DOCUMENT DE DÉVELOPPEMENT

E-Scape Game

PROJET DÉVELOPPÉ PAR L'ÉQUIPE Q8

29 janvier 2021

Table des matières

1 Domaine d'application

1.1 Objectif du système

L'objectif de ce projet est de concevoir un logiciel de gestion d'e-escape game en réseau local destiné à la commercialisation par l'entreprise E-Ntertainement, entreprise de services en ligne dans le domaine du divertissement.

Cette application permet au gérant d'un E-escape game de créer des jeux et de gérer les parties jouées en direct par les joueurs.

FIGURE 1 – figure

1.2 Interfaces

1.2.1 INTERFACE MAÎTRE DE JEU (GÉRANT)

Le Maître de jeu possède 4 interfaces qui lui sont propres :

1. **MainMenu :** Menu principal où le maître de jeu choisit d'accéder à la partie création ou à la gestion.
2. **GameManagement :** Interface permettant d'accéder aux jeux, de les ajouter à une salle ou de les supprimer.
3. **RoomManagement :** Interface permettant de voir les parties en cours, d'accéder à la liste des jeux ou de rejoindre la salle de suivi des joueurs.
4. **PlayerManagement :** Interface de suivi par le gérant de la partie en cours chez un joueur.

1.2.2 INTERFACE JOUEUR

Le joueur possède quand à lui 4 interfaces qui lui sont propres :

1. **RoomAccess :** Menu d'accès aux différentes salles de jeux côté joueur.
2. **CurrentGame :** Interface de jeu.
3. **ChooseRanking :** Interface de choix du jeu dont on veut obtenir le classement.

2 Documents de référence

Les documents suivants sont à utiliser en référence à la lecture de ce document :

1. **Document de pilotage :** Il contient le descriptif du planning et des méthodes de développement.
2. **Document de tests :** Il contient les rapports des tests utilisateurs.
3. **Manuel utilisateur :** Le manuel utilisateur permet la prise en main du logiciel par les utilisateurs.

3 Outils de développement

3.1 Environnement et outils de développement

3.1.1 GITHUB

L'équipe pouvant travailler en parallèle, un logiciel de gestion de versions était nécessaire. GitHub étant un logiciel connu de l'équipe, sa place était toute faite dans notre processus de développement. Toutes les versions du projet sont stockées sur ce dépôt, ce qui a permis une plus grande transparence avec la MOA quant à l'avancée du projet.

3.1.2 JIRA

L'équipe travaillant avec une méthode agile, il a fallu mettre en place cette méthode ainsi qu'avoir un suivi de la progression du projet. Nous avons donc utilisé Jira afin de nous aider à organiser les sprints ainsi que le backlog. La MOA pouvait également avoir accès à ce Jira, toujours dans le but d'être transparent avec elle.

3.1.3 IntelliJ IDEA

Le développement du projet a été réalisé en grande partie avec IntelliJ IDEA. L'équipe utilisant régulièrement l'IDE, la prise en main fut rapide et l'équipe n'a pas perdu de temps à mettre en place l'environnement de travail.

3.1.4 Atom

Cependant certains de nos développeurs ont utilisé Atom par préférence personnelle ce qui ne causait pas de problème de comptabilité car la version de Java était la même.

3.1.5 Discord

La plateforme Discord a été utilisée pour faciliter la communication entre les membres de l'équipe. Lors des sessions de programmation l'équipe se retrouvait sur les canaux vocaux pour travailler et a utilisé l'option de partage d'écran pour se présenter les fonctionnalités au cours du développement.

3.2 Conventions de codage/nommage

Pour la convention de nommage nous avons utilisé CamelCase. Les fonctions sont majoritairement nommées en anglais.

4 Conception générale et architecture

4.1 Langages utilisés

— **SQL** pour les scripts de création, d'insertion, de mise à jour (...) sur la base de données.
— **Java JDK 8+** pour le développement de l'application générale.

4.2 Base de données

La base de données a été conçue au début du développement, elle sert à stocker toute les informations de l'application. Elle a été conçue de façon à pouvoir identifier chaque table par un id. Chaque table reliée avec une autre sur le schéma contient l'identifiant de cette dernière de manière à pouvoir relier les tables entre elles. Cette base de données est administrée via phpMyAdmin et hébergée chez AlwaysData. Les comptes sont stockés dans la table User et vont servir à l'authentification lors de la connexion. Les énigmes, jeux, salles et scores

seront stockés dans les tables homonymes et seront accessibles via des méthodes et des procédures.

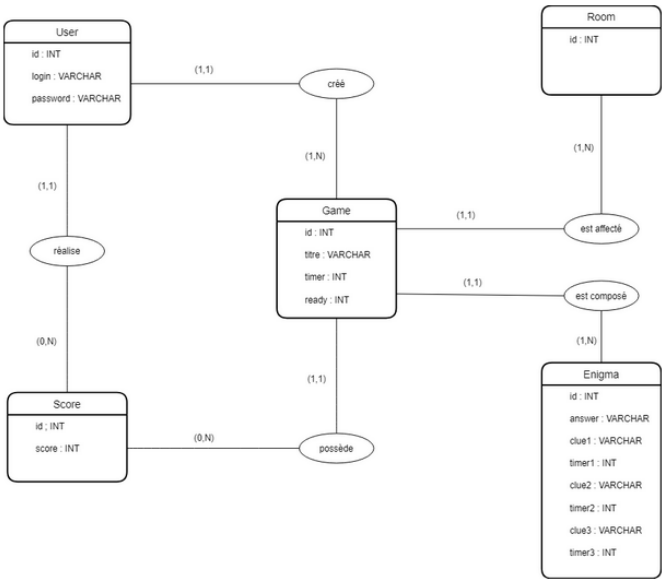


FIGURE 2 – Diagramme de classe de la base de données.

4.3 Librairies

Lors des communications, l'application était figée car les méthodes liées aux sockets étaient exécutées au premier plan, il a donc fallu trouver une solution pour faire exécuter ses fonctions en arrière plan. ExecutorService est une interface qui permet de gérer un service d'exécution de tâches qui seront exécutées soit dans un thread dédié soit dans le thread actuel. L'objectif est donc d'appeler les méthodes dans des threads dédiés de manière à ce que l'application puisse continuer de s'exécuter sans interruption.

Les relations entre l'application et la base de données devant être fiables, nous nous sommes donc tournés vers le framework de test unitaire JUnit dans le but de tester ces méthodes.

Pour pouvoir utiliser des méthodes en lien avec la base de données il a fallu installer des drivers compatibles. Le connecteur pour les bases de données Java, "MySQLConnector", a donc été utilisé afin d'avoir un package de driver compatible avec Java et avec notre base de données MySQL.

4.4 Organisation des fichiers et des classes

Le projet étant conséquent, il a fallu organiser les fichiers de manière à pouvoir se repérer facilement dans le projet. Les fichiers sont donc organisés comme suit : Les fichiers relatifs aux relations de la base de données dans le dossier “Database”. Les fichiers relatifs à la déclaration des objets manipulés dans le dossier “Model”. Les fichiers relatifs à la connexion entre le côté client ou le côté gérant dans le dossier “Sockets”. Les fichiers relatifs à toute la partie visuelle de l’application dans le dossier “View”. Il existe 2 sous dossiers dans ce dossier, ces derniers contiennent les déclarations des styles et des couleurs ainsi que les loaders qui permettent de gérer le chargement des images. Le fichier permettant le lancement de l’application dans le dossier “Launcher”.

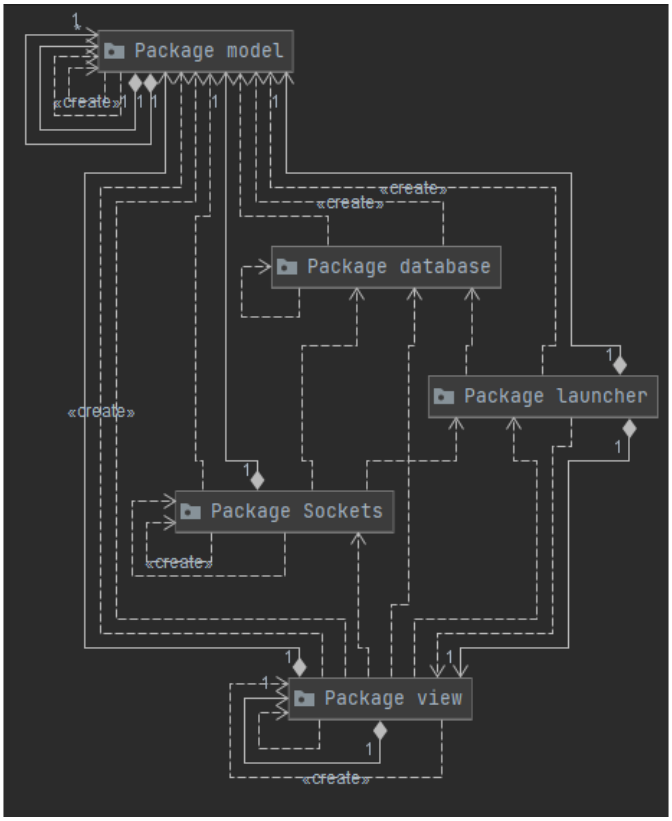


FIGURE 3 – Dépendances entre les packages

Ces dossiers sont appelés packages et sont importés dans les fichiers qui nécessitent un objet de ce package. Si par exemple dans un menu (qui est un visuel donc dans le dossier “View”) à besoin de créer un objet de type "Énigme" il faudra qu’il importe le package dans lequel l’objet "Énigme" est déclaré (ici il importera Model.Énigme).

4.5 Navigation entre les visuels

La navigation entre les visuels s'effectue à l'aide de boutons sur les différentes interfaces. Il est possible à tout moment de revenir à l'interface précédente. Une exception cependant, il est impossible pour un joueur de quitter la partie en cours.

4.6 Communication entre client et administrateur

4.6.1 Sockets

La communication entre client et administrateur s'effectue par l'intermédiaire de sockets. En effet lors de la connexion d'un client à l'application il va effectuer une requête vers l'administrateur pour savoir quel est l'administrateur et donc pouvoir savoir à quelle salle le client peut se connecter. Pour ce faire, un serveur de sockets va être lancé dès que l'administrateur va lui aussi se connecter. Ce serveur tourne en tâche de fond et n'influence donc pas le déroulement du programme côté administrateur. Quand un client voudra se connecter à une salle, l'administrateur recevra un pop-up lui demandant s' il souhaite accorder l'accès à la salle. Au moment où le client rentre dans la salle, il va lancer également un serveur de sockets de manière à ce que l'administrateur puisse lui transmettre des informations (indices ou messages). Toutes ces communications fonctionnent grâce à la transmission d'objets sérialisables. Les serveurs de sockets vont être mis en tâche de fond grâce aux interfaces Executor qui ont été détaillées plus tôt.

4.6.2 Threads

Les Threads ont été utilisés avec pour but de faire transiter les informations comme le timer ou les énigmes. On les préfère aux serveurs de sockets pour leur facilitée à récupérer les valeurs transitées.

4.7 Stockage des données

4.7.1 Préparation

Les données sont stockées dans une base de données (décrite précédemment), l'objectif étant que chaque donnée modifiée soit inscrite dans la bdd. La création et la suppression des données fonctionne de la même façon, en effet les modifications prennent effet immédiatement ce qui permet de garder une base de données à jour tout le temps. Le stockage de données est assuré par des

requêtes préparées qui permettent de gagner du temps sur des requêtes simples car il suffit de fournir les arguments sans avoir à reconstruire la requête s' il y a une multiple exécution.

4.7.2 SÉCURITÉ

Certaines données stockées (comme le mot de passe par exemple) ont besoin d'être chiffrées de manière à ce que la donnée reste sécurisée. Pour ce faire, la création d'une classe spéciale pour le hachage est nécessaire (on utilisera MD5 ici). Le mot de passe, qui est une chaîne de caractère, sera découpé en octets de manière à pouvoir être haché par notre classe précédemment créée. On convertira par la suite nos octets en chaîne de caractère pour que cette dernière soit stockée dans la base de données. Lors d'une connexion on ne déchiffre pas la chaîne par souci de sécurité. On répète donc la même opération que décrite précédemment pour comparer les deux chaînes hachées entre et autoriser/refuser la connexion.

5 Déploiement/Installation/Lancement

Cette partie vise à fournir un ensemble d'instruction permettant de diffuser le logiciel E-Scape Game.

5.1 Déploiement

Dans l'état d'avancement actuel, un déploiement à grande échelle n'est pas envisageable et nous préconisons donc une distribution à quelques clients en attendant une version plus aboutie permettant de l'étendre à une plus large clientèle. La version actuelle étant un artefact jar, nous vous conseillons de la distribuer numériquement accompagnée d'une clé administrateur.

L'équipe Q8 préconise d'utiliser des machines de moins de 7 ans sur un réseau contrôlé et administrable facilement. En effet, il est nécessaire que les postes soient sur le même réseau et que les ports réseaux utilisés par l'application soient autorisés dans la configuration. De plus il est nécessaire d'installer JAVA avec JRE 8 ou ultérieure pour pouvoir lancer le logiciel dans sa forme actuelle. Celui-ci peut être trouvé à l'adresse suivante :
https ://www.oracle.com/fr/java/technologies/javase-jre8-downloads.html.

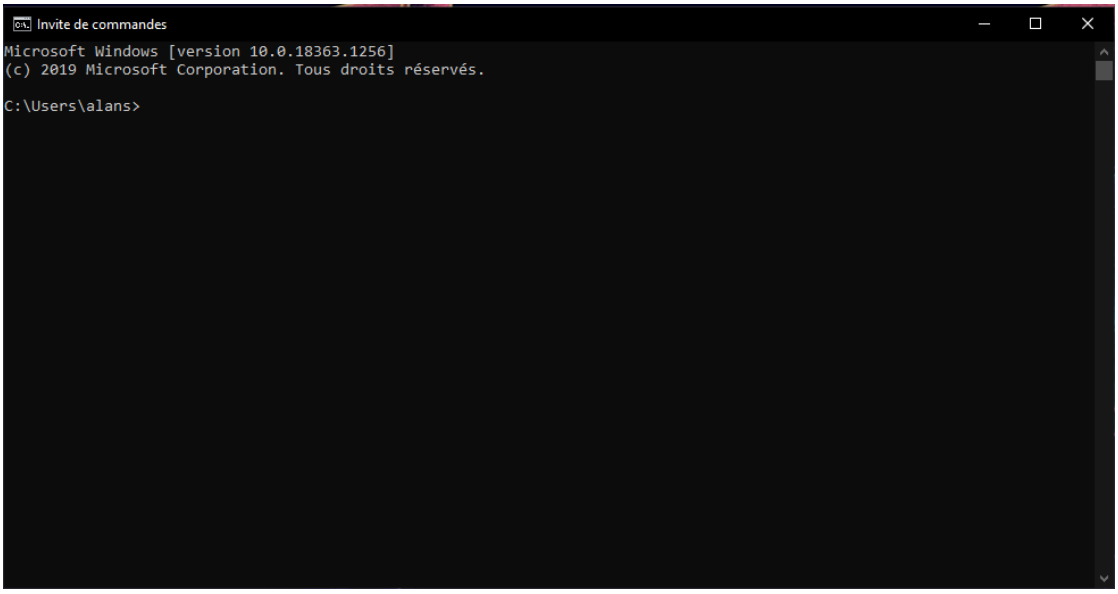
5.2 Installation

L'application est distribué sous la forme d'un fichier JAR qu'il suffira de télécharger. Ce dernier permet de lancer l'application.

5.3 Lancement

Une fois le fichier JAR téléchargé et enregistré dans les fichiers de l'utilisateur, il doit être exécuté par ligne de commande. Celles ci sont légèrement différentes pour une utilisation dite « maitre de jeu » et une utilisation dite « joueur ».

1- En tout premier lieu, il vous faudra ouvrir un invite de commande. Pour cela il vous suffit de taper « cmd » dans la barre de recherche Windows. La fenêtre suivante s'ouvre alors :

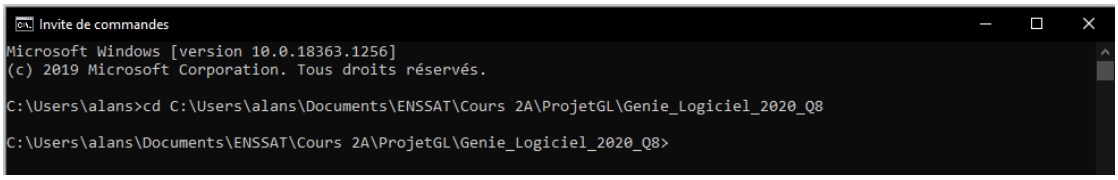


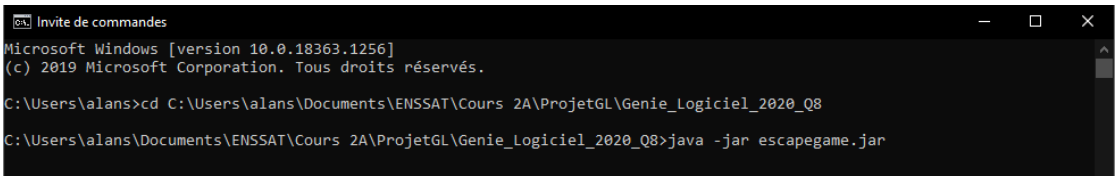
2- Vous devez ensuite vous placer dans le répertoire où se trouve le fichier jar grâce à la commande cd + le chemin d'accès :

5.3.1 Utilisation administrateur

Vous trouverez dans cette section la ligne de commande permettant de lancer E-Scape Game sur un poste destiné à un administrateur. Cela nécessite de taper la commande suivante dans l'invite de commande :

java -jar escapegame.jar



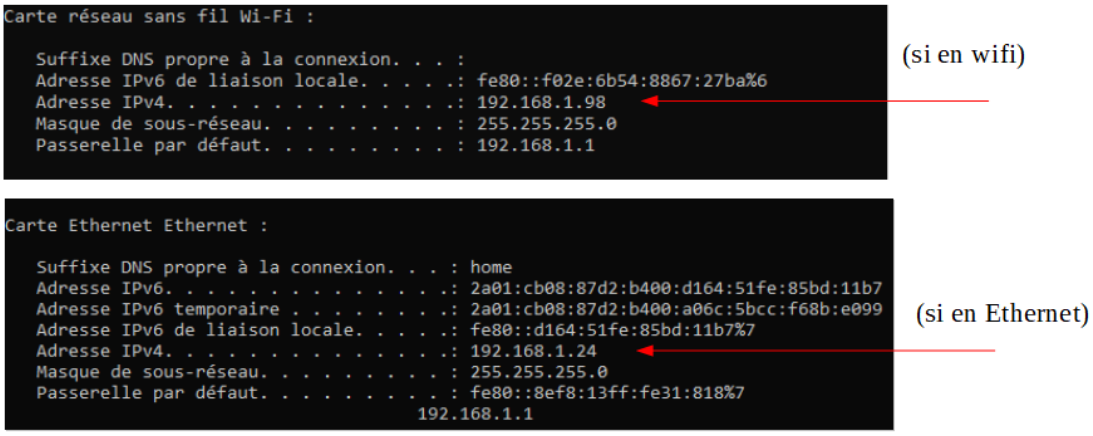


5.3.2 UTILISATION JOUEUR

Vous trouverez dans cette section la ligne de commande permettant de lancer E-Scape Game sur un poste destiné à des joueurs. Cela nécessite de taper la commande suivante dans l'invite de commande :

java -jar escapegame.jar « ip du poste de l'administrateur en charge des clients »

Pour récupérer cette adresse ip, il faut à nouveau utiliser un invite de commande sur le poste de l'administrateur et utiliser la commande ipconfig.



Il est conseillé d'avoir des ip fixes sur les postes des administrateurs. Il suffit alors d'exécuter la commande avec la bonne ip :