égales et compte le nombre de points dans cette surface plus il y a de points plus la colonne est haute. Dans un contexte statique, on peut afficher tous les étudiants dans leur lieux d'études indépendemment de la période de la journée. (enjeu de la deuxième partie du scénario). Dans un contexte dynamique, on affiche à l'écran la position des étudiants au fil du temps. (enjeu de la dernière partie du scénario)

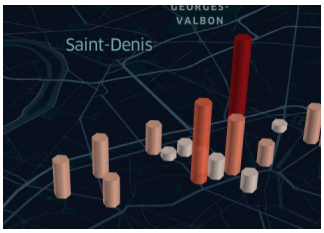

Figure 4: Répartition des étudiants selon leur lieu d'étude

4 Étude technique : Choix des logiciels et langages

4.1 Outils concernant la génération des fichiers de données

Afin de générer nos fichiers de données, nous avons choisi d'utiliser Python. Comme nous générons nos données à l'aide de statistiques, Python nous a semblé être une bonne solution pour générer des attributs sous condition. De plus, ce langage possède de nombreux modules permettant par exemple d'attribuer des adresses réelles sous contraintes. Nous placerons ensuite nos fichiers de données dans les pages WEB de notre site.

La partie applicative est cependant gérée par Nodejs vu qu'il offre des modules npm permettant la manipulation rapide des grands flux de JSON qui regroupent les statistiques. Les données utilisées sont en forme de fichiers .json téléchargées à partir de l'API OpenData du Ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation [9].

Ces données sont ensuite filtrées sous leur forme actuelle en Nodejs et ensuite, avant le peuplement de la base de données, certaines fonctions Python sont appliquées pour en générer d'autres attributs spécifiés dans le cahier de charges.

Outre les bibliothèques offertes sur npm, le choix de Nodejs se justifie aussi par la gestion asynchrone native des "Threads" par JavaScript. C'est-à-dire, les fonctionnalités ne sont pas bloquantes et cela permettra une manipulation plus souple de la grande quantité de données sans aucune configuration supplémentaire. Ce principe est brièvement expliqué à travers les photos ci-dessous:

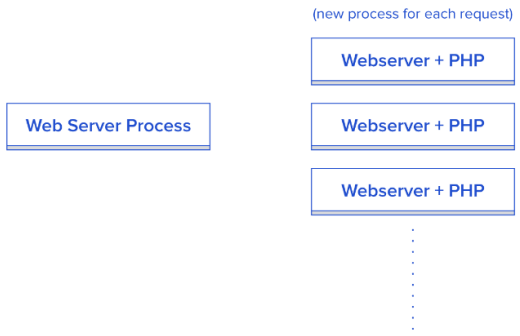

Figure 5: Gestion des fils avec PHP

PHP est contraignant quand il s'agit d'une multitude de requêtes simultanées qui consomment énormément de ressources ; chose qui met en péril notre serveur qui pour chaque utilisateur est obligé d'afficher une carte personnalisable et alors une dizaine de requêtes à exécuter.

Figure 6: Gestion des fils avec Java

Quant à Java, la possibilité d'activer le non-blocage des fonctionnalités existe mais demande une configuration à part et est rarement utilisée par des serveurs.

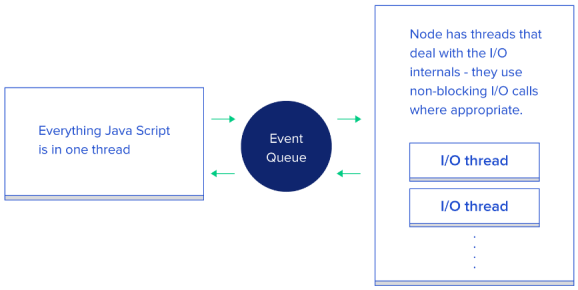

Figure 7: Gestion des fils avec JS

En ce qui concerne Nodejs, un appel exhaustif de données ne bloquera pas l'appel d'autres fonctions en parallèle chose qui convient le mieux à nos besoins.

Brièvement et plus clairement, vu que le site sera hébergé sur un serveur distant et sachant que le nombre de requêtes coté serveur est grand, on a choisi de travailler la totalité de cette partie avec du javascript sous le framework de node pour minimiser les blocages entre les appels des fonctions.

Pour plus de détail, vous pourrez retrouver le lien dans notre bibliographie [10].

4.2 Outils concernant la visualisation dynamique des données

Pour réaliser la deuxième partie du projet, nous avons décidé d'utiliser des techniques de programmation WEB en prenant en compte les modèles de cartes dynamiques proposés par nos commanditaires. Nous avons choisi ces techniques plutôt qu'un SIG car il sera plus facile de créer notre propre charte graphique. D'autre part, le partage de l'application sera plus aisé (avec le commanditaire) puisque qu'il n'y aura pas d'installation nécessaire à l'utilisation du produit.

3 librairies et API nous ont semblé adaptées à nos besoins : Leafletjs, l'API de Google Maps et Mapbox.

- L'API de Google Maps contient de nombreuses sous-fonctionnalités (géolocalisation, trafic routier, calcul d'itinéraire...) à la carte. La plupart des fonctionnalités sont séparées l'une de l'autre et sont payantes à partir d'un certain nombre de téléchargements. Le principal défaut de l'API de Google Maps est son instabilité au niveau des conditions d'utilisation qui peuvent changer à tout moment. L'API de Google ne dispose pas de données en offline, ce qui constitue une faiblesse.

- La librairie Leafletjs est totalement gratuite et simple d'utilisation. Contrairement à l'API Google Maps, elle est en OpenSource et ne nécessite pas de clé d'utilisation. Elle est particulièrement adaptée lorsque l'on souhaite créer une application de A à Z, mais elle est plus limitée dans la visualisation dynamique que l'API qui va suivre.

- L'API Mapbox combine les avantages des deux solutions précédentes. Mapbox dispose d'outils et de cartes orientés sur l'esthétisme. Mapbox est donc meilleure pour les interactions dynamiques et au niveau visuel, car plus customisable que Google Maps. Par ailleurs, des exemples d'applications attendues par nos commanditaires ont été réalisées par Mapbox, c'est pourquoi nous avons décidé de nous pencher sur cette API. Mapbox est Opensource. L'application devient payante au bout de 25000 requêtes par mois, ce qui ne devrait pas être notre cas.

Nous avions initialement choisi l'API MapboxGL pour réaliser notre application WEB qui combinait plusieurs aspects que nous recherchions. MapboxGL est une alternative moderne de MapboxJS. MapboxJS n'a plus de mise à jour d'après le site officiel [11]. L'intérêt principal

de MapboxGL est surtout d'intégrer les fonctionnalités proposées par l'API WebGL. En effet, WebGL permet d'afficher, de créer et de gérer dynamiquement des éléments graphiques complexes en 3D dans la fenêtre du navigateur web d'un client. Il nous était donc possible de représenter des flux ou des stocks en 3D. Ainsi, MapboxGL peut afficher des informations dynamiques en 3D. Enfin, MapboxGL est encore en développement avec l'ajout régulier de fonctionnalités et la correction de bugs.

L'inconvénient de Mapbox était que nous étions obligé de repartir à zéro pour coder le visualiseur cartographique. Ce codage était risqué dans la mesure où l'on risquait d'obtenir un résultat qui ne soit pas à la hauteur des espérances de nos commanditaires. De plus, nos commanditaires semblaient intéressés par les rendus graphiques proposés par Kepler.gl [12] qui permettait justement de visualiser les informations suivant différentes représentations prédéfinies. Kepler.gl permettait de réaliser des représentations 3D à partir de fichiers geojson ou CSV. À partir de cette décision, l'équipe de génération des données a décidé de ne plus produire de base de données, mais des fichiers CSV et geojson directement. L'équipe de visualisation des données avait pour objectif de guider l'autre équipe dans la génération de ces données, mais aussi d'utiliser de manière optimale, l'outil Kepler.gl.

4.3 Architecture de l'application

L'application suit un formalisme n-tiers et regroupe une partie client, une partie serveur d'application et une partie données. Ces données sont sous la forme de fichiers de type Geojson ou Csv et sont générés grâce à des scripts python. Le schéma ci-dessous décrit cette architecture ainsi que les technologies utilisées:

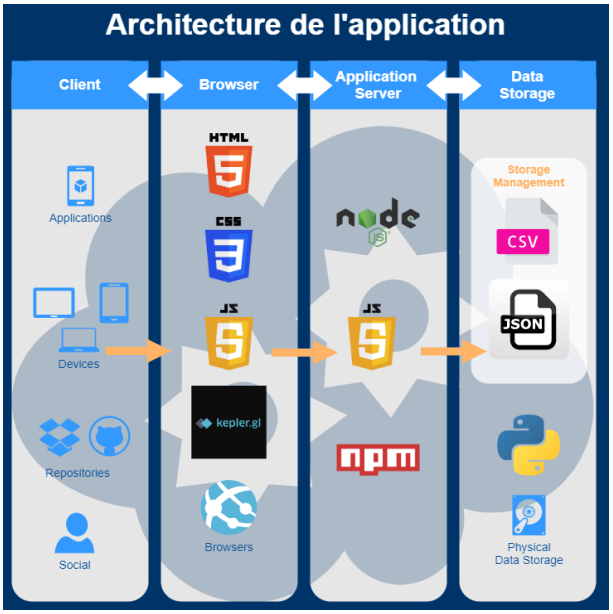

Figure 8: Architecture de l'application (figure réalisée par nos soins)

4.4 Déploiement de l'application

Afin de centraliser les travaux, les commanditaires nous ont proposé la plate-forme Heroku comme serveur de production qu'on a lié après à notre répertoire privé de GitHub. Cette plateforme est configurable pour assurer des détections automatiques des changements au niveau du répertoire ainsi qu'un tracking en mode Logs des erreur et warnings dans le code. Vous trouverez le lien de notre site WEB en annexe [13].

En ce qui concerne le lien et les builds automatiques, l'interface ressemble à:

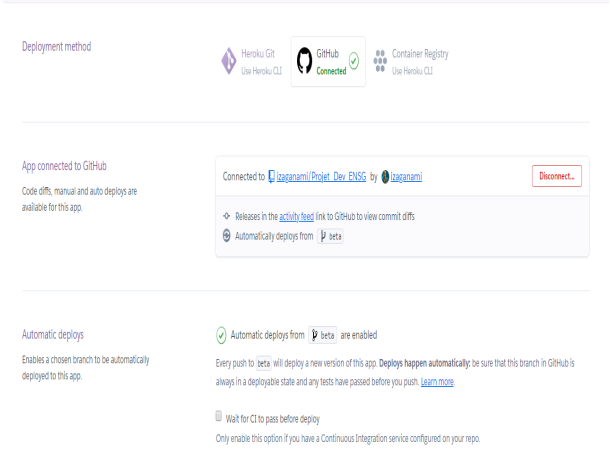

Figure 9: Déploiement en production sous Heroku(Capture d'écran)

Pour la partie debugging et le suivi des erreurs, on dispose de l'interface suivante:

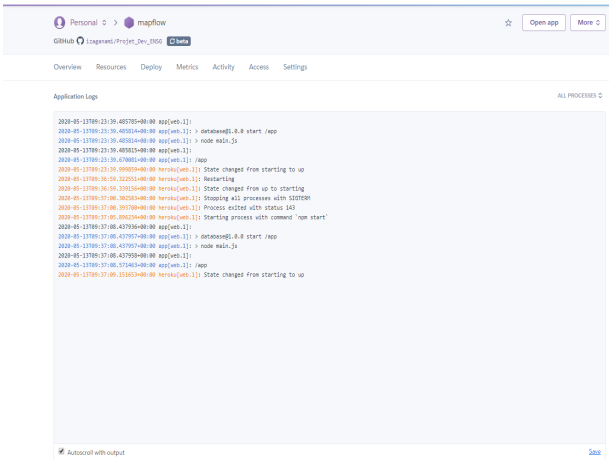

Figure 10: Les logs sous Heroku(Capture d'écran)

5 Réalisation et suivi du projet

Au début du projet, nous avons réalisé nos outils de suivi de projet directement sur LaTeX, finalement, il était plus facile de les modifier directement un fichier Excell sur un Drive. Le planning est en effet devenu colossal au fil des semaines, à mesure que la visibilité s'améliorait. Nous vous en donnons le lien via la bibliographie [14]. La première page contient l'ensemble des livrables commandés (et ceux que nous avons abandonnés) par les commanditaires et l'ENSG accompagnés d'un pourcentage de progression, les deux suivantes contiennent notre planning

initial et notre planning final, et les deux dernières contiennent nos matrices des risques initiales et finales.

5.1 Risques

Durant notre projet, nous avons été confrontés à des risques techniques comme à des risques humains. Nous allons nous baser sur la matrice de risque présente sur le lien que nous vous avons indiqué ci-dessus.

Tout d'abord, le premier risque que nous avions identifié avait été une mauvaise communication au sein du groupe. En effet, ceci aurait pu arriver dans le cas, où chaque personne du groupe n'informait pas correctement ses collègues concernant les avancées qu'il a menées pendant la séance. Finalement, cela ne c'est pas trop produit en pratique, puisque nous notions régulièrement nos avancées dans un Framapad.

Un autre risque de notre projet était une demande de changement d'une fonctionnalité au cours du projet. Finalement, ceci n'est pas arrivé puisque nos commanditaires ont toujours été clairs dans leurs lignes de conduite. À chaque rendez-vous, ils faisaient un récapitulatif de ce qui avait été fait et demandaient des rendus intermédiaires. Ces rendus nous ont d'ailleurs bien aidés à baliser le projet.

Concernant, l'aspect technique, nous avions considéré que la génération puis le téléchargement des données de la base de données à l'application présentaient une complexité dont l'impact était inconnu à l'époque de la rédaction du premier rapport d'analyse. En réalité, ce n'est pas la téléchargement des données qui pesait lourd mais plutôt l'importation des fichiers de données dans l'outil Kepler.gl. Finalement, cet aspect a été bloquant car l'outil Kepler.gl ne nous a pas permis de représenter plus de 200 000 étudiants. En effet, s'il y avait trop de données l'affichage des données devenait très haché voire impossible.

D'un point de vue organisationnel, nous étions conscients que la génération optimale des données pouvait prendre du retard. Ceci n'a pas eu un impact trop fort finalement dans la mesure où l'utilisation de l'outil Kepler.gl a permis de nous faire gagner un temps précieux.

Au début, nous avions également envisagé la possibilité d'accumuler du retard pour la création du démonstrateur puisque nous avions prévu de coder nous même l'application. Ce risque a disparu lorsque nous avons adopté l'outil Kepler.gl

Les coupures de connexion Wifi ou la disposition limitée de données 4G a parfois posé problème pour notre groupe lorsque nous souhaitions réalisé des réunions avec nos commanditaires. Il arrivait parfois que certains membres du groupes ne puissent pas être présents.

Le confinement en lui-même a aussi un défi majeur de notre projet dans la mesure où nous avons rapidement dû maîtriser les différents outils de développement et de gestion de projet en ligne. Finalement, nous pensons avoir relevé le défi. Cette situation nous aura permis de nous adapter.

Pendant le projet, il fallait bien veiller à donner les tâches les plus précises à chaque personne afin d'éviter toute duplication du code qui pourrait non seulement rendre le code illisible mais aussi occasionner des erreurs. Ce risque n'a pas été considéré initialement mais il est primordial d'y prendre garde.

Le dernier risque qui peut sembler universel est la non anticipation ou la sous-estimation d'un risque. En effet, il est toujours possible d'omettre un risque comme le prouve le fait que nous ayons sous-estimé le poids des données.

5.2 Outils de communication

Au début du projet, nos commanditaires organisaient des rendez-vous téléphoniques hebdomadaires afin qu'ils puissent s'informer de l'avancement de nos développements. Nous avons eu l'honneur de les rencontrer physiquement au début du projet. Malheureusement, cela n'a plus été possible à partir du début du confinement. Les rendez-vous téléphoniques devenaient également plus compliqués. C'est pourquoi, nous avons suggéré à nos commanditaires de plutôt réaliser des réunions Teams à la place. Ainsi, cela nous a permis d'utiliser le partage d'écran qui s'est avéré utile pour présenter nos résultats au fur et à mesure de nos développements.

Pour communiquer entre les membres du groupe, nous avons utilisé différents supports. Tout d'abord, nous avons naturellement utilisé l'outil Teams pour communiquer entre nous. Nous faisions un rendez-vous lors de la plupart des séances. Ensuite, nous avons créé une page Framapad afin de recenser les avancées hebdomadaires de notre projet. Cette page nous permettra donc de savoir si nos avancées sont en cohérence ou non avec notre planning prévisionnel. Elle nous servira également, si l'on souhaite échanger des informations capitales, pour écrire les comptes-rendus de nos entrevues avec nos commanditaires ou encore partager des liens Internet intéressants. Nous avons utilisé la plateforme GitHub pour déposer notre code et pour le gérer. Nous avons créé une branche func sur laquelle l'équipe "généralisation des données déposait son code". La branche dev, elle était plutôt utilisée par l'équipe "visualisation des données". Enfin, la branche bêta a été créée pour hébergée l'application finale.

5.3 Subdivision des tâches

Nous avons défini quatre grandes périodes qui jalonneront notre projet : l'analyse, le développement de la solution, la préparation des rendus finaux et les soutenances. Pour réaliser notre projet développement, nous disposons de 28 séances sur notre emploi du temps. Les 8 premières ont été dédiées à la phase d'analyse. À partir de ce moment, nous nous sommes rapidement rendus compte que l'application nécessitait plus de temps que les horaires prédéfinis. C'est pourquoi, nous avons décidé d'ajouter 30 séances de travail sur notre temps libre (vacances, jours fériés...), à partir du moment où nous avions fini la phase d'analyse.

Les quatre membres du groupe ont contribué à la phase d'analyse. Les différentes grandes tâches de cette phase sont la reformulation du besoin gérée par A.G. et A.M., la recherche de statistiques sur les logements étudiants et sur les étudiants gérée par C.S. et Y.J., la planification du travail gérée par A.G. et A.M., l'analyse informatique de l'application gérée par A.M. et C.S. (diagramme de classes initial, de cas d'utilisations...), la réflexion sur la manière de générer la base de données assurée par Y.J. et A.G. et enfin la réalisation d'un benchmark des différentes méthodes de visualisation assurée par A.M. et C.S.. La rédaction du rapport d'analyse sera réalisée sur les huit premières séances.

Puisque notre projet est composé de deux tâches bien distinctes, la création de la base de données et la création du site web, nous avons décidé de scinder l'équipe en deux groupes de 2. A.G. et Y.J. se sont chargés de la création des données, tandis que C.S. et A.M. se sont occupés de leur visualisation.

- A.G. et Y.J. ont dû créer les données nécessaires à l'équipe visualisation. L'objectif était qu'ils aient terminé suffisamment tôt pour que l'autre équipe puisse utiliser ces données dans la partie visualisation. Cette étape a finalement pris plus de temps que prévu, dans la mesure où l'objectif final de cette équipe avait évolué au fil du temps. À l'origine, leur mission était seulement de créer une base de données. Lorsque cette idée a été abandonnée, ils ont dû s'adapter en générant des données CSV et geojson directement ce qui s'est avéré chronophage. Finalement, cette perte de temps a été compensée par le fait que le choix de l'utilisation de Kepler.gl a fait gagné du temps par ailleurs.
- C.S. et A.M. se sont formés pendant les 2 premières séances de développement à l'utilisation

de l'API Mapbox, afin d'exploiter au maximum le potentiel de cette dernière. Parallèlement, et jusqu'à la fin de la phase de développement, ils ont dû définir et redéfinir l'esthétisme de l'application. À partir de la semaine du $1^{er} avril$, ils ont commencé à se former à Kepler.gl pour utiliser tous les outils disponible de manière optimale. Ce changement a été l'occasion de remettre en cause le planning initial. En effet, nous avions décidé d'utiliser un outil cartographique déjà codé plutôt que de repartir à zéro. Finalement, le rôle de l'équipe visualisation a été de guider l'autre équipe pour la génération des bons types de fichiers, mais aussi de créer une page d'accueil esthétique dans l'esprit des designs proposés par Kepler.gl. C'est d'ailleurs C.S. qui s'est principalement occupé de la page d'accueil. Parallèlement, A.M. a momentanément épauler l'équipe génération en développant la fonction générant le revenu fiscal du foyer de rattachement.

D'après nos commanditaires, la génération des données aurait dû prendre un peu moins de temps que la visualisation. C'est finalement le contraire qui s'est produit suite au choix d'utiliser l'outil Kepler.gl.

Dans le rapport d'analyse initial nous avions omis de prévoir des phases de recette. Nous avons décidé d'en intégrer dans notre planning suite à notre premier rendez-vous avec Véronique Pereira. Ceci comprend non seulement les séances ou nous présentons nos avancées aux commanditaires, mais aussi des phases où un membre d'une équipe vérifie le code ou le travail d'une autre équipe. Cette vérification doit être régulière et ponctuelle, c'est pourquoi cette action n'est pas continue dans notre GANTT.

Une autre action importante est la mise à jour des outils de gestion de projet globalement. Ceci comprend la mise à jour hebdomadaire du planning et la mise à jour régulière de l'avancement des livrables et de la matrice des risques. Il est en effet important de bien piloter son projet lorsque des événements majeurs surviennent pendant le développement. Ces tâches ont été réalisées par l'ensemble de l'équipe.

À l'instar de la phase d'analyse, les phases de préparation des rendus finaux puis de préparation des présentations se sont fait avec tous les membres de l'équipe.

5.4 Explications sur le titre et le logo

Le titre du projet "MapFlow" est la combinaison de carte et flux en anglais. Ces deux termes sont très importants car ils représentent tout l'enjeu de notre projet: visualiser des flux sur une carte ayant un aspect attrayant.
Concernant le logo, nous avons voulu qu'il représente bien l'activité de notre projet. Il est divisé en 2 parties. En haut se trouve un livre symbolisant les étudiants. Dans la partie basse, on peut voir un marqueur représentant les lieux fréquentés par l'étudiant auquel on a ajouté un itinéraire en pointillé pour montrer l'importance des déplacements. Nous avons choisi des couleurs plutôt ternes pour ne pas qu'elles soient trop agressives et ainsi pouvoir en utiliser plusieurs et faire ressortir les 2 éléments les plus importants.

6 Conclusion

Le projet "MapFlow" a pour objectif de créer une application pour promouvoir le logement étudiant français. Nos commanditaires Bertrand MOINEAU et Charles-Henri ARNOULD, respectivement Directeur Général et associé du cabinet de conseil 1630, souhaitent une application montrant les flux des étudiants au cours du temps entre les différents lieux qu'ils fréquentent. Ils souhaitent présenter cette application au Ministère de la Cohésion des territoires et des Relations avec les collectivités territoriales et au Ministère de l'Enseignement supérieur et de la Recherche et ont insisté sur un graphisme attrayant de l'application. Au final, nous sommes satisfaits des résultats obtenus qui sont conformes aux attentes de nos commanditaires. Nous sommes également satisfaits d'avoir acquis de compétences multiples, que ce soit pour le développement informatique ou la gestion de projet. Enfin, la situation de confinement nous aura permis de nous confronter à des situations difficiles et inédites.

Glossaire

A.G. Arthur GENET. 16

A.M. Alfred MENGIN. 16, 17

C.S. Christophe SAMBOUN. 16, 17

revenu fiscal du foyer de rattachement Le revenu fiscal du foyer de rattachement des étudiants correspond aux revenus nets imposables gagnés par chaque ménage en une année.. 7

scénario Un scénario de visualisation correspond à la mise en contexte des données, afin de montrer un phénomène ou une tendance. Avec les données dont on dispose par exemple, on peut montrer que les étudiants ont tendance à étudier près de chez leurs parents.. 2

Y.J. Younes JALLOUF. 16

Références bibliographiques

[1] Site web de 1630 conseil. http://www.1630conseil.com/.

[2] Ministère chargé du travail. Gratification minimale d'un stagiaire. https://www.service-public.fr/professionnels-entreprises/vosdroits/F32131.

[3] Ministère de l'enseignement supérieur et de la recherche. Chiffres clés de la parité dans l'enseignement supérieur et la recherche. 2012. https://cache.media.enseignementsup-recherche.gouv.fr/file/Charte_egalite_femmes_hommes/90/6/Chiffres_parite_couv_vdef_239906.pdf.

[4] Ministère de l'enseignement supérieur et de la recherche. Les effectifs dans l'enseignement supérieur en 2018-2019. 2019. https://cache.media.enseignementsup-recherche.gouv.fr/file/2019/28/7/Synthese_effectifs_etudiants_2018-2019_1163287.pdf.

[5] Séverin Graveleau. Chiffres clés de la parité dans l'enseignement supérieur et la recherche. 2017. https://www.lemonde.fr/campus/article/2017/10/05/enseignement-superieur-38-des-etudiants-sont-boursiers_5196726_4401467.html.

[6] Le Monde. 46 % des étudiants travaillent pendant leurs études. 2017. https://www.lemonde.fr/campus/article/2017/05/22/46-des-etudiants-travaillent-pendant-leurs-etudes_5131551_4401467.html.

[7] Ministère de l'enseignement supérieur et de la recherche. Publications statistiques sur l'enseignement supérieur et la recherche. https://data.enseignementsup-recherche.gouv.fr/explore/dataset/fr-esr-publications-statistiques/api/.

[8] lafinancepourtous. Niveau et composition des revenus moyens en france. 2019. https://www.lafinancepourtous.com/decryptages/finance-perso/revenus/niveau-et-composition-des-revenus-moyens-en-france/.

[9] de la recherche et de l'innovation Ministère de l'enseignement supérieur. Principaux établissements d'enseignement supérieur, 2016. https://data.enseignementsup-recherche.gouv.fr/explore/dataset/fr-esr-principaux-etablissements-enseignement-superieur/api/?disjunctive.type_d_etablissement&sort=uo_lib.

[10] Brad Peabody. Server-side i/o performance: Node vs. php vs. java vs. go, 2016. https://www.toptal.com/back-end/server-side-io-performance-node-php-java-go.

[11] Page décrivant l'api mapbox.js. https://docs.mapbox.com/help/glossary/mapbox-js/.

[12] Site web de kepler.gl. https://kepler.gl/.

[13] Alfred MENGIN Christophe SAMBOUN Younes JALLOUF, Arthur GENET. Projet mapflow. https://mapflow.herokuapp.com/.

[14] Notre fichier excell contenant les outils de gestion de projet. https://ensgeu-my.sharepoint.com/:x:/g/personal/alfred_mengin_ensg_eu/ET3qutuYJvdAoIpTGiJ198MB2baoiJmS5Mi3uy7sLWLEig?e=w15L6n.

Annexes

Revenu fiscal du foyer de rattachement

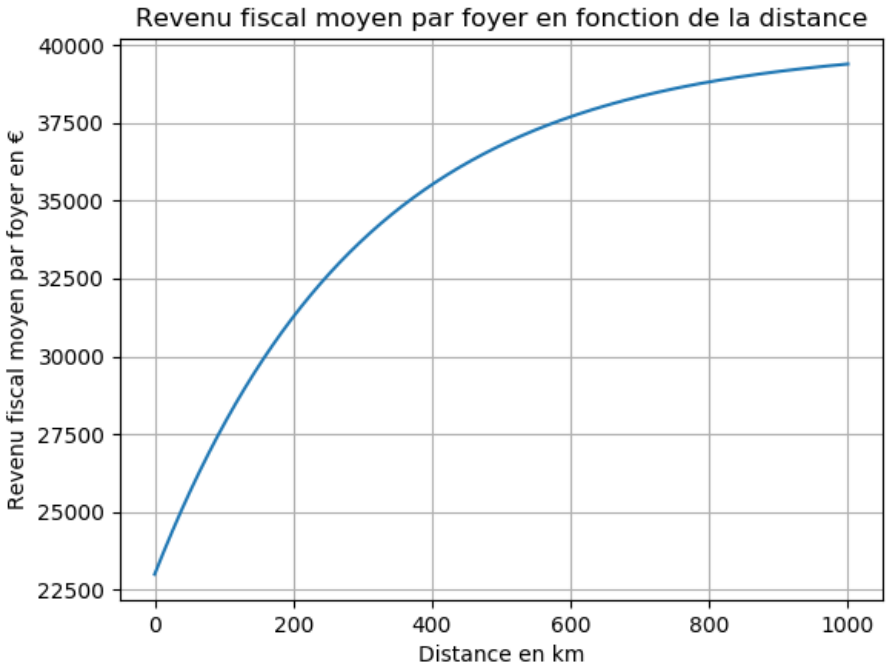

Figure 11: Revenu fiscal moyen par foyer en fonction de la distance entre l'école et le domicile familial

La génération des lieux fréquentés par l'étudiant

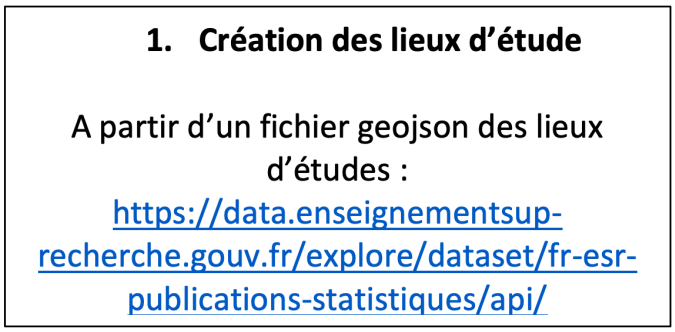

Figure 12: La création des lieux d'étude

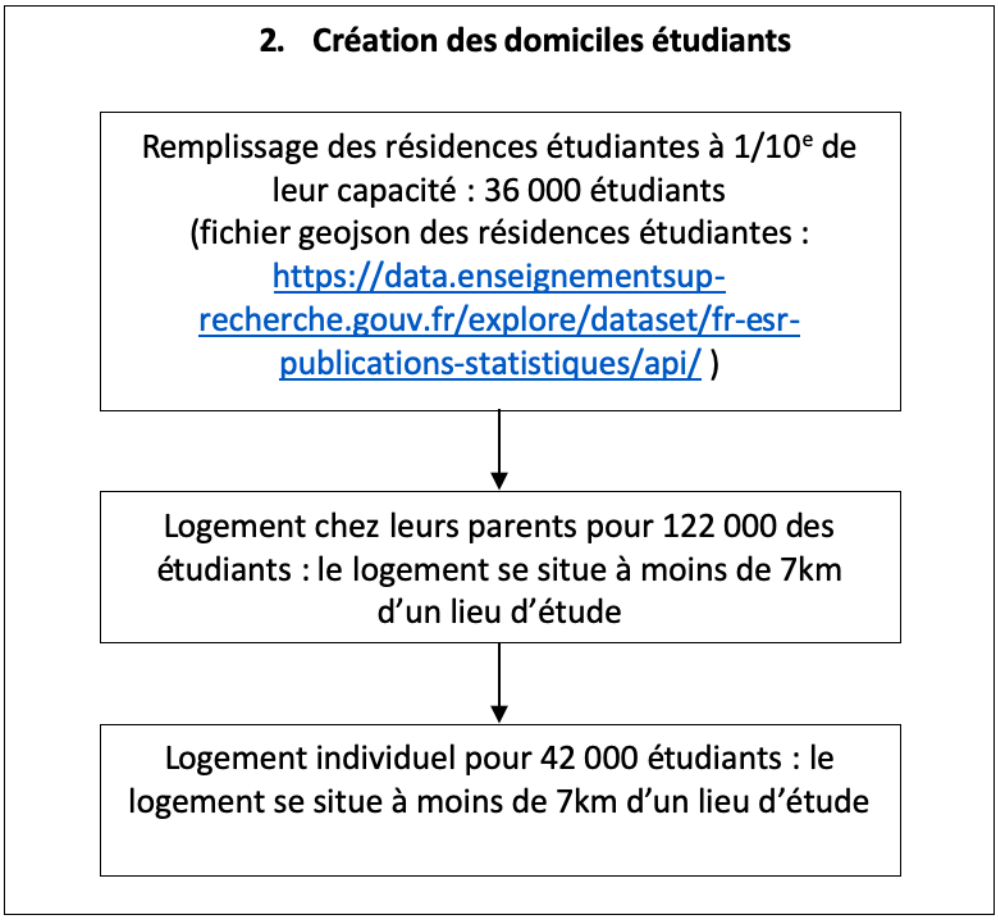

Figure 13: La création des domiciles où logent les étudiants

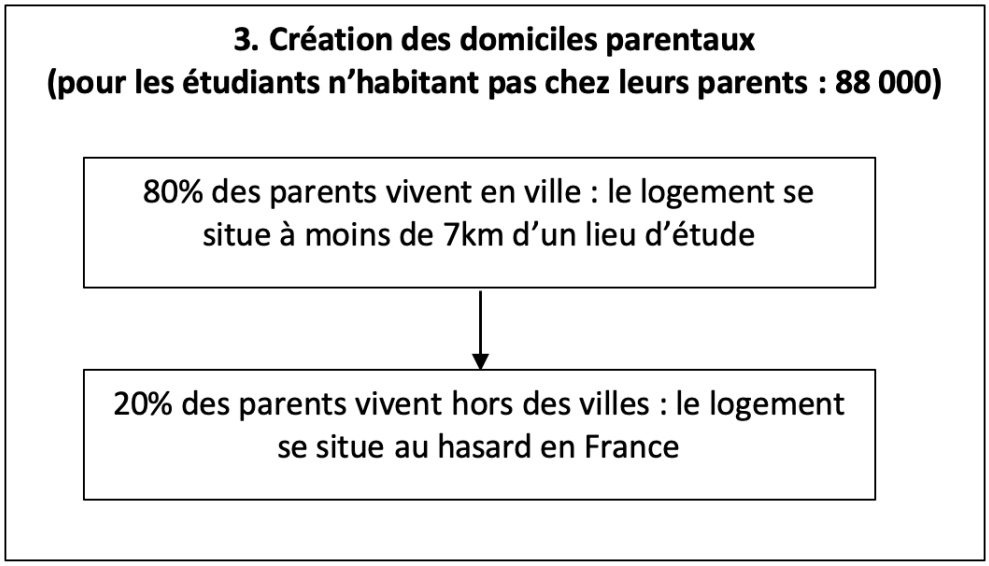

Figure 14: La création des domiciles parentaux

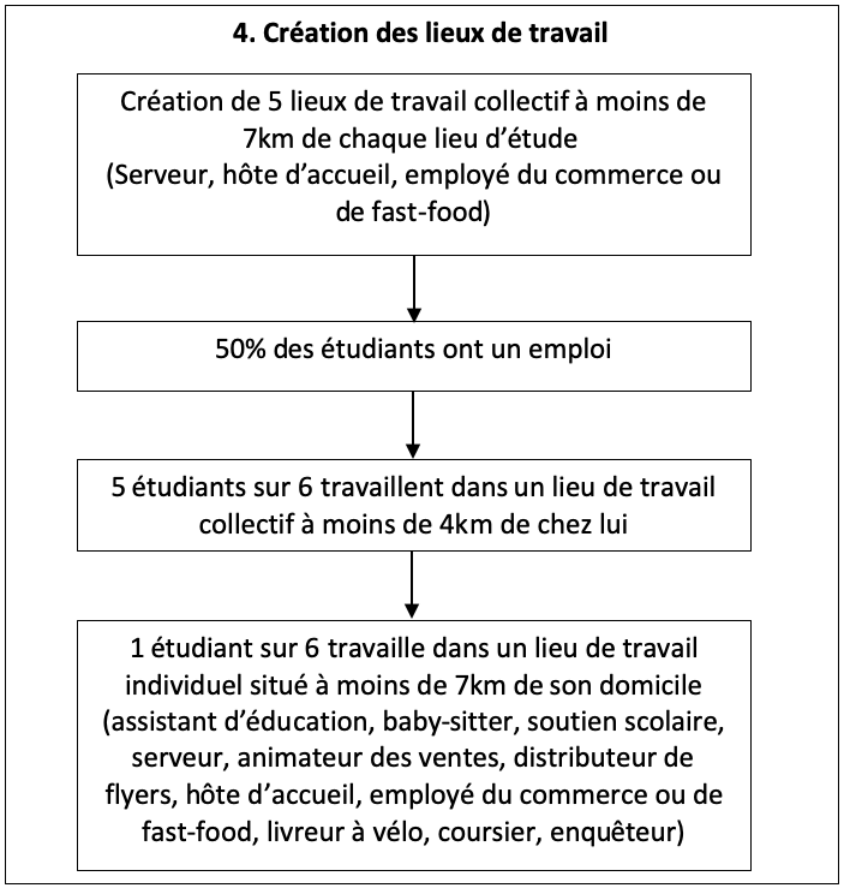

Figure 15: La création des lieux de travail

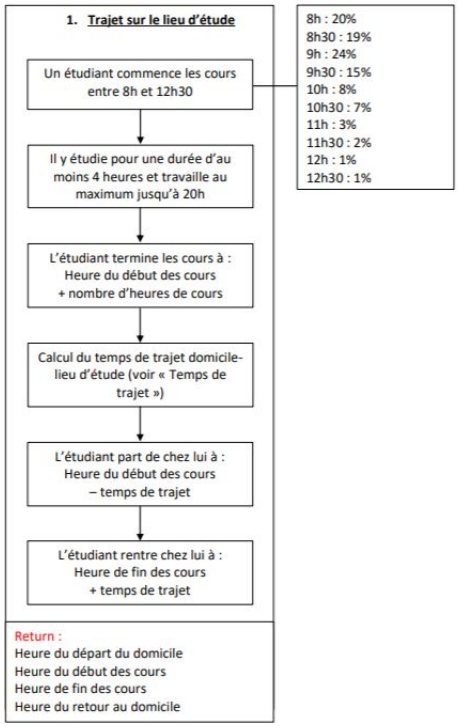

Figure 16: Les déplacements des étudiants entre leur logement et leur lieu d'étude

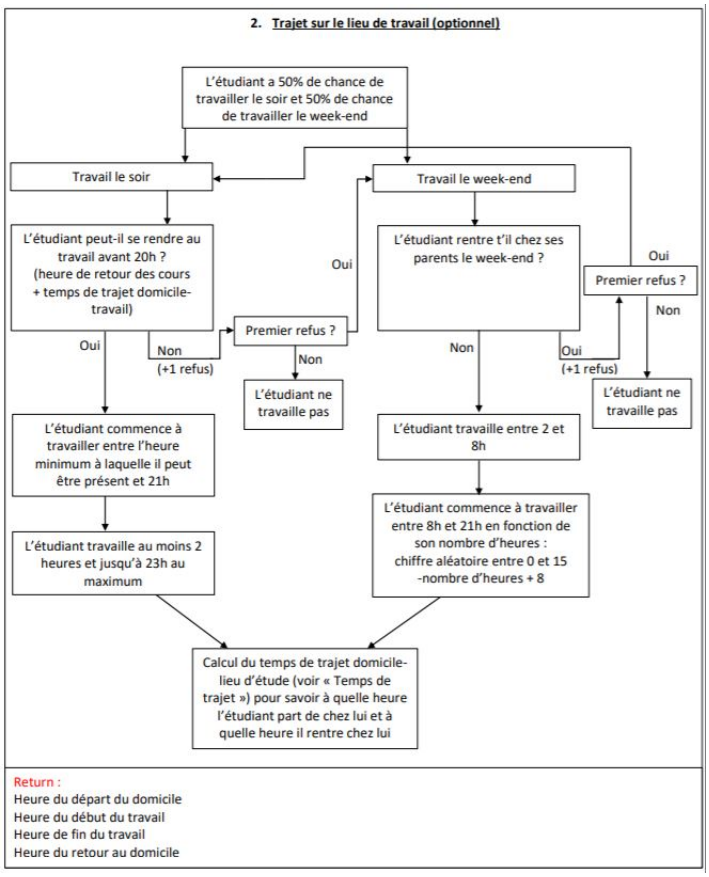

Figure 17: Les déplacements des étudiants entre leur logement et le domicile parental

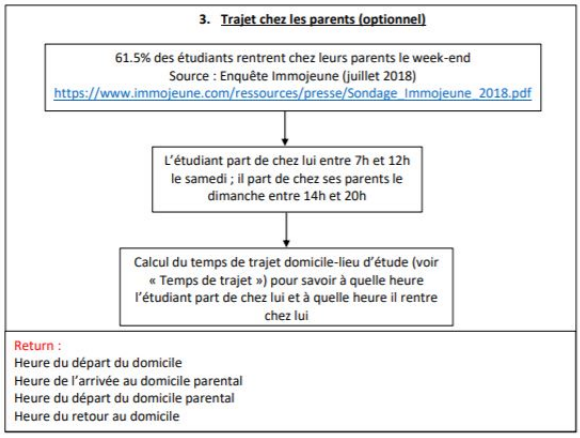

Figure 18: Les déplacements des étudiants entre leur logement et le domicile parental

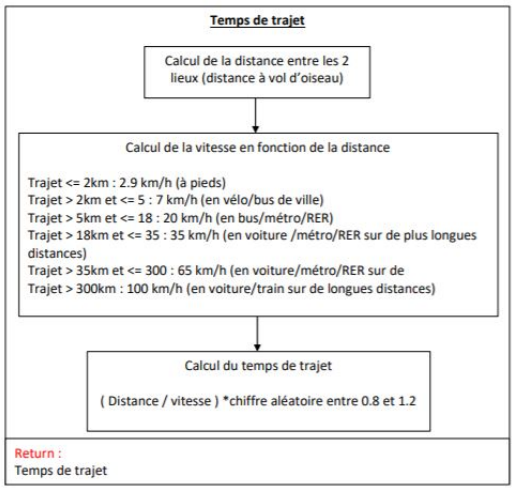

Figure 19: Les déplacements des étudiants entre leur logement et le domicile parental