Rapport final de Projet Pluridisciplinaire d'Informatique Intégrative

# Développement d'un Réseau de Recharge de Véhicules Électriques

Alexis MARCEL
Lucas LAURENT
Noé STEINER
Mathias AURAND-AUGIER

*Responsable du module :*
Olivier FESTOR
Gerald OSTER

24 Mai 2023

# Table des matières

# 1 Contexte du projet

Ce rapport rend compte du Projet Pluridisciplinaire d'Informatique Intégrative dans le cadre de la première année du cycle ingénieur à TELECOM Nancy. L'objectif de ce projet est de concevoir, en groupe, une application en C dédiée à la simulation d'un réseau de recharge de véhicules électriques. Ce projet est encadré par M. Olivier Festor et M. Gérald Oster.

# 2 Introduction

Dans un contexte où la mobilité durable et la réduction des émissions de gaz à effet de serre sont devenues des enjeux majeurs, la transition vers l'utilisation de véhicules électriques se positionne comme une solution de plus en plus pertinente. Ce rapport présente notre projet de développement d'un réseau de recharge de véhicules électriques, en réponse à l'interdiction récente de la Commission européenne de mettre sur le marché des véhicules à moteur thermique à partir de 2023. Notre objectif est de fournir un ensemble de fonctions utiles aux usagers, aux autorités de régulation et aux acteurs économiques pour faciliter le déploiement et le dimensionnement d'un réseau de recharge adapté aux besoins. En utilisant des algorithmes avancés et des données en temps réel, ce projet cherche ainsi à trouver les itinéraires les plus optimaux en termes de distance, de temps et de disponibilité des bornes de recharge pour les conducteurs de voitures électriques.

# 3 Structures de Données

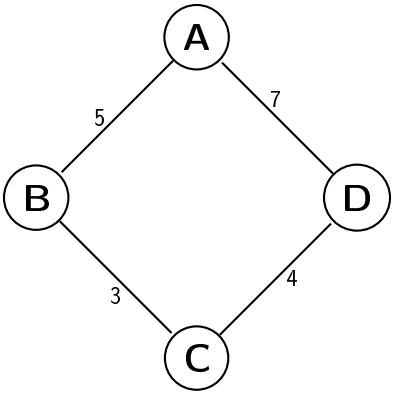
## 3.1 Graph

### 3.1.1 Présentation

Pour parvenir à choisir la structure de données optimale pour notre projet d'optimisation de trajet pour voitures électriques, nous avons exploré différentes options. Notre objectif était de trouver une structure de données qui puisse représenter efficacement les connexions entre les différentes stations, tout en facilitant les calculs d'itinéraires optimaux.

Nous avons commencé par étudier les structures de données vues en cours tels que les listes, les arbres et les graphes. Nous avons d'abord envisagé les arbres. Cependant, nous avons réalisé que cette structure ne convenait pas à notre cas d'utilisation, car elle ne permettait pas de représenter efficacement toutes les connexions entre toutes les stations.

Nous avons donc choisi de représenter le réseau de stations de recharge à l'aide d'un graphe, une structure de données couramment utilisée pour modéliser des systèmes de points connectés. Chaque sommet représente une station de recharge et chaque arête représente un chemin direct entre deux stations. La pondération de chaque arête est la distance entre les deux stations correspondantes. Ainsi, la représentation ci-dessous, est une représentation graphique simplifiée de la structuration de notre graphe. A, B, C et D sont des stations, la valuation des arcs correspond au nombre de kilomètres à vol d'oiseau entre les stations. Ainsi, la valuation de l'arc entre A et B est de 5, entre B et C de 3, entre C et D de 4 et entre D et A de 7 :




Dans notre approche, chaque nœud du graphe représente ainsi une station spécifique, ou une borne de recharge et les arêtes du graphe représentent les connexions entre ces positions, et donc les routes même si nous avons supposé que nous travaillions en vol d'oiseau pour simplifier le problème.

L'utilisation des graphes nous permet de bénéficier d'algorithmes de recherche de chemins bien établis, tels que l'algorithme de Dijkstra que nous avons étudié en cours de MSED ou l'algorithme A*, plus efficace mais aussi plus difficile à mettre en oeuvre.

### 3.1.2 Implémentation

La structure Graph est composée de deux champs : V, qui représente le nombre de sommets du graphe, et adjMat, qui est une matrice d'adjacence représentant les arêtes du graphe. La matrice d'adjacence est un tableau à deux dimensions de taille V*(V+1)/2-V, où chaque élément est un entier représentant la pondération de l'arête correspondante. Si deux sommets ne sont pas connectés, la valeur de l'élément correspondant est -1 :

```
typedef struct Graph {
    int V;
    int* adjMat;
} Graph;
```

Listing 1 – Structure du Graph

### 3.1.3 Spécification algébrique

- $createGraph : Z \rightarrow Graph^*$
- $printGraph : Graph^* \rightarrow void$
- $createGraphFromStations : ChargingStation^* \times Z \rightarrow Graph^*$
- $freeGraph : Graph^* \rightarrow void$
- $dijkstra : void^* \rightarrow void^*$
- $printPath : ChargingStation^* \times Z^* \times Z \times Coordinate^* \times Coordinate^* \rightarrow void$
- $serializeGraph : Graph^* \times char^* \rightarrow void$
- $deserializeGraph : char^* \times Z \rightarrow Graph^*$

### 3.1.4 Parallélisation

Afin de faciliter l'instanciation du graphe, nous avons également mis en place une solution pour la création des arêtes du graph. En effet, la création des arêtes du graph est une opération coûteuse en temps, et nous avons donc décidé de la paralléliser. Pour cela, nous avons réalisé ces 2 structures additionnelles :

```
typedef struct {
    ChargingStation* stations;
    int start;
    int end;
    Graph* graph;
} ThreadParamsGraph;

typedef struct {
    Graph* graph;
    ChargingStation* stations;
    int autonomy;
    int range;
    Coordinate* src;
    Coordinate* dest;
    int* n;
} ThreadParamsDijkstra;
```

Listing 2 – Structure Annexes à Graph

La structure ThreadParamsGraph est conçue pour permettre le partage des paramètres nécessaires pour la création parallèle des arêtes du graphe. Elle contient un pointeur vers un tableau de stations de charge, des indices start et end définissant la plage de stations pour lesquelles le thread doit créer des arêtes, ainsi qu'un pointeur vers le graphe dans lequel les arêtes seront créées.

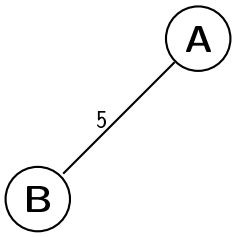
D'autre part, la structure ThreadParamsDijkstra est utilisée pour le partage des paramètres nécessaires à l'exécution parallèle de l'algorithme de Dijkstra. Elle contient un pointeur vers le graphe sur lequel l'algorithme doit être exécuté, un pointeur vers un tableau de stations de charge, ainsi que les paramètres autonomy et range utilisés dans l'algorithme. De plus, elle contient des pointeurs vers les coordonnées source et destination pour le chemin à trouver par l'algorithme de Dijkstra, ainsi qu'un pointeur vers un entier n qui stockera la longueur du chemin trouvé.

Ces structures permettent une modularité et une flexibilité accrue lors de la mise en place de la parallélisation dans le traitement du graphe, ce qui peut conduire à une amélioration significative des performances, en particulier pour les grands graphes.

## 3.2 ChargingStation

### 3.2.1 Présentation

La structure ChargingStation représente une station de recharge, avec des informations telles que le nom de la station, les coordonnées géographiques, le nombre de points de charge et le nombre de points de charge disponibles. Elle contient également une file d'attente pour gérer les véhicules en attente de recharge. Les ChargingStation sont utilisés comme noeuds du graphe :




### 3.2.2 Implémentation

La structure ChargingStation est composée de cinq champs : name, qui est un pointeur vers une chaîne de caractères contenant le nom de la station, coord, qui est un pointeur vers une structure Coordinate contenant les coordonnées géographiques de la station, nbChargingPoints, qui est un entier représentant le nombre de points de charge de la station, nbAvailableChargingPoints, qui est un entier représentant le nombre de points de charge disponibles, et queues, qui est un tableau de pointeurs vers des files d'attente, une pour chaque point de charge.

```
1 typedef struct ChargingStation {
2     char* name;
3     Coordinate* coord;
4     int nbChargingPoints;
5     int nbAvailableChargingPoints;
6     Queue** queues ;
7 } ChargingStation;
```

Listing 3 – Structure ChargingStation

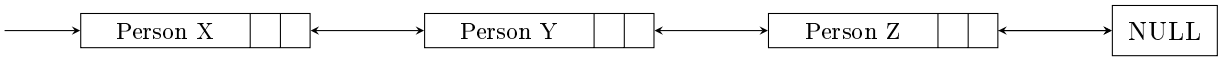
### 3.2.3 Spécification algébrique

- $readJSONstations : char^* \times Z \rightarrow ChargingStation^*$
- $serializeStations : char^* \times ChargingStation^* \times Z \rightarrow void$
- $deserializeStations : char^* \times Z \rightarrow ChargingStation^*$
- $addPersonToStation : ChargingStation^* \times Person^* \times Z \times Z \rightarrow void$
- $removePersonFromStation : ChargingStation^* \times Person^* \rightarrow void$
- $getBestQueue : ChargingStation^* \rightarrow Queue^*$

## 3.3 Queue

### 3.3.1 Présentation

La structure Queue est utilisée pour gérer la liste des véhicules en attente de recharge à une station donnée. Elle est basée sur une liste doublement chaînée, qui permet des opérations d'ajout et de suppression efficaces à la fois en tête et en queue de liste :




### 3.3.2 Implémentation

La structure Queue est composée de trois champs : data, qui est un pointeur vers une structure Person contenant les informations sur le véhicule en attente, next, qui est un pointeur vers la file d'attente suivante, et prev, qui est un pointeur vers la file d'attente précédente.

```
1 typedef struct Queue {
2     Person* data;
3     struct Queue* next;
4     struct Queue* prev;
5 } Queue;
```

Listing 4 – Structure Queue

### 3.3.3 Spécification algébrique

- $createQueue : void \rightarrow Queue^*$
- $del\_person : Queue^* \rightarrow void$
- $push : Queue^* \times Person^* \times Z \rightarrow void$
- $last : Queue^* \rightarrow Person^*$
- $index\_of\_from : Queue^* \times Z \rightarrow Person^*$
- $timeToWait : Queue^* \rightarrow Z$
- $first : Queue^* \rightarrow Person^*$
- $pop : Queue^* \rightarrow void$

# 4 Fonctionnalités et algorithmes

## 4.1 Algorithme de Dijkstra

### 4.1.1 Présentation

Pour déterminer le parcours optimal d'une station à une autre, nous avons utilisé l'algorithme de Dijkstra. C'est un choix naturel pour ce problème, car il trouve le chemin le plus court entre deux sommets d'un graphe pondéré, ce qui est exactement ce dont nous avons besoin pour minimiser la distance de conduite et donc la consommation d'énergie. Nous avons également exploré un potentiel choix alternatif, l'algorithme A*. Cependant, en raison des contraintes de temps liés au projet, nous avons finalement décidé de nous en tenir à l'algorithme de Dijkstra qui est celui vu en cours de MSED, mais en implémentant une version parallèle pour améliorer les performances, et également en retirant un grand nombre de sommets du graphe pour réduire le temps d'exécution.

### 4.1.2 Analyse de compléxité

La complexité de l'algorithme de Dijkstra est en général de $O(V^2)$, où $V$ est le nombre de sommets du graphe. Cependant, si le graphe est implémenté à l'aide d'une liste d'adjacence et d'une file de priorité, la complexité peut être réduite à $O((V+E)logV)$, où E est le nombre d'arêtes. Dans notre cas, nous avons utilisé une matrice d'adjacence pour représenter le graphe, donc la complexité est de $O(V^2)$. Cependant, nous avons également utilisé une version parallèle de l'algorithme, qui réduit considérablement le temps d'exécution. En effet, l'algorithme de Dijkstra est composé de deux boucles imbriquées, et la boucle

interne peut être parallélisée, ce qui permet d'obtenir une complexité de $O(V^2/p)$, où $p$ est le nombre de threads utilisés. En outre, nous avons également réduit le nombre de sommets du graphe, ce qui réduit encore le temps d'exécution.

## 4.2 Instanciation du Graph

### 4.2.1 Load des stations

Pour instancier le graphe, nous avons besoin de charger les stations de charge depuis un fichier JSON. Pour cela, nous avons utilisé la bibliothèque cJSON, qui permet de lire et d'écrire des fichiers JSON en C. Nous avons créé une fonction readJSONstations qui prend en paramètre le nom du fichier JSON et le pointeur vers le nombre de stations que la fonction va actualiser avec le nombre de stations trouvé dans le fichier JSON, et qui renvoie un tableau de stations de charge. Cette fonction utilise la bibliothèque cJSON pour lire le fichier JSON et créer un tableau de stations de charge à partir des données du fichier. Elle renvoie ensuite ce tableau.

### 4.2.2 Instanciation de la matrice d'adjacence

Une fois les stations de charge instanciées, nous pouvons instancier la matrice d'adjacence. Pour cela, nous avons créé une fonction createGraphFromStations qui prend en paramètre un tableau de stations de charge et le nombre de stations, et qui renvoie un graphe. Cette fonction crée un graphe vide, puis crée les arêtes du graphe en utilisant la fonction createGraphFromStations. Enfin, elle renvoie le graphe.

### 4.2.3 Analyse de compléxité

La fonctionnalité d'instanciation du graphe est composée de deux fonctions majeures : createGraphFromStations et createGraphFromStationsThread.

createGraphFromStationsThread est une fonction qui remplit une partie de la matrice d'adjacence du graphe. En particulier, chaque thread exécute deux boucles imbriquées, où l'index i varie de start à end - 1 et l'index j varie de i + 1 à end. Par conséquent, cette partie a une complexité de $O((end - start)^2)$. Dans le cas le plus équilibré, chaque thread doit gérer $n/numThreads$ sommets, ce qui donnerait une complexité d'environ $O((n/numThreads)^2)$ pour chaque thread.

Cependant, l'aspect important de la parallélisation est que les threads s'exécutent simultanément. Par conséquent, bien que chaque thread individuel puisse avoir une complexité de $O((n/numThreads)^2)$, l'ensemble de l'opération createGraphFromStations a une complexité approximative de $O((n/numThreads)^2)$.

Il est à noter que l'efficacité de la parallélisation dépend de nombreux facteurs, dont la disponibilité des ressources du système, la gestion des threads par le système d'exploitation et la concurrence d'accès aux ressources partagées. Par conséquent, dans la pratique, le gain de performances peut ne pas être aussi important que suggéré par cette analyse.

En ce qui concerne la complexité spatiale, chaque thread n'utilise que très peu de mémoire supplémentaire (juste quelques variables locales). Ainsi, la complexité spatiale globale est dominée par la taille du graphe, qui est $O(n^2)$, où n est le nombre de sommets (stations) dans le graphe, car nous utilisons une matrice d'adjacence pour représenter le graphe.

## 4.3 Visualisation d'itinéraires

### 4.3.1 Présentation

Afin de visualiser le parcours trouvé lors de l'exécution de l'algorithme de Dijkstra, nous avons fait appel à une fonctionnalité de Google Maps, permettant de relier des coordonnées entre elles. Ainsi, nous générons une URL Google Maps, qui prend en paramètre chaque étape du parcours. Celle-ci est ensuite ouverte dans le navigateur par défaut de l'utilisateur et affiche le parcours.

## 4.4 Mode simulation

### 4.4.1 Présentation

Le mode simulation permet de simuler le comportement de plusieurs véhicules électriques en circulation. Ce mode permet d'instancier un grand nombre de *Person*, qui correspondent aux "automobilistes". Le mode simulation permet de leur attribuer des destinations et de les faire circuler dans le réseau. Le mode simulation permet également de simuler le temps de recharge des véhicules, en fonction de leur

autonomie et de leur niveau de charge actuel. Nous avons également implémenté un système de queue pour gérer les véhicules en attente de recharge.

# 5 Tests

## 5.1 Introduction

Nous avons effectué des tests sur chacune de nos fonctions pour nous assurer qu'elles fonctionnent correctement. Ces tests comprenaient des cas de tests simples ainsi que des tests de stress pour évaluer la performance et l'efficacité des fonctions. En outre, nous avons utilisé des outils d'analyse dynamique pour détecter les fuites de mémoire et autres problèmes liés à la gestion de la mémoire.

### 5.1.1 Exemple

```
~/Developer/GitLab/telecom_nancy/s6/ppii2/ppii2-grp_30/src git:(v3) (0.989s)
./test_queue

test_queue(14887,0x1fe9c9e00) malloc: nano zone abandoned due to inability to reserve vm space.
Je rentre dans une station Station 1 avec 2 places disponibles
= testing La station doit avoir une place de moins disponible
[OK] ✅
= testing La personne doit être dans la première file
[OK] ✅
= testing La personne ne doit pas attendre
[OK] ✅
= testing La personne doit charger pendant 300 secondes
[OK] ✅
Je sors de la station Station 1
= testing La station doit avoir une place de plus disponible
[OK] ✅
= testing La première file doit être vide
[OK] ✅
= testing La personne ne doit plus charger
[OK] ✅
Je rentre dans une station Station 1 avec 2 places disponibles
= testing Il reste 10 km d'autonomie, la personne doit charger pendant 100 secondes
[OK] ✅
Je rentre dans une station Station 1 avec 1 places disponibles
= testing La station doit être pleine
[OK] ✅
= testing La personne doit être dans la deuxième file
[OK] ✅
= testing La personne ne doit pas attendre
[OK] ✅
= testing La personne doit charger pendant 100 secondes
[OK] ✅
```

FIGURE 1 – Sortie Console

# 6 Gestion de projet

## 6.1 Équipe de projet

Ce projet est un projet local réalisé en groupe de 4 personnes :
- Alexis MARCEL
- Lucas LAURENT
- Noé STEINER
- Mathias AURAND-AUGIER

Le comité de pilotage est constitué de :
- Olivier FESTOR
- Gérald OSTER

Ces personnes constituent les parties prenantes de notre projet ainsi que les acteurs influents sur le livrable.

| *Ceux qui...* | *Sont...* |
|---|---|
| *... Demandent, financent le projet, propriétaires du livrable* | Télécom Nancy. |
| *... Pilotent le projet et assurent la coordination avec le client* | Les professeurs gérants les différents groupes. |
| *... Réalisent le projet* | Les élèves de l'équipe projet. |
| *... Sont également concernés* | Les entreprises de bornes de recharges des voitures électriques, les fabriquants... |

FIGURE 2 – Parties prenantes

## 6.2 Organisation au sein de l'équipe projet

Nous avons réalisé plusieurs réunions, en présentiel dans les locaux de Télécom Nancy mais la plupart de notre collaboration a eu lieu sur Discord. Ces réunions nous ont permis de mettre en commun nos avancées régulièrement, de partager nos connaissances sur des problématiques et de nous organiser de manière optimale. En plus des réunions d'avancement régulières, nous avons également réalisé des réunions techniques afin de résoudre un problème ou bien de réfléchir à la conception. Les comptes rendus des réunions réalisées sont présents dans l'Annexe 1.

Contrairement au premier projet, nous avons choisi de travailler cette fois-ci tous ensemble plutôt qu'individuellement. Nous avons donc utilisé une application pour que nous puissions tous modifier le code en même temps, ce qui nous a permis de tous travailler sur le même code en même temps.

Ensuite, nous avons utilisé GitLab pour gérer les différentes versions du développement de notre application, ainsi que les différentes branches nous permettant de travailler simultanément sans conflit.

Enfin, la rédaction des différents comptes rendus de réunion et des rapports a été faite en LaTeX.

## 6.3 Objectifs SMART

La méthode SMART que l'on rappelle ci-dessous nous a permis de définir nos différents objectifs :

| | Critère | Indicateur |
|---|---|---|
| S | Spécifique | L'objectif est clairement défini. |
| M | Mesurable | On peut suivre et quantifier la progression de l'objectif. |
| A | Atteignable | L'objectif prend en compte la capacité des membres du projet à l'atteindre et des moyens mis à disposition. |
| R | Réaliste | L'objectif doit être réaliste, réalisable et pertinent par rapport à la situation. |
| T | Temporellement défini | Le projet doit être limité dans le temps, avec une date de fin. |

FIGURE 3 – Objectifs SMART

## 6.4 Matrice des objectifs

Nous avons conçu, à l'aide de la méthode SMART, la matrice des objectifs suivante :

| | Livrable | Documentation | Valorisation | Acquisition compétences |
|---|---|---|---|---|
| **Insuffisant** | Avoir seulement un graph fonctionnel | Pas de recherche effectuée | Projet non approuvé par tous les membres | Acquis superficiels |
| **Réussite acceptable** | Le Dijkstra fonctionne et renvoit bien le plus court chemin | Recherche des différents algortihmes | Projet approuvé par tous les membres | Quelques améliorations en C et en SD |
| **Bon travail** | Les simulations fonctionnent | Recherche | | Etre capable de refaire une application similaire |
| **Excellent** | Interface graphique qui réprésente notre modèle | Tableau comparatif avec applis similaires | Conception validée et approuvé par les professeurs | Avoir intégralement compris toutes les notions utilisées dans notre projet |

FIGURE 4 – Matrice des objectifs

En ce qui concerne le livrable, il sera difficile compte tenu des délais d'atteindre tous les objectifs du projet.

## 6.5 Triangle délai-qualité-coût

Afin d'établir des objectifs cohérents, et réalisables dans les délais, nous avons réalisé le triangle qualité-coût-délai. On remarque ainsi, les délais étant courts, que nous avons tout intérêt à ne pas se fixer des objectifs trop ambitieux sous peine de devoir renoncer à certaines fonctionnalités et de ne pas rendre le livrable annoncé initialement.

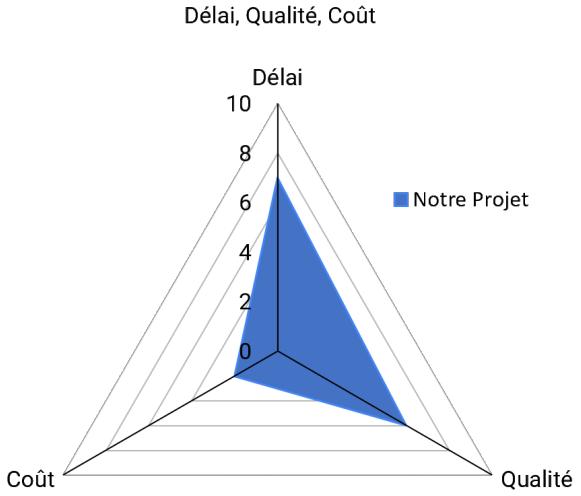



FIGURE 5 – Triangle QCD

## 6.6 Matrice SWOT

Afin d'avoir une vision plus globale de nos ressources et des facteurs internes et externes agissant sur le projet, nous avons ensuite réalisé la matrice SWOT (Strengths, Weaknesses, Opportunities, Threats) de notre projet.

| Forces : | Faiblesses : |
|---|---|
| -Ressources immaterielles : Cours de C et de Structure de données + MOOC GdP<br>-Ressources matérielles : Chacun a son propre ordinateur pour travailler<br>-Ressources financières : Aucun besoin financier particulier<br>-Ressources humaines : compétences des membres du projets en C et SD | -Ressources humaines : Malgré PPII 1, encore peu d'expérience dans la gestion de projets<br>-Ressources organisationnelles : des délais très courts pour le projet. |
| **Opportunités :** | **Menaces :** |
| -Possibilité de travailler à l'école : salles ouvertes<br>-Possibilité de demander de l'aide aux professeurs ou aux élèves<br>-Possibilités d'accéder aux machines de l'école | Partiels en fin d'année<br>Panne de machines/Accidents |

FIGURE 6 – Matrice SWOT

On peut ainsi remarquer que notre projet présente de nombreux points forts notamment grâce aux connaissances acquises lors des cours de Télécom Nancy. Cependant, plusieurs facteurs internes constituent nos faiblesses notamment les courts délais qui nous obligent à être concis et efficaces dans notre travail. Cependant, nous avons également des opportunités notamment celle de pouvoir demander de l'aide aux professeurs, ou encore le fait que les salles soient ouvertes et à notre disposition pour les réunions de groupe.

De plus, nous devons anticiper les charges de travail dans le cadre de notre formation à Télécom Nancy qui s'avèrent être plus élevées au mois de mai pour les partiels de fin d'année. Nous allons donc devoir prendre cela en compte dans notre gestion des tâches.

## 6.7 Profil de projet

Afin d'avoir une vision plus globale sur notre projet, nous avons également réalisé le profil du projet (le budget étant égal à 0, nous avons choisi de ne pas le représenter dans notre profil). On remarque que, du fait des nombreuses fonctionnalités que nous avons l'intention d'implémenter dans notre application, notre projet est de taille moyenne mais de complexité élevée.

Cependant, les enjeux du projet ne sont pas très importants (en dehors de la note finale qui compte dans notre moyenne) car l'échec du projet n'engendrera pas la chute d'une organisation et le budget est négligeable.

De plus, au vu de l'état de l'art établi, l'innovation du projet est importante puisque peu d'application de ce type existent actuellement.

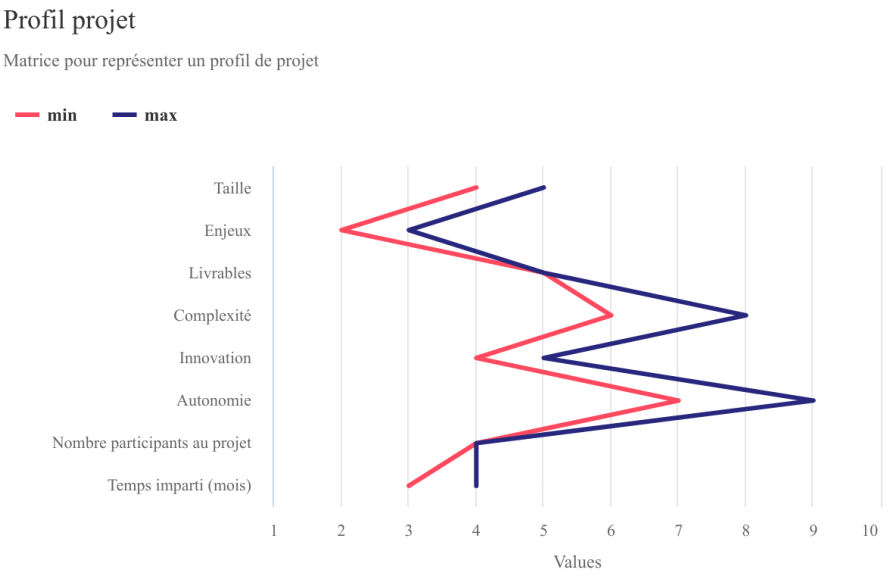



FIGURE 7 – Profil du projet

## 6.8 WBS : comment concrétiser l'application

Ceci étant fait, nous avons maintenant choisi de détailler les lots de travail à effectuer pour fabriquer notre application. Nous avons ainsi réalisé le WBS (Work Breakdown Structure) de notre application : il apparait ainsi les grandes étapes de notre projet que sont : définition du cadre de l'application, développement des fonctionnalités de l'application et écriture du rapport.

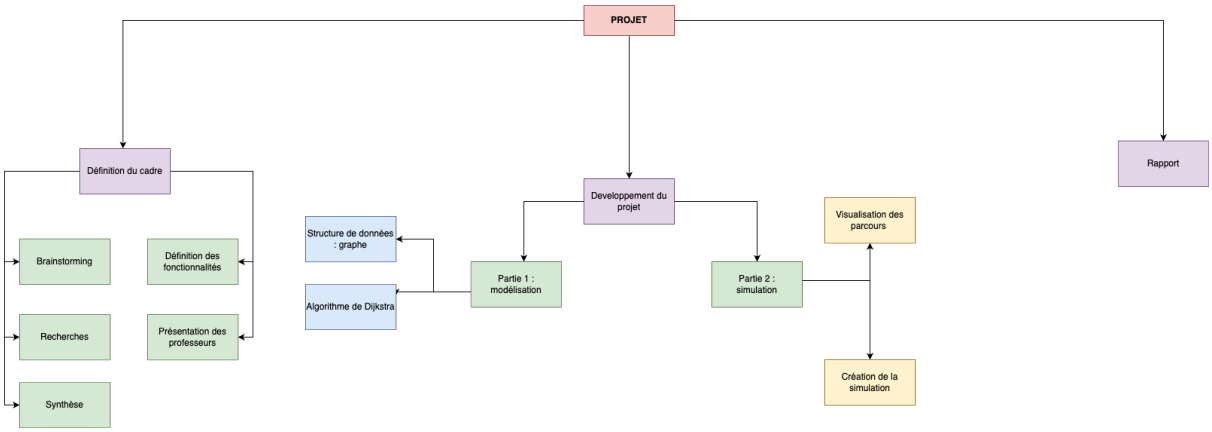



FIGURE 8 – WBS

## 6.9 Diagramme de Gantt : planification

Maintenant que nous avons vu en détail les lots de travail qui constituent notre application, il faut les mettre en relation pour créer un planning efficace où chaque tâche est effectuée dans l'ordre.

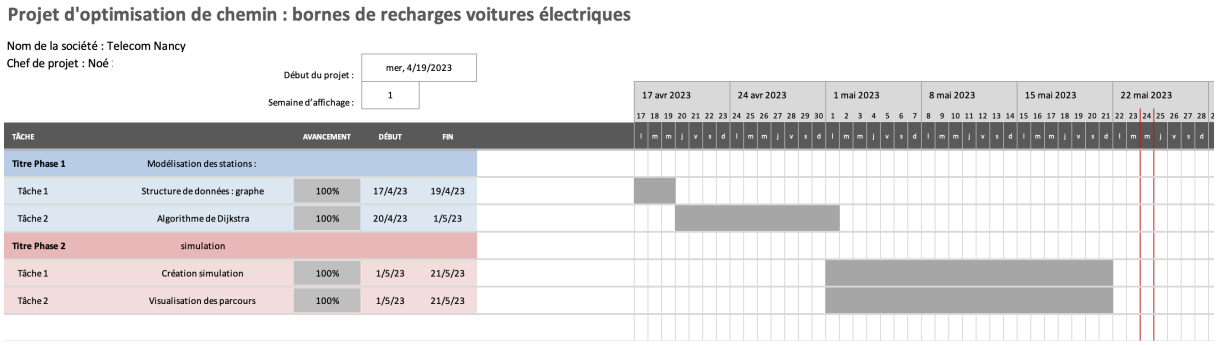



FIGURE 9 – Diagramme de GANTT

Ce diagramme est une première version générale des tâches à effectuer, il sera modifié et détaillé davantage une fois la conception et les maquettes du projet réalisées.

## 6.10 Matrice RACI

Nous avons choisi de ne pas faire de matrice RACI à cause des choix d'organisation que nous avons faits. En effet, nous avons décidé de travailler en groupe sur toutes les tâches, ce qui fait que nous sommes tous responsables de toutes les tâches et qu'il n'y a, de ce fait, pas de répartition précise des tâches.

## 6.11 Gestion des risques

Nous avons également pensé à prévoir une partie des risques pouvant se dresser sur notre route, les risques les plus classiques étant la gestion du temps et le manque de compréhension de certaines personnes de l'équipe.

| Description | Gravité 1-4 | Fréquence 1-4 | Criticité | Resp | Prévention | Réparation / Plan B |
|---|---|---|---|---|---|---|
| Un des membres de l'équipe se démotive ou se désinteresse du projet | 3,5 | 2,5 | 8,75 | Noé Steiner | *A chaque réunion, faire le bilan de ce qui a été fait. La mise en commun de toutes les avancées du groupe est indispensable pour ne perdre personne.* | |
| Mésententes dans l'équipe | 3 | 2,5 | 7,5 | Mathias Aurand-Augier | *Ecouter les autres et se remettre en question. Expliciter les problèmes et trouver une solution commune* | Proposition de changement de tâches vers une partie jugée plus intéressante |
| Délai non respectés/Mauvaise gestion du temps | 3 | 2 | 6 | Alexis Marcel | *Respecter au mieux le diagramme de GANTT établi lors de l'étude préalable du projet.* | Réunion d'urgence pour combler les vides et répartir la charge de travail. |
| Le résultat ne correspond pas aux attentes du client | 3,5 | 1,5 | 5,25 | Mathias Aurand-Augier | *Tenter de respecter au mieux le cahier des charges établi* | |
| Un des membres de l'equipe est incompétent(e) | 2 | 2 | 4 | Noé Steiner | *Réactualiser les connaissances nécessaires, et planifier des formations en fonction des besoins. Mettre à disposition les ressources nécéssaires.* | Alléger les tâches et mettre à disposition les éléments nécessaires pour poursuivre le travail. |
| L'ordinateur ne se lance pas le jour de la soutenance | 3,5 | 1 | 3,5 | Alexis Marcel | *Chacun apportera son ordinateur pour que chaque personne soit en mesure de lancer le projet en cas de problème* | |
| Le projet est inutilisable par le prof (problème de dépendances, ...) | 3 | 1 | 3 | Lucas Laurent | *Expliquer au mieux dans un fichier README tous les modules à installer pour faire fonctionner l'application* | Consolider la documentation utilisateur |

FIGURE 10 – Plan de gestion des risques

# 7 Conclusion

En conclusion, ce projet s'est révélé être une expérience d'apprentissage extrêmement précieuse, nous permettant d'affiner et de développer nos compétences en programmation en langage C. Nous avons approfondi notre compréhension des structures de données, notamment comment les utiliser efficacement pour manipuler, stocker et accéder aux informations. Ce projet nous a aussi donné l'opportunité d'appliquer des algorithmes complexes, comme celui de Dijkstra, dans un contexte réel, nous permettant d'apprécier l'importance de ces outils dans la résolution de problèmes concrets.
En travaillant sur un problème réel - l'optimisation du placement des stations de charge pour véhicules électriques - nous avons pu saisir l'importance des implications de notre travail. Cela a ajouté une dimension plus large à notre apprentissage, en nous faisant comprendre comment les compétences techniques que nous avons acquises peuvent avoir un impact sur des questions d'une grande pertinence sociétale, comme la transition vers une mobilité plus durable.
De plus, la mise en évidence des enjeux de l'énergie durable dans notre projet nous a permis de nous immerger dans un secteur d'une importance cruciale pour l'avenir de notre société. L'efficacité de la distribution de l'énergie, en particulier pour les véhicules électriques, est une question clé dans la lutte contre les changements climatiques. En participant à la résolution de ce problème à travers notre projet,

nous avons non seulement appliqué nos compétences techniques, mais également contribué à un domaine qui aura un impact durable et positif sur notre avenir.
En somme, ce projet nous a offert une occasion inestimable de croissance personnelle et professionnelle. Il nous a permis d'acquérir et de consolider des compétences techniques, tout en nous faisant prendre conscience de l'importance des implications de notre travail dans le monde réel, en particulier dans le domaine de la mobilité et de l'énergie durable.

# 8 Annexes

# Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

5 Avril 2023

## Projet PPII - Compte rendu n°01 - réunion de lancement

| Participants: | Lieu: |
|---|---|
| • Alexis : Présent<br>• Noé : Présent<br>• Lucas : Présent<br>• Mathias : Présent | • Le 5 avril 2023<br>• De 10h à 12h<br>• Visioconférence sur Discord |

### *Ordre du jour:*

- Présentation du projet et des objectifs
- Répartition des tâches
- Établissement du calendrier prévisionnel

# Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

19 Avril 2023

## Projet PPII - Compte rendu n°02 - réunion d'avancement

| Participants: | Lieu: |
|---|---|
| • Alexis : Présent<br>• Noé : Absent<br>• Lucas : Présent<br>• Mathias : Présent | • Le 19 avril 2023<br>• De 14h à 16h<br>• Visioconférence sur Discord |

### *Ordre du jour:*

- Mise à jour sur le progrès de la Partie 1
- Sérialisation des données pour gagner en performance
- Discussion des problèmes rencontrés et des solutions possibles

# Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

24 Avril 2023

## Projet PPII - Compte rendu n°02 - réunion d'avancement

| Participants: | Lieu: |
|---|---|
| • Alexis : Présent<br>• Noé : Présent<br>• Lucas : Présent<br>• Mathias : Présent | • Le 24 avril 2023<br>• De 14h à 16h<br>• Visioconférence sur Discord |

### *Ordre du jour:*

- Discussion sur la fin de l'implémentation de l'algo
- Discussion des problèmes rencontrés et des solutions possibles
- Discussion sur les optimisations possibles

#### *Information échangées*

- L'algo présentait quelques soucis qui ont été corrigés.
- La première version de l'algo est désormais fonctionnelle.
- Il faut maintenant l'optimiser. Problème mis en avant : perte de place mémoire par notre algo.

#### *Remarques / Questions*

Aucune question ou remarque spécifique relevée.

### *Décisions*

- On utilisera une sérialisation des données pour gagner en performance

### *Actions à suivre / Todo list*

- Finir l'implémentation de l'algo (tout le monde).
- Commencer à réfléchir pour la deuxième étape du projet (tout le monde).

### *Date de la prochaine réunion*

La prochaine réunion aura lieu le Mercredi 3 Mai 2023 (horaire à définir). En attendant, de multiples séances de travail seront organisés pour travailler ensemble sur le code.

# Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

3 Mai 2023

## Projet PPII - Compte rendu n°3 - réunion technique

| Participants: | Lieu: |
|---|---|
| • Alexis : Présent<br>• Noé : Présent<br>• Lucas : Absent<br>• Mathias : Présent | • Le 3 mai 2023<br>• De 15h à 17h<br>• Visioconférence sur Discord |

### *Ordre du jour:*

- Établir un plan pour la deuxième étape du projet
- Discussion sur la logique de programmation pour la simulation de plusieurs utilisateurs et la gestion des files d'attente
- Etudes des différentres structures de données possibles

# Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

17 Mai 2023

## Projet PPII - Compte rendu n°4 - réunion d'avancement

| Participants: | Lieu: |
|---|---|
| • Alexis : Présent<br>• Noé : Présent<br>• Lucas : Présent<br>• Mathias : Présent | • Le 17 mai 2023<br>• De 14h à 16h<br>• Visioconférence sur Discord |

### *Ordre du jour:*

- Présentation des avancements sur le module de simulation
- Discussion des problèmes rencontrés et des solutions possibles
- Préparation de la présentation finale du projet