Document
INFO-F307 - Génie logiciel et gestion de projets

Introduction

Ce document détaille tous les choix de conception et d'utilisation d'outils pour mener à bien le projet demandé par le client. Lors de cette deuxième itération, 4 histoires ont été implémentées comme prévu lors de la réunion avec le client. Les 4 histoires sont les suivantes : les réponses de différents types (8), la connexion à un serveur (9), l'import/export de jeux de cartes (15) et le store de jeux de cartes (11) . Il y est aussi indiqué les quelques designs patterns utilisés pour coder efficacement en orienté objet. De plus, le code suit le pattern d'architecture MVC. Pour finir, les difficultés rencontrées sont également mentionnées.

Histoire 8

Difficultés rencontrées

Choix de conception

Structure de données : Nous avons décidé de faire une sous classe de la classe Card pour nos questions à choix multiples. Cette nouvelle classe a un nouveau constructeur qui prend 3 fausses réponses et une bonne réponse. Cette nouvelle classe ressemble assez à la classe Card pour qu'elle puisse être utilisée par les algorithmes déjà implémentés.

Création de cartes multiples : Quand l'utilisateur choisit de créer une nouvelle carte, une nouvelle fenêtre apparaît, lui demandant quel type de carte veut-il créer. S'il choisit une carte à choix multiple alors la nouvelle fenêtre pour la création de cartes à choix multiples apparaît. L'utilisateur doit écrire une bonne réponse ainsi que 3 fausses réponses.

Nous avons codés une classe qui est le contrôleur de vue StudyMultipleChoicesViewController qui hérite de la classe StudyViewController. En effet, nous avons jugé cet héritage cohérent puisque certaines méthodes du StudyViewController sont utilisées dans le contrôleur de vue du menu d'étude à choix multiple.

Histoire 10

Difficultés rencontrées

Pour différencier les utilisateurs et savoir à qui le message doit être envoyé, nous avons attribué un ID unique à chaque session ouverte par un utilisateur. Ensuite, lorsqu'un utilisateur demande au serveur sa liste de deck pour remplir son store, il reçoit des objets Deck complets au lieu de recevoir des ID de decks ce qui risque d'alourdir la charge réseau en fonction du nombre de decks stockés dans le store.

Choix de conception

Les paquets échangés entre le serveur et le client sont sous format json pour faciliter la lecture et le debuggage. Ces derniers, sont construits avec un design pattern builder director pour éviter les constructeurs à rallonge. Les types des paquets échangés sont énumérés dans une structure enumeration appelée Type.

Ensuite, du côté serveur, à la connexion d'un nouveau client au ServerSocket un thread est lancé pour le gérer via la classe ClientHandler. Cette dernière possède une liste static permettant à un ClientHandler lié à un certain client d'avoir connaissance de tous les autres clients connectés à un même instant.

Histoire 11

Difficultés rencontrées

Difficultés DataBase store : Gérer la connexion entre les 2 databases DataBase locale et la DataBase server parce que la connexion était partagée et les 2 databases étaient fermées comme moyen de sécurité. Pour pouvoir éviter cette connection partagée nous avons utilisé **volatile static** qui s'assure du fait qu'une connection une fois ouverte est bien indépendante et pas partagée entre les différentes sections du code.

Choix de conception

DataBase store: Nous avons créé une 2ème database qui est la database pour le serveur pour stocker les decks avec les cartes sur cette database où tous les clients connectés sur le réseau peuvent accéder à cette database pour pouvoir télécharger et upload des decks. Pour gérer le switch entre ces 2 databases nous avons introduit juste un booléen dans chaque méthode de la database qui fait les mêmes queries dans la database spécifié. Pour que tout aille bien ensemble, car il y avait trop de dépendances, on a juste séparé en plusieurs classes db + une façade.

Contrôleur store : Nous avons 2 fenêtres, une pour upload les decks et l'autre pour download. On utilise simplement la database Server et la database locale en envoyant des paquets à travers le serveur.

Histoire 15

Choix de conception

L’import et l’export est fait au moyen de json converti avec la librairie Gson. Cela permet de convertir facilement toutes les informations des cartes et des decks.

Pour conclure l’itération 2, nous avons implémenté toutes les fonctionnalités des 4 histoires.