

# RAPPORT PROJET LONG TOB

## RÉALISATION D'UN GESTIONNAIRE DE TICKET DE CAISSE



# TABLE DES MATIÈRES

- I) SUJET
- II) PRINCIPALES FONCTIONNALITÉS
- III) DÉCOUPAGE DE L'APPLICATION
- IV) DIAGRAMMES DE CLASSE
- V) CHOIX DE CONCEPTION ET DE RÉALISATION
- VI) DIFFICULTÉS ET SOLUTIONS
- VII) ORGANISATION DE L'ÉQUIPE ET MÉTHODES AGILES

## I) NOTRE SUJET

### CONTEXTE DE L'ÉTUDE :

En France, 12,5 milliards de tickets de caisse sont imprimés chaque année, soit l'équivalent de 150000 tonnes de papier. Dans le monde, on compte la mobilisation de 25 millions d'arbres, 18 milliards de litres d'eau et 22 millions de barils de pétrole pour leur production. Au vu des enjeux environnementaux non-négligeables que les tickets représentent, il semble intéressant de les numériser. Au-delà de l'enjeu écologique on retrouve un réel avantage pour les clients : la numérisation évitent la perte évite l'encombrement et facilitent l'organisation.

Notre application TRICKET a pour objectif de stocker intuitivement les tickets de caisses des consommateurs et d'en extraire les données.

Pour: public concerné par le produit	Pour les consommateurs
Qui souhaitent: besoin des cibles	qui souhaitent suivre leurs dépenses
Notre produit est: ce qu'est le produit	un gestionnaire de ticket de caisse
Qui: le bénéfice majeur, l'utilité de la solution	centraliser des tickets de caisse numériques
A la différence de: pratique actuelle, concurrence	à la différence des tickets de caisse au format papier
Permet de : éléments différenciables majeurs	Ne plus être encombré par des tickets au format papiers, réaliser des bilans mensuels des achats.

*Figure 1: Pitch elevator de l'application*

## II) PRINCIPALES FONCTIONNALITÉS :

### 1) PRÉSENTATION :

Pour atteindre nos objectifs nous avons établis plusieurs fonctionnalités. Ces dernières sont illustrés dans le backlog produit ci-dessous.

(Nous l'avons découpé pour qu'il puisse être lisible depuis le rapport, mais nous vous invitons à cliquer sur le lien suivant pour une visualisation d'ensemble : )

On y retrouve les thème (en vert), les feature (en rose), et les user-stories (en blanc).

D : représente la difficulté associée à la fonctionnalité.

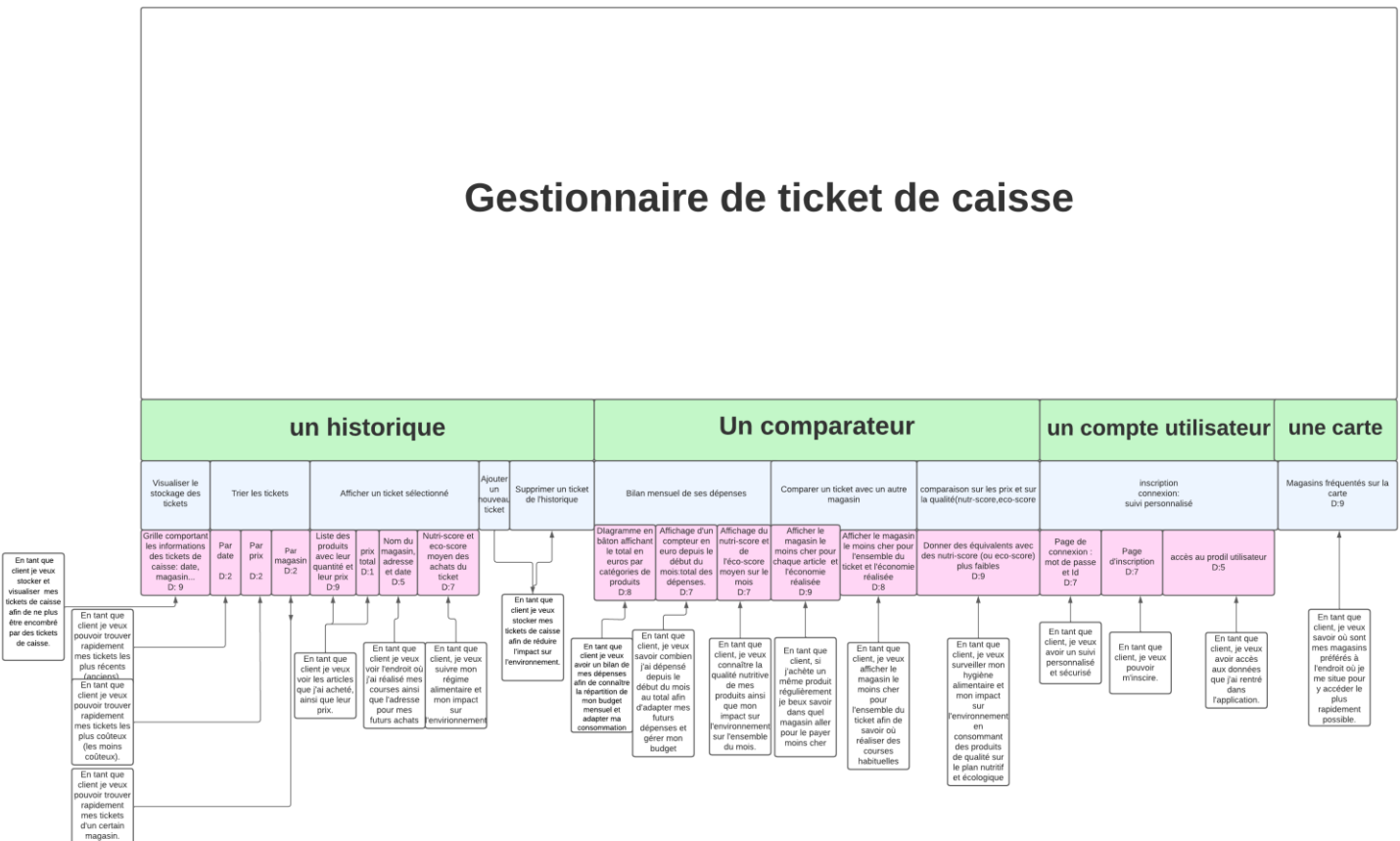


Figure 2: Vision d'ensemble du backlog produit

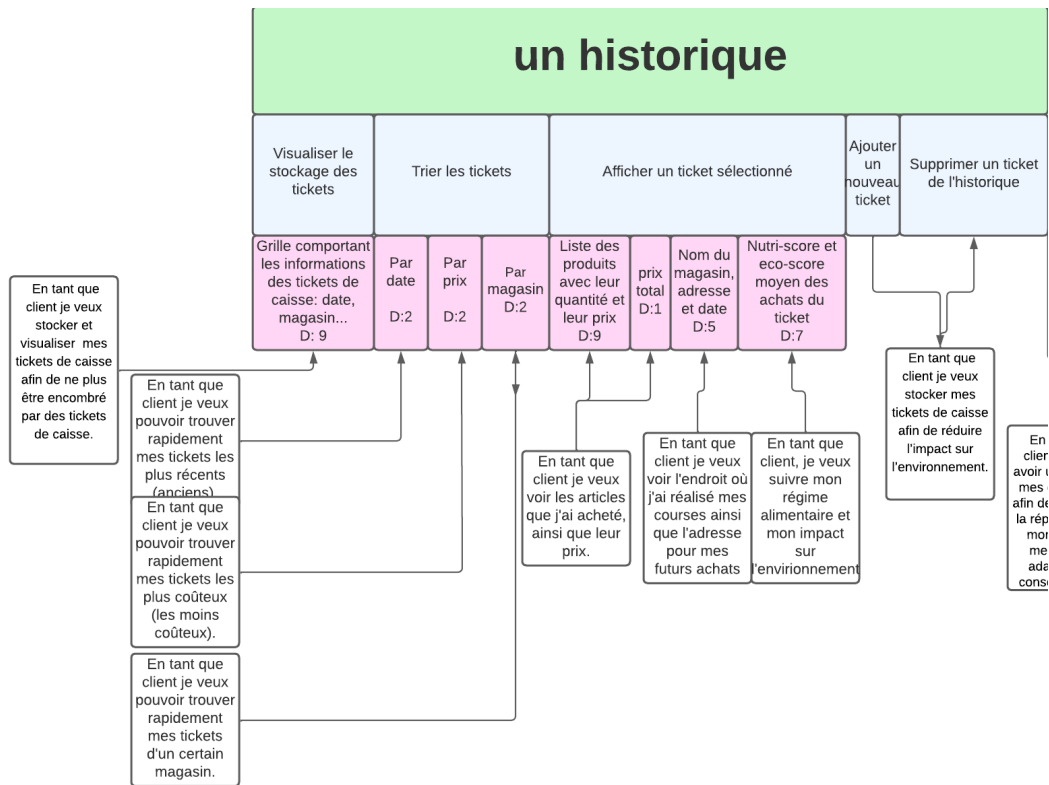


Figure 3: zoom sur la partie historique

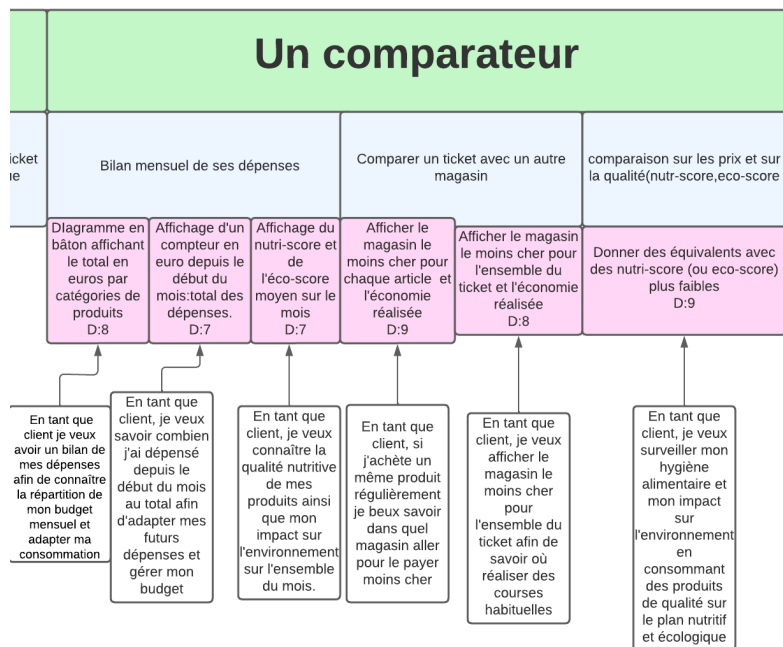


Figure 4: zoom sur la partie bilan

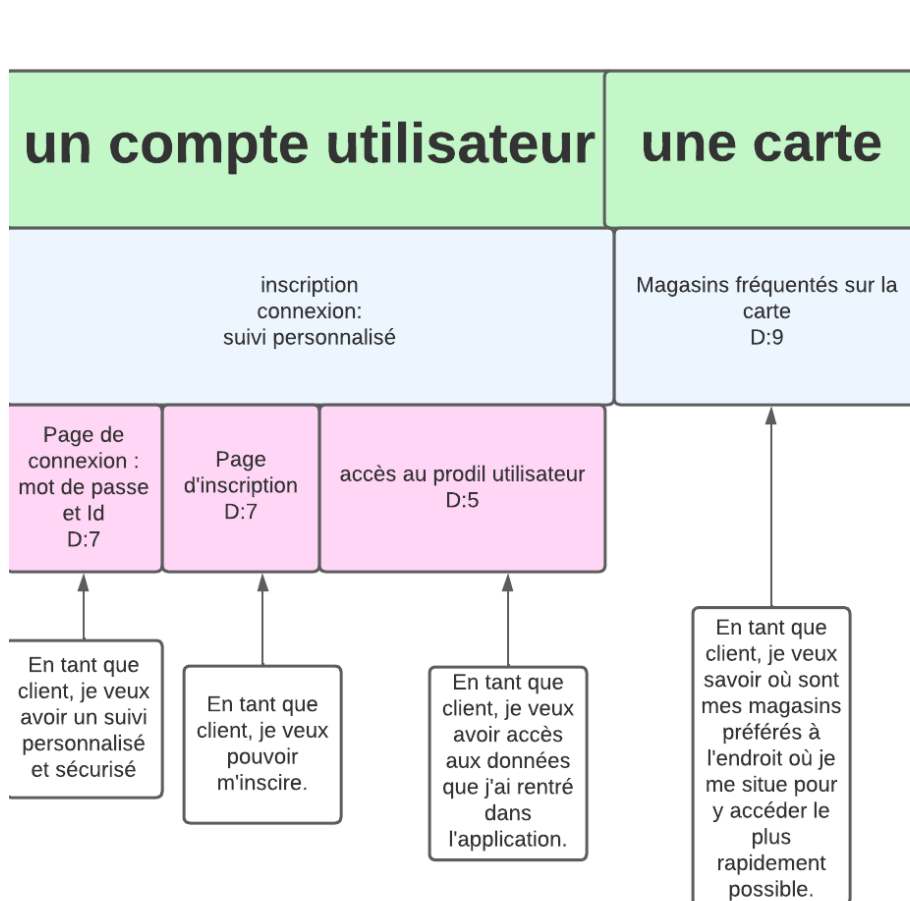


Figure 5: zoom sur les parties utilisateur et map

## 2) ETATS DE RÉALISATION DES FONCTIONNALITÉS

### PARTIE HISTORIQUE :

- Interface graphique de l'historique : réalisé
- Ajout d'un nouveau ticket dans l'historique : réalisé
- Trier les tickets par date : réalisé
- Trier les tickets par prix croissant : réalisé
- Trier les tickets par prix décroissant : réalisé
- Afficher un ticket : réalisé
- Afficher un ticket de comparaison des prix pour un ticket sélectionné : réalisé
- Affichage du nutri-score moyen d'un ticket de caisse : réalisé

- Affichage de l'éco-score moyen pour un ticket de caisse : réalisé
- Equivalence de produits avec un meilleur nutri-score : non réalisé

#### PAGE DE BILAN :

- Diagramme en bâton affichant le total en euros par catégorie : réalisé
- Compteur en euro : non réalisé

#### INTERFACE DE CONNEXION :

- Page d'inscription reliée à une base de données : réalisé
- Page de connexion reliée à une base de données : réalisée
- Interface graphique correspondante : réalisée
- Page montrant les informations de l'utilisateur : réalisée
- Possibilité de deconnexion : réalisée

#### PAGE DE MAP :

- Non réalisée : problème avec les API

### III) DÉCOUPAGE DE L'APPLICATION :

Nous avons réalisé trois packages : un pour la connexion, un autre pour la gestion des tickets de caisse ainsi que des bilans, et un dernier pour le front-end. Cependant, nous avons rencontré des conflits entre ces packages et nous avons finalement abandonné. Nous avons travaillé avec un unique package pour ce projet.

Ce ne fût pas contraignant puisque nous n'avions pas un très grand nombre de classe. Cependant, nous sommes conscients que pour une application plus volumineuse nous aurions dû résoudre ce point.

Découpage visuel de l'application réalisé en amont du code avec le logiciel Canva :

Page d'accueil

“Tickets de caisse”

Connexion

Inscription

Page de connexion

Connexion :

Nom d'utilisateur

...

Mot de passe

...

Valider

Page principale

←

Déconnexion

“Tickets de caisse”

Paramètres

Ajouter un ticket

🕒

Historique




🔍

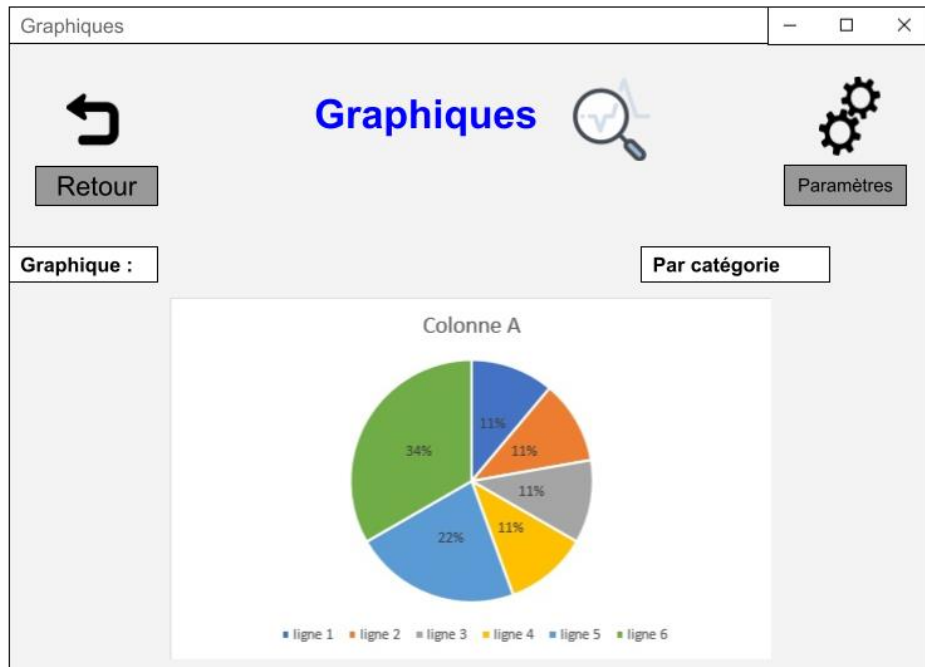
Graphiques

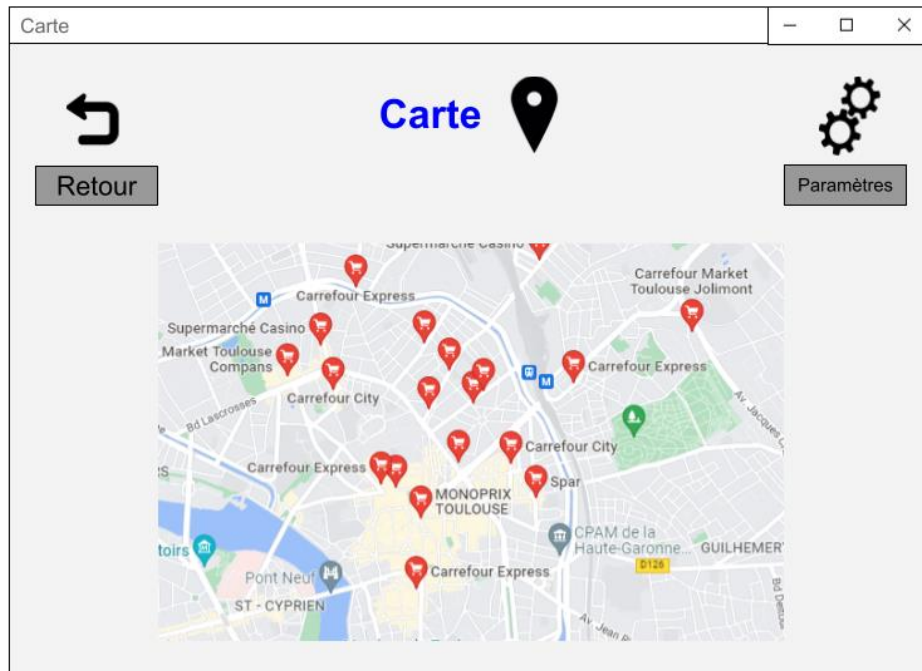
📍

Carte



Historique		— □ ×
<div> <div>  <div>Retour</div> </div> <div> <div>Historique</div>  </div> <div>  <div>Paramètres</div> </div> </div>		
Trier par :		Date
Ticket du 05/04/22 - Carrefour		
Ticket du 01/04/22 - Monoprix		
Ticket du 25/03/22 - UGC		
Ticket du 25/03/22 - Lidl		
Ticket du 22/03/22 - Carrefour		
Ticket du 12/03/22 - Mcd		
Ticket du 04/03/22 - Lidl		
Ticket du 01/03/22 - Carrefour		
Ticket du 19/02/22 - Monoprix		
Ticket du 15/02/22 - Lidl		





#### IV) DIAGRAMMES DE CLASSE :

(LE DIAGRAMME SERA PUBLIÉ DANS UNE PROCHAINE VERSION DE CE RAPPORT : NOUS AVONS RENCONTRÉ UN PROBLÈME AVEC LE LOGICIEL DE RÉALISATION).

#### V) CHOIX DE CONCEPTION

- Le premier choix que nous avons fait concerne les tickets de caisses. En effet, l'application idéalement devrait permettre à l'utilisateur de ne plus avoir de tickets de caisses en papiers. Une solution réelle est alors d'utiliser un système de QR Code personnalisé présenté aux commerçants qui permet d'identifier le client et de directement lui envoyer son ticket. Cependant, l'application est réalisée sur un ordinateur, donc ce système n'est pas réalisable à notre niveau. La solution a été la simulation numérique : nous avons créé une base de données contenant des produits de supermarchés (dans diverses catégories) et un algorithme qui à partir de cette base va créer un ticket de caisse numérique aléatoirement : cela permet de simuler un passage en caisse d'une personne lambda.
- Pour l'utilisation des bases de données nous avons utilisé la bibliothèque POI Apache. Nous avons étudié l'utilisation de SQL et de Maven pour gérer ce point, ce qui n'a pas abouti.

- Pour la création de la base de données nous avons étudié des produits dans diverses domaines : nourriture, hygiène, bureau,..Dans des magasins différents : le but étant de représenter des courses du quotidien. Nous avons pris 75 articles dans 5 enseignes différentes : exactement les mêmes articles avec la même quantité pour que la comparaison puisse s'établir.
- Pour l'interface de connexion : nous avons également décidé de travailler avec des bases de données et avec la bibliothèque POI APACHE avec des bases de données Excel
- Pour essayer de "factoriser le code" au maximum, et ne pas refaire des pages swing intégralement pour chacune des pages du projet, nous avons fait le choix de commencer par créer une classe PageBase, qui permet d'avoir une page initialisée avec les différents boutons communs à toutes les pages (bouton de retour, bouton paramètres). Il permet aussi d'avoir une gestion de l'historique de ticket implémentée pour chaque page sans avoir à rien redéfinir (vu qu'il faut garder l'historique sur chaque page, pour ne pas le perdre/le réinitialiser sans faire exprès).
- Les classes InterfacePrincipale, Connexion2, ConnexionErreur, Inscription2, InscriptionErreur, InscriptionSucces et pagePrincipale2 utilisent PageBase pour afficher les boutons communs. Nous avons ensuite utilisé des éléments de Swing pour pouvoir spécifier chaque page en ajoutant des Jtext field, des boutons et des Jlabels. Puis, nous avons adapté les classes connexion\_backend et Inscription\_backend pour faire le lien entre la partie frontend créée précédemment.
- 

## VI) DIFFICULTÉS SOLUTIONS

- La première difficulté technique était l'utilisation de la bibliothèque POI Apache, cela nous a demandé beaucoup de temps puisque les .jar à importer ne fonctionnaient pas : il n'y a qu'une version qui a fonctionné pour notre projet. La solution a été de tester les versions jusqu'à trouver la bonne.
- Nous avons également eu un problème lors de l'ajout d'un utilisateur dans la base de données : il écrasait l'utilisateur qui s'était inscrit précédemment en écrivant le nouveau sur la même ligne. La solution a été de remplir initialement les cases de la base de données par des points.
- Nous n'avons pas réalisé la fonctionnalité qui consistait à trouver des produits équivalents avec des meilleurs nutri-score (Eco-score), puisque notre base de données ne contenait pas assez d'éléments. En effet, pour que la comparaison soit efficace il faut au moins 1 produits quasiment identiques à celui que l'on veut comparer et avec un meilleur nutri-score. Cela aurait été possible, mais nous aurions dû sacrifier plus de temps à la base de données. Nous avons choisi de faire d'autres fonctionnalités. (Nous avons tout de même

essayé d'avoir les bases de données des magasins directement grâce aux API, sans résultat).

- L'édition de la base de données contenant les utilisateurs ainsi que leurs informations a demandé beaucoup de temps à pour se documenter et un problème a persisté. En effet, l'inscription d'un nouvel utilisateur était censée ajouter une nouvelle ligne sur la bdd avec les informations de cet utilisateur. Malheureusement, le programme final permet d'écraser la dernière ligne non vide et d'y écrire les informations du nouvel utilisateur. Les tests de Connexion/Inscription ont déjà été faits sur les premiers utilisateurs inscrits dans la base de données.
- Selon la page où l'on se trouve, certains boutons n'ont pas le même comportement, il a fallu filtrer dans le constructeur de PageBase les "PageBase" initialisées selon leur titre, afin d'appeler la méthode de SetUp du bouton retour adaptée à la situation.
- Un autre problème était le fait de devoir réinitialiser l'historique du ticket au début de l'application (pour simuler un vrai utilisateur avec ses tickets), à travers les mêmes appels de Pages que lorsque l'on devait passer l'historique d'une page à l'autre. Ce problème a été éventuellement résolu à travers une manipulation efficace des constructeurs des différentes pages.
- Notre projet repose davantage sur du back-end que sur le front-end. Les fonctionnalités ont demandé beaucoup de temps à être élaboré car nous devons apprendre des nouvelles notions : les comprendre et les maîtriser. Notre interface graphique mérite d'être améliorée.

## VII) ORGANISATION DE L'ÉQUIPE ET MISE EN ŒUVRE DES MÉTHODES AGILES

Nous avons à 8 réfléchis au sujet : chacun a proposé un PDF sur son idée puis nous avons réalisé un vote anonyme pour décider du sujet final. Nous avons organisé trois réunions afin de cadrer le sujet et de décider des fonctionnalités.

Une fois le sujet bien cadré nous avons établi un pitch elevator pour s'assurer du besoin et de la cible.

Ensuite, nous avons profité des séances de méthodes agiles pour établir un backlog produit précis contenant : les thèmes, les features et des user-stories. Le but était de définir les attentes de notre projet ainsi que l'ordre d'importance des fonctionnalités (pour prioriser).

Nous avons travaillé sous forme d'itération (sprint scrum) en fixant à chaque début de sprint des objectifs. A la fin de chaque itération, chacun rédigeait un rapport personnel, un manuel utilisateur et un rapport plus général.

Nous avons utilisé l'application scrum time afin que chacun exprime son avis concernant la difficulté des fonctionnalités

Nous avons tous travaillé sur la construction de la base de données : 75 articles par enseignes avec plusieurs informations sur les articles. Nous avons également réalisé le design de l'application sur le logiciel Canva pour cibler l'interface graphique.

Nous avons ensuite codé : Clémentine s'est occupée du back-end : création des tickets, gestion des tickets, gestion de l'historique, comparaison entre les magasins, nutri-score (eco-score) moyen, graphique en bâton. Meyssan a travaillé avec Yessine sur l'interface de connexion (back-end et front-end). Louis a travaillé avec Yessine sur la partie Front-end. Et charly a étudié la fonctionnalité de la carte.

Nadesan a travaillé avec Ruudy sur l'interface graphique des informations de l'utilisateur.

Pour la fin de la dernière itération nous avons travaillé ensemble pour mettre en commun nos classes.

Pour cela nous avons fonctionné par itération : chaque semaine nous organisons une réunion à l'école pour montrer les avancés et réfléchir ensemble sur la suite. Chaque fin d'itération nous avons rédigé un rapport général, des rapports individuels et un manuel utilisateur.

#### VIII) Conclusion et sources :

Pour conclure, ce projet a été très intéressant à réaliser pour l'ensemble de l'équipe.

Premièrement il a permis à ceux qui ont codé de développer leurs connaissances en java par la maîtrise de nouveau concept tel que la maniabilité des bases de données, mais aussi l'utilisation de nouvelle fonction. Nous avons appris à chercher des solutions à nos problèmes : via des ouvrages, des cours annexes, les documentations sur les classes JAVA,..

Nous avons aussi appris à collaborer grâce aux méthodes agiles. Cela nous a permis de réaliser que nous devons définir précisément chaque fonctionnalité mais aussi les trier par importances : cela nous a permis de réaliser les fonctionnalités les plus utiles.