

Le palais du bonbon

Projet Data / BI



06 juillet 2018

EXIA.CESi Labège – 3e annee

Dylan CATTELAN – Anais BANOS – Anthony LUQUE – Cédric MONTES

**Table des matières**

[1. Cadrage du projet 3](#_Toc518682185)

[Charte projet 3](#_Toc518682186)

[Cahier des charges 4](#_Toc518682187)

[GESTION 5](#_Toc518682188)

[2. Réalisation du projet 8](#_Toc518682189)

[DOSSIER DE CONCEPTION DE BDD 8](#_Toc518682190)

[DOSSIER REALISATION DU PROJET 8](#_Toc518682191)

[3. Clôture du projet 10](#_Toc518682192)

[BILAN DE L’IMPACT DES AMELIORATIONS APPORTEES 10](#_Toc518682193)

[Axes d’améliorations des différents services 10](#_Toc518682194)

[Picking 14](#_Toc518682195)

[Livraison 16](#_Toc518682196)

[4. Conclusion 21](#_Toc518682197)

**Table des figures**

[Figure 1 – OBS 6](#_Toc518682198)

[Figure 2 - WBS 6](#_Toc518682199)

[Figure 3 - WBS 7](file:///C:\Users\Dylan%20Cattelan\Desktop\BIProject\Project%20Management\Dossier%20Réalisation%20de%20Projet\Rapport%20de%20Projet%20Data%20BI.docx#_Toc518682200)

[Figure 4 – PERT 7](#_Toc518682201)

[Figure 5 - GANTT (MS Project) 8](#_Toc518682202)

[Figure 6 - Capture d'écran Cloud9Charts 10](#_Toc518682203)

[Figure 7 - Solveur fabrication 13](file:///C:\Users\Dylan%20Cattelan\Desktop\BIProject\Project%20Management\Dossier%20Réalisation%20de%20Projet\Rapport%20de%20Projet%20Data%20BI.docx#_Toc518682204)

[Figure 8 - Solveur conditionnement 14](file:///C:\Users\Dylan%20Cattelan\Desktop\BIProject\Project%20Management\Dossier%20Réalisation%20de%20Projet\Rapport%20de%20Projet%20Data%20BI.docx#_Toc518682205)

[Figure 9 - Graphe basique 17](#_Toc518682206)

[Figure 10 - Graphe + courts chemins 18](#_Toc518682207)

[Figure 11 - Graphe optimisé 19](#_Toc518682208)

[Figure 12 - Plannings 21](#_Toc518682209)

# Cadrage du projet

## Charte projet

### Objectifs

Ce projet Data/BI, appelé « Le Palais du Bonbon », avait pour but de remettre en place une hiérarchie Base de Données, développer un générateur de commandes de bonbons, optimiser la production de bonbons et de paquets, optimiser la chaine de paquetage des bonbons, et optimiser la livraison de ceux-ci.

Afin de réaliser cela, nous avions un jeu de données disponible en fichier Excel, et nous devions adapter nos réflexions à celles-ci, tout comme réfléchir aux possibilités manquantes ( par exemple, le nombre de camions disponibles pour livrer les bonbons. )

### Enjeux

Comme cité précédemment, les enjeux étaient d’optimiser la production et d’augmenter le retour sur investissement quotidien de l’entreprise. De plus, nous devions également optimiser les livraisons ainsi que la zone de picking, zone où sont finaliser les commandes.

Nous devions également créer un jeu de commandes afin de pouvoir en sortir des indicateurs et des résultats précis.

### Acteurs

Afin de réaliser ce projet, nous étions 4 étudiants de l’EXIA.CESi Labège 3e année, et nous avions 1 semaine pour réaliser cela. Nous avons utilisé GitHub.com afin de versionner les modifications et de ne rien perdre, Oracle 12 pour la base de données relationnelle, MongoDB 4.0 pour la base de données NoSQL, Talend afin d’interconnecter ces bases de données, Cloud9Charts pour le Dashboard, et Python 3.7 pour le générateur.

### Définition des responsabilité

Dylan CATTELAN, Chef de Projet, s’est chargé du générateur de données, de la création de données ainsi que du Dashboard.

Anais BANOS s’est chargée de la recherche opérationnelle et de la base de données Oracle, ainsi que de la gestion de projet.

Anthony LUQUE s’est chargé de l’interconnexion entre les bases de données via Talend, ainsi que de l’optimisation Picking / Shipping à l’aide des théories des graphes.

Cédric Montes s’est chargé de la gestion de projet, de la structure de la base de données, des indicateurs ainsi que de la base de données Mongo DB.

## Cahier des charges

### Besoins

* Optimiser la production ainsi que l’envoi et la préparation des colis
* Générer des commandes aléatoirement en fonction d’un nombre de commandes demandées
* Affichage de graphiques des données ciblées en tant qu’indicateurs
* Création de rôles et droits utilisateur sur la base Oracle

### Contraintes

Durant ce projet, nous avions comme contrainte d’utiliser une base de données Oracle et de nous baser sur les données fournies via fichier Excel. De plus, nous devions utiliser MS Project pour réaliser le GANTT.

En dehors de ces contraintes, il n’y en avait pas d’autres.

## GESTION

### OBS

Figure 1 – OBS

### PBS

Figure 2 - WBS

### WBS

Figure 3 - WBS

### PERT

### 

Figure – PERT

### GANTT (MS PROJEcT )

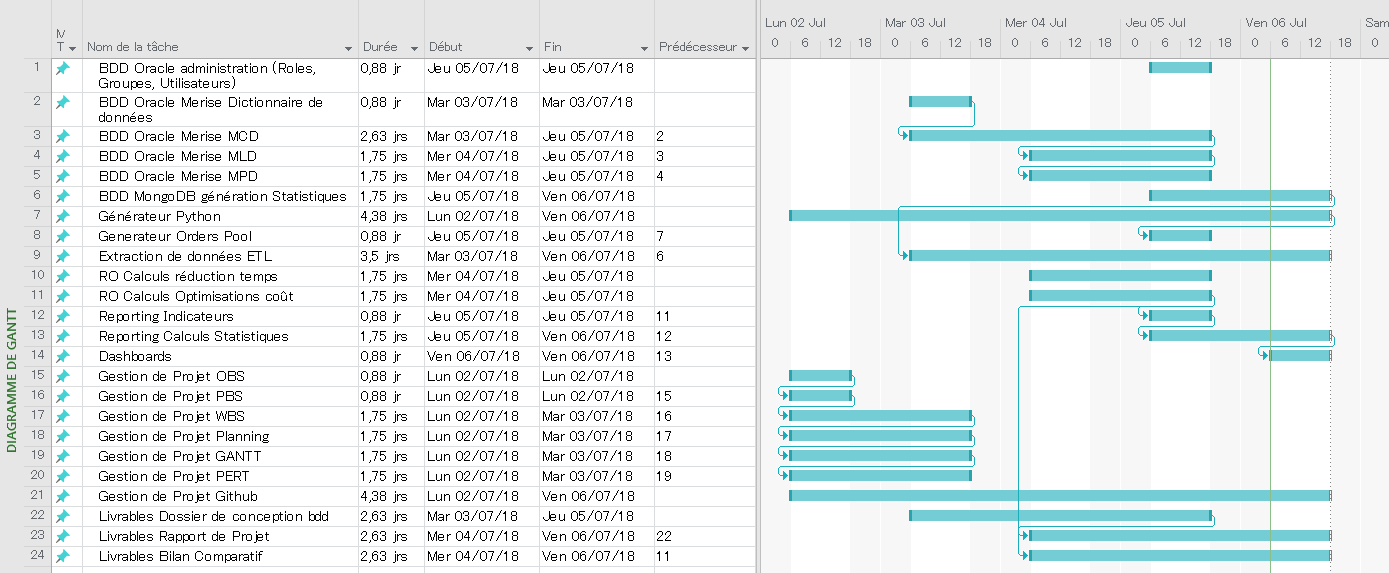


Figure - GANTT (MS Project)

( Tous ces fichiers sont disponibles individuellement dans le dossier « Annexes » du projet. )

# Réalisation du projet

## DOSSIER DE CONCEPTION DE BDD

CF dossier « Dossier de Conception BDD », un dossier à part entière a été écrit.

## DOSSIER REALISATION DU PROJET

### DESCRIPTION ET FONCTIONNEMENT DES GENERATEURS

Le générateur, développé en Python 3.7, est utilisé afin de créer des références ainsi que des commandes de manière aléatoire.

Ce dernier est connecté à la base de données Oracle, et peut SELECT des données de n’importe quelle table, et INSERT INTO les tables Orders et CandyReferences, et UPDATE la table Stock.

Nous avons choisi de mettre en place ce générateur de telle sorte à ce qu’il récupère n’importe quelle donnée ajoutée dans la base Oracle, ce qui fait qu’il suivra la progression de la base, et pourra se tenir à jour de chaque changement effectué, tout comme vérifier les références si elles existent ou pas.

L’utilisation de ce générateur est relativement simple. Il suffit de se placer à la racine du dossier « BIProject », là où se situe le fichier generator.py, et de télécharger une libraire servant à se connecter à la base Oracle facilement en faisant « pip install cx\_oracle »

Esuite, il faut rentrer dans le shell Python grâce à la commande « python », et d’ici, vous pourrez importer generator.py en utilisant la commande « import generator ».

Enfin, il suffit d’appeler la commande principale et de lui donner un nombre de commandes à générer, et elle se chargera de créer les commandes, comparer les références si elles existent ou non, et les créer si elles n’existent pas. Il faut, de ce fait, faire « generator.createOrderPool(nbOrders) » où nbOrders est un INT.

Ce générateur a été pensé pour également gérer les stocks des matières premières de la table Stock, gérer également les prix totaux des commandes, ou encore le mode d’envoi, le numéro de tracking des colis, les stocks de bonbons disponibles et l’optimisation des machines.

Pour plus d’informations sur les différents fichiers et méthodes, une description fichier par fichier est disponible dans le Dossier de Conception BDD.

### PRESENTATION DES INDICATEURS ET TABLEAUX DE BORD

Afin de proposer une 1ere version du tableau de bord, nous avons utilisé un jeu de données disponible dans le dossier « Jeu de Données MongoDB ».

Nous avons utilisé le programme Cloud9Charts ainsi que le site web associé afin de créer un dashboard de qualité affichant les informations que nous voulons. De ce fait, il a suffit de mettre en place ce programme sur l’ordinateur ayant la base de données MongoDB, et de créer des requêtes directement depuis le site internet, et les données sont directement transmises et gérées via le site.

Un point positif à ce programme est que vu que les informations sont disponibles via site internet, elles sont gérables et consultables de partout.

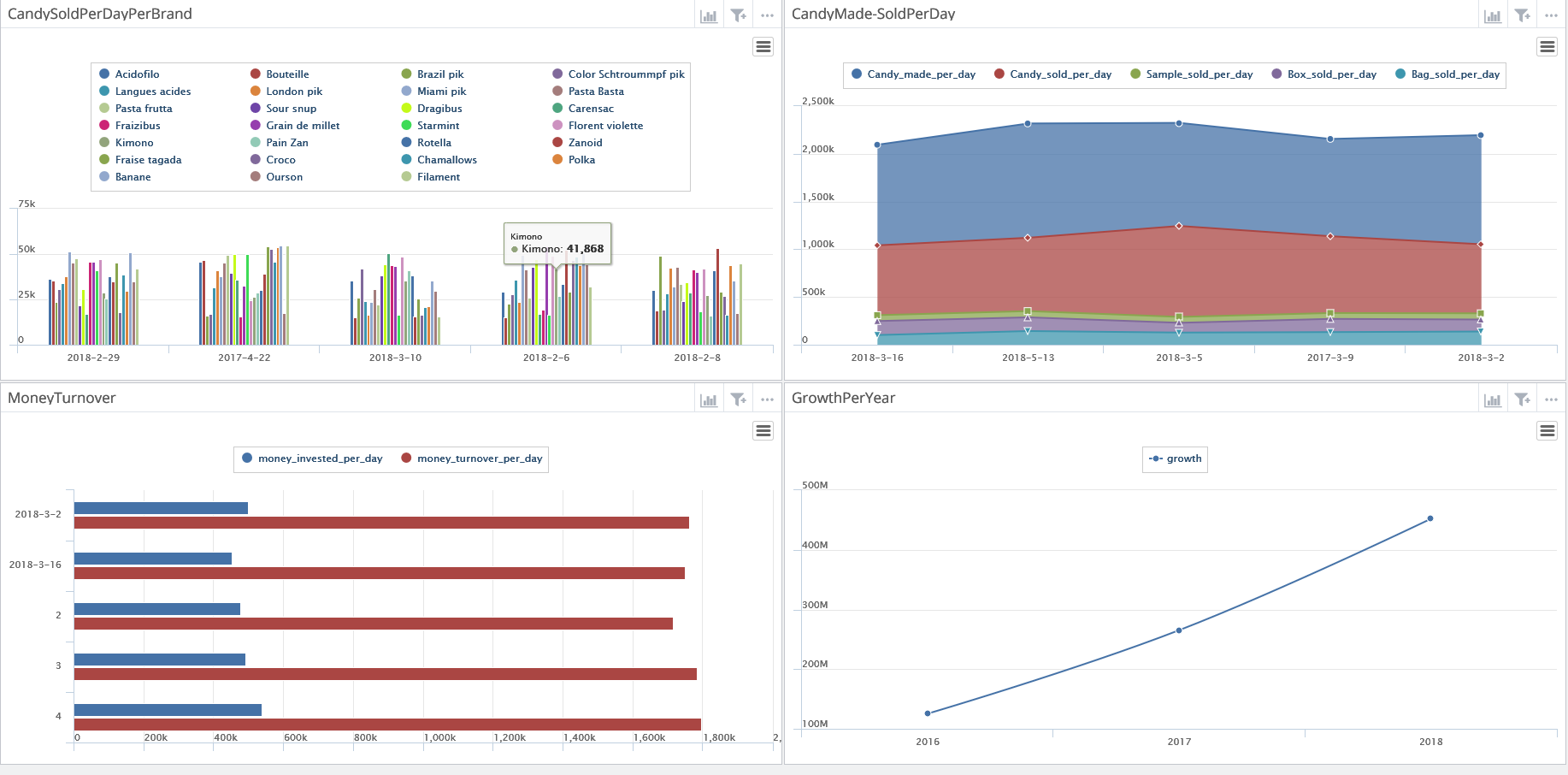


Figure 6 - Capture d'écran Cloud9Charts

# Clôture du projet

## BILAN DE L’IMPACT DES AMELIORATIONS APPORTEES

Actuellement nous avons une entreprise qui fonctionne avec 3 grand services :

* Fabrication
* Conditionnement
* Préparation des commandes

Ces services fonctionnent de manière totalement aléatoire. Plus clairement on peut dire que chacun d’entre eux traite les différentes demandes sans ordre d’importance.

Pour revenir sur les faits, nous avons un service de **Fabrication** qui regroupe 4 machines :

* La première s’occupe de tous les Bonbons en variante acidulé
* La seconde s’occupe des variantes sucrées
* La troisième des variantes gélifiées
* La quatrième est une machine bivalente qui peut produire des Bonbons sucrés ou des bonbons gélifiés

Ces machines exécutent les différentes demandes de production de bonbon à la suite sans avoir aucune logique d’exécution ou aucun calcul de priorité.

Sur le même fonctionnement nous avons le service de **Conditionnement** qui possède 6 machines. Trois d’entres elles s’occupe des commandes en sachets, deux autres des boites et la dernière des échantillons.

Nous pouvons dire que chaque machine possède ses propres caractéristiques au niveau de la cadence d’exécution et au niveau du délais de changement d’outils.

Le service du **Picking** est assez chronophage, en l’état, dans la chaine de production des bonbons. Actuellement un paquet doit parcourir l’intégralité des 2 boucles comportant chacune 20 gares de 100 zones de picking différentes.

Nous avons un service de **livraison** qui prends énormément de ressources avec des envoies et retour de Camion/Bateau/Avions qui se font en départ de la fabrique vers tous les pays de livraison de manière indépendante.

## Axes d’améliorations des différents services

Nous avons à améliorer certaines parties de l’entreprise, comme par exemple la partie Logistique et Conditionnement.

Nous allons donc expliquer en detail comment nous pouvons optimiser la charge de travail entre les différentes machines de ces services.

En Recherche Opérationnelle, nous pouvons maximiser ou minimiser la fonction économique définissant ce que nous devons améliorer.

Dans ce cas, nous devons améliorer le temps d’utilisation des machines.

### SYSTEME D’EQUATION - SERVICE Fabrication

Nous avons autant de variables que de machines, en l’occurrence 4.

X1, X2, X3, X4 sont respectivement le nombre d’heures de travail de chaque machine.

Nous avons choisi de prendre une période de 12h maximum pour pouvoir effectuer nos calculs et pouvoir faire une optimisation dynamique par cycle de 12h, soit 2 cycles par jour si l’entreprise décide de les faire fonctionner à plein régime.

De ce fait nous avons :

* FC = X1, X2, X3, X4 <=12

Ensuite nous prenons en compte les différentes caractéristiques des machines qui nous ont été transmise et nous les retranscrivons en inéquations. Nous avons décidé de prendre un jeu de données aléatoire pour déterminer un nombre moyen de bonbon, par type de variante, à fabriquer sur un cycle.

Notre Fonction économique regroupe toutes les heures de fonctionnement des 4 machines confondus, nous devons donc minimiser cette dernière tout en essayant de maximiser la production. Pour cela nous avons choisi de permettre le fait que la production quotidienne n’atteigne pas forcément la demande.

La cadence est le nombre de bonbon produit en 1min/machine.

Le délai de changement d’outils est le temps (en min) qu’il faut pour changer les outils de la machine, on part du principe que l’on fait en moyenne un changement de type de bonbon toutes les heures et donc un changement d’outils.

Nous avons :

* FC1 = 750 \* X1 \* 60 – (25 \* 750 \* X1) >= 150000
* FC2 = [1230 \* X2 \*60 – (45 \* X2 \* 1230)] + [1230 \* X4 \*60 – (45 \* X4 \* 1230)] >=20000
* FC3 = [625 \* X3 \* 60 – (25 \* X3 \* 625)] + [625 \* X4 \*60 – (25 \* X4 \* 625)] >=25600
* FC4 = X1, X2, X3, X4 <=12
* Fe(min)= X1 +X2 +X3 + X4

FC1 correspond aux bonbons acides

FC2 correspond aux bonbons sucrés

FC3 correspond aux bonbons gélifiés

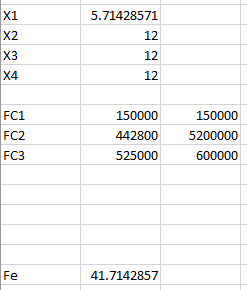
Pour résoudre ces inéquations nous avons utilisés la méthode solver qui nous a donné les résultats suivants :

Figure 7 - Solveur fabrication

Nous devons donc allumer la 2e, la 3e et la 4e machine Durant 12 heures afin de valider les différentes commandes. Nous avons rempli la totalité des commandes de bonbons acides, mais, comme nous pouvons le voir, sur FC2 et FC3, les bonbons sucrés et gellifiés ont une trop haute demande pour que les machines associées puissent remplir les commandes en 12 heures.

Notre fonction économique définit le total en heures durant lequel les machines doivent êtres allumées, sur une plage de 12 heures.

### système d’équation - Service de conditionnement

Comme définit précédemment, nous avons autant de variables que de machines. Dans ce service, nous en avons 6, donc nous prenons 6 variables: X1, X2, X3, X4, X5, X6.

Cela correspond aux heures de travail de chaque machine, sur une plage de 12 heures, comme pour le service de Conditionnement.

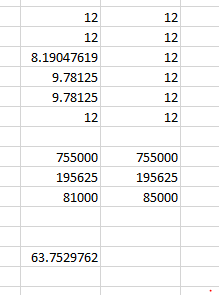
La cadence est le nombre de conditionnements effectués en 1min/machine.

Le délai de changement d’outils est le temps (en min) qu’il faut pour changer les outils de la machine, on part du principe que l’on fait en moyenne un changement de type de conditionnements toutes les heures et donc un changement d’outils.

Nous avons donc les inéquations suivantes :

* FC1: [500\*60\*X1 – (500\*15\*X1)] + [ 500\*60\*X2 – (500 \* 60\* X2)] + (750\*60\*X3 – (750\*25\*X3)>=755000
* FC2: [200 \* 60 \* X4 – (200\*X4 \* 10)] + [200\*60\*X5- 200\*X5\*10)] >=195625
* FC3: 150 \* X6 \* 60 – (150 \* X6 \* 15)>=85000
* Fe(min)= X1 +X2 X3 +X4 + X5 +X6

Pour résoudre ces inéquations nous avons utilisés la méthode solver qui nous a donné les résultats suivants

La première des trois machines concerne les paquets, les deux autres les échantillons.

Comme nous pouvons le voir sur cette image, nous avons rempli les commandes de paquets et de boites, mais pas celle des échantillons, il en manque 4000. Ce resultat arrive car la productivité des machines, bien qu’augmentée au possible, ne permet pas de générer autant de bonbons en si peu de temps. Nous pouvons en faire que 81000 en 12 heures.

Au contraire, concernant les paquets et les boites, il y a encore possibilité d’amélioration car nous avons rempli toutes les commandes sans allumer les machines durant les 12 heures totalement, il reste encore possibilité à produire d’autres bonbons.

Figure 8 - Solveur conditionnement

* *Comme améliorations possible nous avons pensé au fait d’organiser également la production en fonction de la tendance d’achat et des quantité commandés de bonbons.*

## Picking

Nous avons pu remarquer quelques inconvénients au fonctionnement actuel de ce service. Tout d’abord la répartition aléatoire des bonbons dans les gares, ce qui oblige ensuite aux différents cartons de passer devant l’intégralités des gares pour compléter sa commande.

* Nous avons pensé à dissocier les deux boucles de cette zone.
  + Ce qui nous permet de gagner 152 minutes sur une commande qui ne contient aucuns produits présents dans une boucles dans laquelle il ne sera pas allé.
* Ensuite nous avons prévu d’organiser un certain nombre de zones de picking par type de bonbons.

Actuellement nous avons :

* 8 couleurs
* 4 variantes
* 2 textures
* 3 conditionnements

Si on les multiplie les uns aux autres on obtient :

8 \* 4 \* 2 \* 3 = 144

On a 144 références uniques par type de Bonbon.

On vérifie :

* 144 \* 27 = 3888

On retrouve notre nombre total de références de Bonbons.

On sait que notre entrepôt à deux boucles de 20 gares chacune. Ces dernières possèdent chacune 100 zones de picking, dont une zone de picking peuvent comporte 7 cartons maximum.

On le ramène dont à 40 \* 1000 = 4000. On a 4000 zones de picking.

On ramène en %, on utilise donc 38.88/40 = 97% de la place

##### Optimisation de la répartition (équitable)

On prend 148 gares par Bonbon.

1.48 \* 27 = 39.96 soit 3996 gares utilisés.

On a 4 gares de vides.

Dans ce cas on utilise 39.96/40 = 99.9 % de la place disponibles.

**Mais ce n’est pas forcément la meilleure optimisation au niveau du temps ou de la gestion des emplacements en fonction des préférences.**

Cette solution ne sera pas retenue pour la répartition, car elle ferait perdre un nombre de zones de picking libres.

* La seconde manière d’organiser à laquelle nous avons pensé reprends la première quant au nombre de zone de picking par types de bonbons, soit 144 zones pour chacun.
  + Dans cette solution nous gardons les 112 zones de libres pour dupliquer les bonbons les plus populaires.
    - Cela permettra de gagner du temps dans le remplissage des cartons de commandes.

## Livraison

Nous avons décidé que nous devrions optimiser la façon dont les colis sont livres. Nous nous sommes donc concentrés sur les livraisons via Camion.

Nous sommes partis du postulat que l’entreprise étaient située à **Berlin, en Allemagne.**

### Création du Graphe

Nous avons décidé de représenter le service de livraison en utilisant un graphe liant les différents pays. La première chose était donc de construire le graphe, en mettant en tant que Sommets les pays frontaliers.

Afin de determiner les distances entre deux pays, nous avons considéré deux aspects :

- La capitale de chaque pays en tant que lieu de livraison

- La distance entre deux capitales es ten KM et les camions prennent la route la plus courte pour y arriver.

Sur l’image suivante, nous pouvons voir le graphe obtenu.

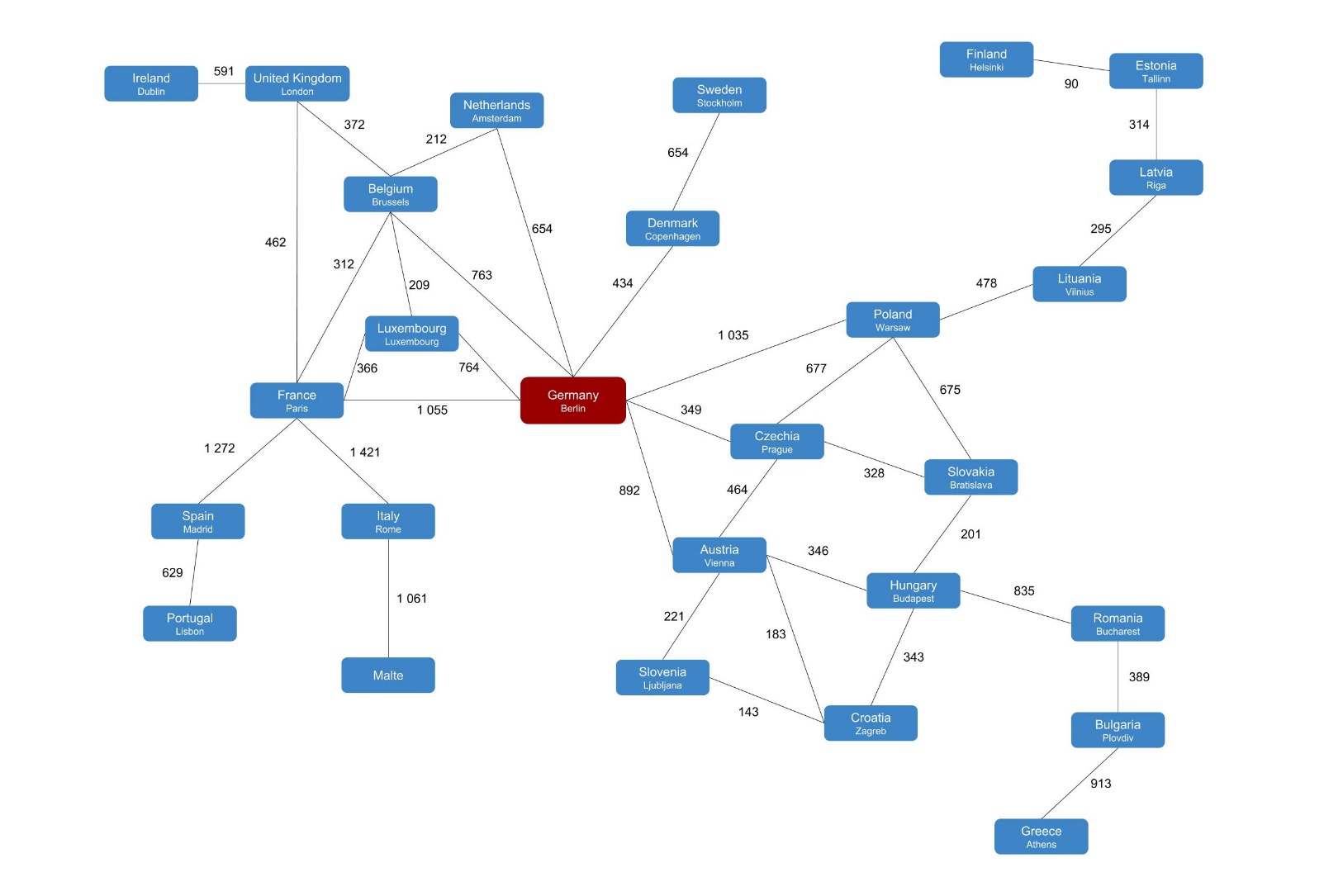


Figure - Graphe basique

### Algorithme de Dijkstra

Maintenant, nous voulons obtenir le chemin le plus court entre Berlin et les autres villes. Nous utilisons donc l’algorithme de Dijkstra.

L’algorithme consiste à trouver le chemin le plus court en voyageant de sommet en sommet, et en commençant par le sommet directement lié du point de départ.

* **Pour chaque** sommet, nous essayons de rejoindre les sommets directement liés et nous marquons la distance de voyage ( en provenance du point de départ ) associés à ces sommets liés, par exemple :
  + **SI** aucune distance n’est associée au sommet, **alors** on associe la distance en provenance du sommet
  + **SINON SI** une distance existe déjà, alors on compare les deux distances et garde la plus courte.
* On continue à suivre cet algorithme jusqu’à qu’il n’y ai plus de sommet restant.

Grace à l’algorithme, nous sommes garantis de trouver les chemins les plus courts.

Sur l’image suivante, nous avons le Graphe résolu via l’algorithme, et de fait les plus courts chemins :

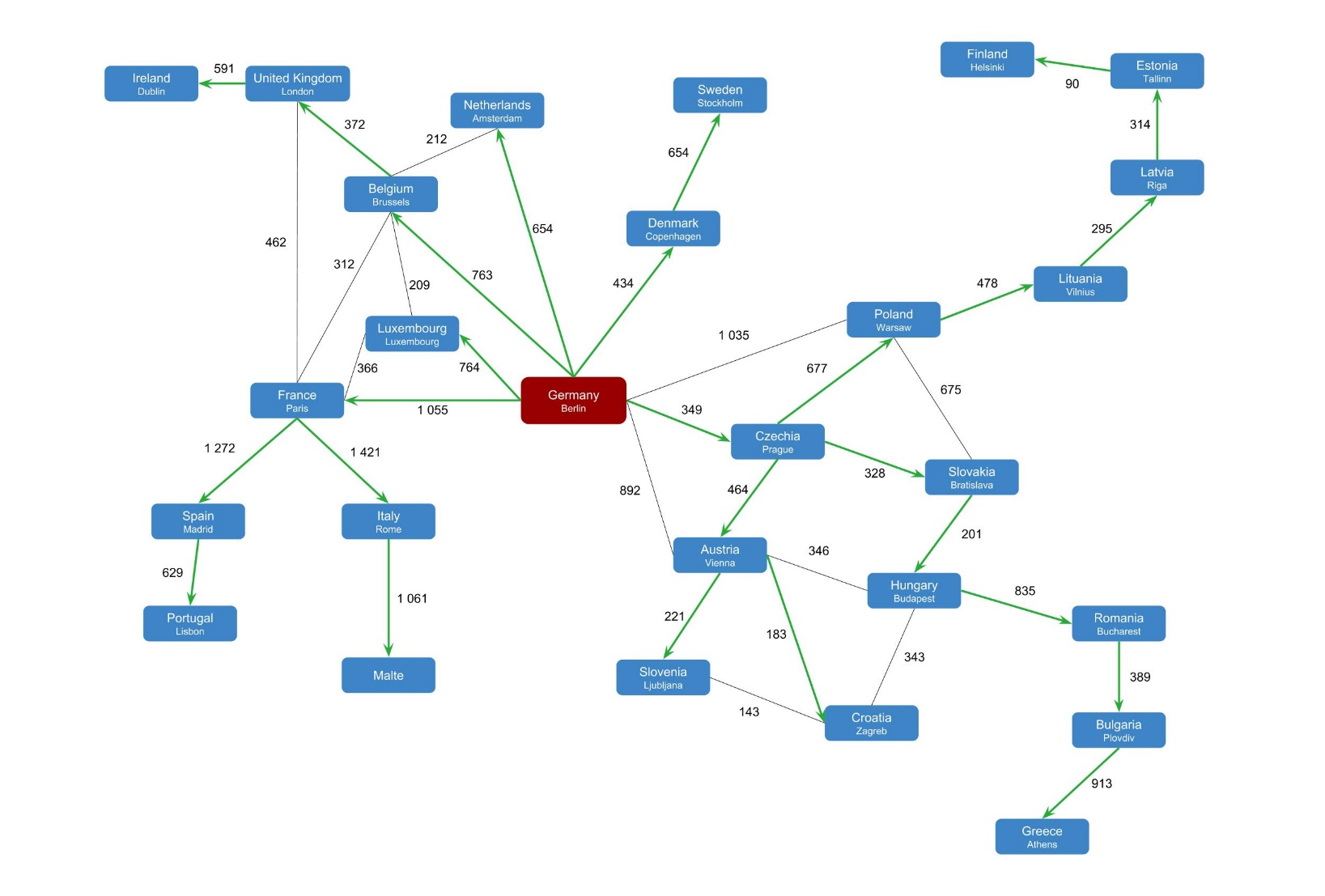


Figure - Graphe + courts chemins

### Optimisation

Malgré les résultats précédents, si nous envoyons un camion par chemin, nous devrions envoyer 10 camions pour tout livrer

Ces camions voyageront comme suit :

* T1 (Malte) = 1 055 + 1 421 + 1 061 = 3 537 km
* T2 (Portugal) = 1 055 + 1 272 + 629 = 2 956 km
* T3 (Luxembourg) = 764 km
* T4 (Ireland) = 763 + 372 + 591 = 1 726 km
* T5 (Netherlands) = 654 km
* T6 (Sweden) = 434 + 654 = 1 088 km
* T7 (Finland) = 349 + 677 + 478 + 295 + 314 + 90 = 2 203 km
* T8 (Greece) = 349 + 328 + 201 + 835 + 389 + 213 = 2 315 km
* T9 (Croatia) = 349 + 464 + 183 = 996 km
* T10 (Slovenia) = 349 + 464 + 221 = 1 034 km

Total = 3 536 + 2 956 + 764 + 1 726 + 654 + 1 088 + 2 203 + 2 315 + 996 + 1 034 = 17 272 km

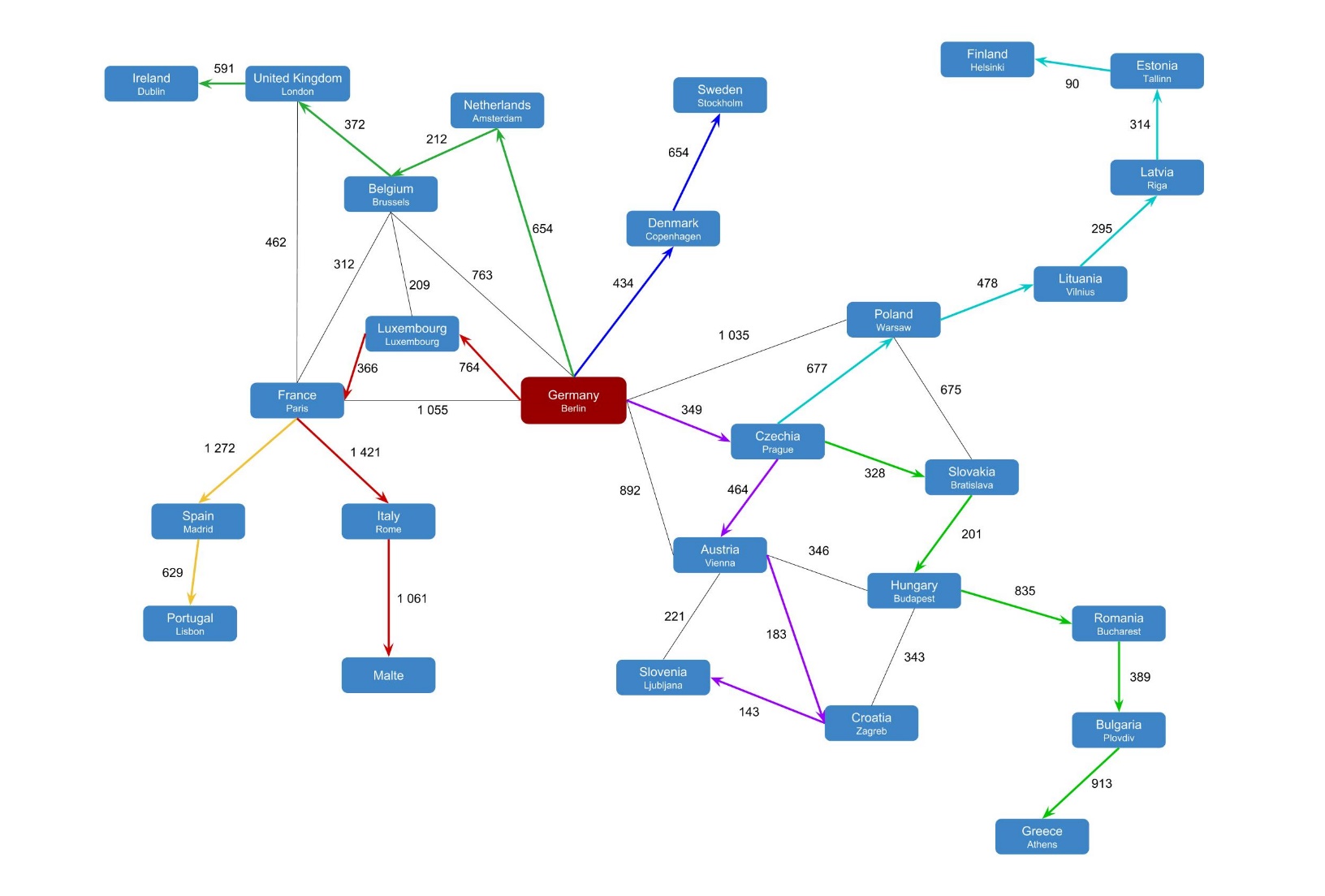
En dessous, les voyages optimisés pour chaque camion, avec le moins de camion et le moins de chemins possibles. 

Figure - Graphe optimisé

Les chemins ont été combines afin d’optimiser les distances parcourues. Dorénavant, nous pouvons réaliser les livraisons avec uniquement 7 camions de manière optimale :

* T1 (Malte) = 764 + 366 + 1 421 + 1 061 = 3 612 (+ 75)
* T2 (Portugal) = 764 + 366 + 1 272 + 629 = 3 031 (+ 75)
* ~~T3 (Luxembourg)~~ = 0 (- 764)
* T4 (Ireland) = 654 + 212 + 372 + 591 = 1 829 (+ 103)
* ~~T5 (Netherlands)~~ = 0 (- 654)
* T6 (Sweden) = 434 + 654 = 1 088 (- 0)
* T7 (Finland) = 349 + 677 + 478 + 295 + 314 + 90 = 2 203 (- 0)
* T8 (Greece) = 349 + 328 + 201 + 835 + 389 + 213 = 2 315 (- 0)
* ~~T9 (Croatia)~~ = 0 (- 996)
* T10 (Slovenia) = 349 + 464 + 183 + 143 = 1 139 (+ 105)

Total = 3 612 + 3 031 + 1 829 + 1 088 + 2 203 + 2 315 + 1 139 = 15 216 km (-2 056)

Grace à cette optimisation, les camions voyagent 2056 km moins, en livrant les mêmes pays.

### Résultat comparatif

Avant cela, les camions voyageaient par zones et en fonction des commandes, sans vraiment être optimisé.

Maintenant, il est garanti que les distances parcourues pour chacun des 7 camions sont optimisées et les plus courtes possible.

### Gestion des stocks

Nous avons remarqué également que les stocks n’étaient pas très bien gérés. Nous sommes partis de postulat selon lequel le stock est réapprovisionné selon un cycle pour pouvoir continuer le projet.

Grâce aux optimisations de production dans le service correspondant nous pouvons ainsi calculer la matière utilisée et anticiper celle qui le sera avec les commandes à venir, mais aussi grâce aux différentes tendances que nous pourront souligner avec les Dashboard. Tous ces indicateurs nous permettrons d’anticiper l’utilisation des composants.

### PlanningS

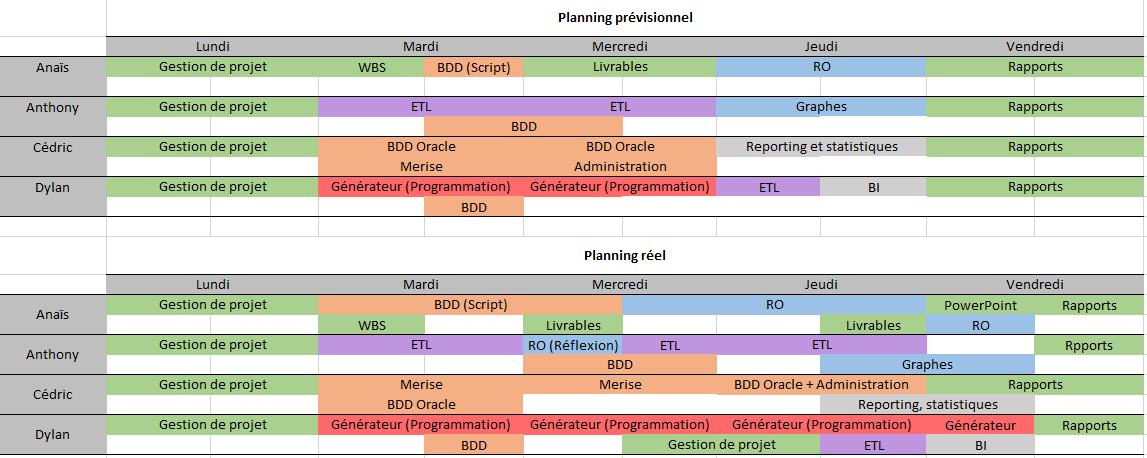


Figure 12 - Plannings

Chaque journée était débutée par un point sur ce qui a été fait la veille, ce qu’il reste à faire, et ce qu’il allait être fait le jour arrivant. De plus, chaque soir était clôturé par un point de ce qui allait être fait durant la soirée chez soi.

# Conclusion

En se basant sur les prérequis ainsi que sur les attendus, nous avons réussi à remplir parfaitement ceux-ci, et nous proposons de ce fait un projet terminé et avec peu d’écart. Nous aurions voulu développer une IHM pour le programme Python, mais par manque de temps, nous avons laissé la branche comme telle dans le projet github. De plus, nous avons pensé à différents axes d’améliorations possibles, et nous les avons écrits précédemment, sans pour autant les appliquer car nous manquions soit de temps, soit de données.

Ce projet a pu nous apporter un apport en connaissance important concernant la jonction entre différentes bases de données, mais également concernant le développement en Python d’une application. De plus, la méthodologie que nous avons utilisé durant tout ce projet nous a forcé à être rigoureux et à se tenir aux tâches que nous avions.