Le groupe étant composé de 10 étudiants (Alexandre A., Alexandre B., Ismail, Raphaël, Philippe, Lina, Brenno, Wassim, Romain, Nicolas) de l'École Polytechnique de Bruxelles. La vélocité du groupe est de 28h par itération.

L'objectif de ce document est de décrire le suivi de planification du projet. Il reprend notamment les explications des fonctionnalités implémentées et les méthodes utilisées pour contourner les différentes difficultés techniques. Il reprend également les différents choix pris lors de la conception du projet.

Gestion de projet

Afin de s'assurer que tout soit bien réalisé à la fin de l'itération, il a été décidé d'utiliser un Microsoft To-Do pour lister et s'assurer que les tâches et documents avaient bien été complétés.

Suivi de planification

Itération 1

Lors de la première itération, certaines fonctionnalités n'ont pas su être implémentées, principalement au niveau de l'affichage de l'application. Cela est dû à une mauvaise répartition des tâches et à un apprentissage qui s'est avéré plus complexe que prévu du JavaFX.

En effet, il y a eu beaucoup de problème au niveau de la structure même du projet qui ont été réglés grâce à l'utilisation de Maven. L'outil Maven a permis de faciliter la construction, la gestion des dépendances et le déploiement de l'application.

Afin de s'assurer que toutes les tâches soient réalisées lors de l'itération suivante, une réunion a été faite afin de s'assurer que l'équipe communique bien ses avancées et que les tâches soient définies de façon claire pour tous les membres de l'équipe.

Afin de faciliter et de gagner du temps lors de la programmation, le module Lombok a été implémenté (voir Lombok.md).

Itération 2

À partir de l'itération 2, après en avoir parlé en réunion, il a été décidé de définir 2 personnes par itération qui définissent les tâches et les écrivent dans le document. De la sorte, toutes les tâches sont clairement définies et il est facilement possible de communiquer entre binômes.

L'affichage des cartes étant une tâche principale, plusieurs autres tâches ont dû être dépendante de celle-ci, ralentissant le travail.

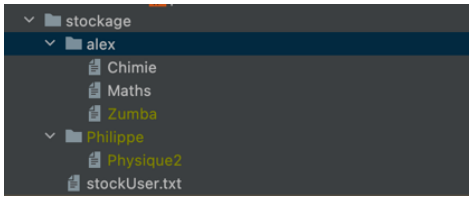
Il faudra dès lors assurer une meilleure communication et création des tâches.

L'histoire 5 n'a pas pu être terminée, les différents types de cartes sont créés, stockés et l'affichage fonctionne. Mais il n'est pas possible d'éditer et d'afficher les différents types de cartes sur l'application. Il a fallu trop de temps pour créer l'édition du paquet et l'ajout de cartes dans ce dernier. Dès lors il a été décidé de termine l'histoire 5 à l'itération 3 avec 3 points supplémentaires. C'est notamment pour cela que certaines fonctions ne sont pas utilisées dans l'affichage, elle seront donc utilisées dès le début de l'itération 3.

Choix de conception

Itération 1

L'utilisation des fichiers pour le stockage de données a été implémenté pour faciliter l'accès aux données. Les utilisateurs sont stockés dans "stockUser.txt" dans le dossier ''stockage''. Chaque utilisateur a un dossier personnel avec à l'intérieur une série de fichiers texte correspondant aux paquets.



Les couples utilisateurs / mot de passe sont structurés comme suit "utilisateur#motdepasse". C'est une solution d'encodage qui est très basique, facile à utiliser dans le code et qui n'a pas de contraintes sur la sécurité pour l'instant.

"stockUser.txt"

```
alex#pomme
philippe#orange
```

Les fichiers "paquets" sont structurés comme suit : la première ligne correspond au nom du paquet et les lignes d'après sont les questions et réponses, séparés par des "#".

"Physique2"

```
Physique2
#Qu'est ce que la divergence du rotationnel ?#Une bien belle chose
#PV ?#=NRT bien sûr !
```

La structure MVC n'a pas été implémentée et n'a pas été prise en compte lors du développement du code de l'itération 1. Par conséquent, des parties du code se sont mélangées ne suivant pas les principes du MVC. Ce qui rendait l'utilisation et l'assemblage des différentes parties du programme très complexe et non optimal.

Itération 2

Lors de l'itération 2, il a été décidé d'implémenter la structure MVC recommandée lors du cours théorique. Elle facilite grandement l'implémentation de nouvelles fenêtres et la lecture du code.

Pour l'application de l'histoire 8 (cartes textes à trous et QCM), il a été décidé dans un premier temps de ne pas faire d'héritage au niveau des cartes car cela allait compliquer fortement le stockage de ces dernières. La solution adoptée a été d'écrire dans les String *recto* et *verso* les questions et réponses sous des formats différents selon le type de la carte. Des fonctions permettant de décrypter ces formats ont été implémentées afin de faciliter l'affichage et l'utilisation de ces String *recto* et *verso.*

La gestion des exceptions n'étant pas implémentée de manière correcte, elle a dû être remaniée.

Il a été décidé que chaque paquet de cartes soit stockés suivant la structure suivante :

"Nom du paquet"

```
Nom du paquet
#Categorie1#Categorie2#...#CategorieN
```

```
#TypeDeCarte#Recto#Verso#Score
#TypeDeCarte#Recto#Verso#Score
#TypeDeCarte#Recto#Verso#Score
#TypeDeCarte#Recto#Verso#Score
```

Pour l'application de l'histoire 8 (cartes textes à trous et QCM), il a été décidé dans un premier temps de ne pas faire d'héritage au niveau des cartes car cela allait compliquer fortement le stockage de ces dernières. La solution adoptée a été d'écrire dans les String *recto* et *verso* les questions et réponses sous des formats différents selon le type de la carte. Des fonctions permettant de décrypter ces formats ont été implémentées afin de faciliter l'affichage et l'utilisation de ces String *recto* et *verso.*

Pour le format texte à trou (*tt)* : `#tt#débutdephrase§findephrase#réponse#Score`

Pour le format qcm (*qcm)* : `#qcm#question§choix1§choix2§choix3#réponse#Score`