Code ::TimeRewind

Par Mathieu RABOT

Glossaire

Cette partie du dossier va contenir différents mots et leur définition et explication s'il y a des termes spécifiques.

Classes mappées :
Souvent dans le projet, quand je parle de base de données, je parle de « Classe mappée » qui est enfaite une fonctionnalité de Hibernate, l'ORM que j'utilise, qui permet que dans le fichier de configuration d'Hibernate on écrive le chemin direct vers le fichier comprenant la classe ce qui permet à Hibernate de savoir de quelle classe on parle et surtout ça la crée dans la base de données.

```
<mapping class="model.chapter.Chapter" />
<mapping class="model.account.Account" />
<mapping class="model.account_Own_Character.Account_Own_Character" />
```

Gacha :
Un gacha est un type de jeu ou le but est d'obtenir des héros, au moyen d'un système aléatoire.

IDE :
C'est environnement de développement pour plusieurs langages de programmation tel que le Java ou autre, qui comporte un éditeur de texte ainsi que des fonctions propres au codage tel que le compilateur ou le débogueur.

Java :
Java est un langage de programmation orienté objet qui se compile en passant par une machine virtuelle Java ignorant le système d'exploitation de l'appareil qui exécute le programme.

ORM (Mapping objet-relationnel) :
C'est un programme informatique qui se place entre l'application et la base de données qui permet de simuler une base de données orientée objet. Il peut donc convertir les différents objets dans l'application en requête pour la base de données.

Package :
Quand on parle de package c'est qu'on parle d'un dossier qui regroupe des fichiers contenant des classes.
Par exemple : model.databasemanager est un package qui regroupe toutes les classes qui ont un rapport avec la gestion de la base de données.

Table des matières

1 Analyse préliminaire....4
1.1 Introduction.... 4
1.2 Organigramme du projet.... 4
1.3 Objectifs 5
1.3.1 Différentes difficultés à venir 5
1.3.2 Planification initiale.... 6

2 Analyse / Conception 7
2.1 Concept.... 7
2.1.1 Gestion des maquettes 7
2.1.2 Gestion des données 13
2.2 Stratégie de test 15
2.3 Risques techniques 15
2.3.1 Apprendre l'utilisation d'un ORM (Hibernate) 15
2.3.2 Gestion du temps....15
2.4 Planification 16
2.5 Dossier de conception 17
2.5.1 Choix du matériel : 17
2.5.2 Environnement de travail :....17
2.5.3 Réalisation des maquettes :....18
2.5.4 Gestion des bases de données :....19
2.5.5 Diagramme projet :....20
2.6 Convention de nommage....22
2.6.1 PascalCase....22
2.6.2 CamelCase 22
2.6.3 Autres conventions....22
2.6.4 Raisons....23

3 Réalisation....24
3.1 Dossier de réalisation 24
3.1.1 Répertoire du logiciel : 24
3.1.2 Liste des fichiers de mon projet :....24
3.1.3 Version de mon produit : 28
3.1.4 Description des librairies utilisées : 29
3.2 Choix effectué durant le projet....30
3.2.1 Choix de la création de la base donnée 30
3.3 Description des tests effectués....31
3.3.1 Différents tests : 31
3.3.2 Tests effectués par des externes 32
3.4 Erreurs restantes....33
3.5 Liste des documents fournis....34

4 Conclusions 34
4.1 Objectifs atteints ?....34
4.2 Points positifs / négatifs....34
4.3 Difficultés particulières....34
4.4 Suite pour le projet 34

5 Annexes....35
5.1 Aide reçu lors du projet....35
5.2 Sources – Bibliographie – Acquisition des connaissances....35

5.3 Journal de travail 37
5.4 Manuel d'Installation 43
5.5 Archives du projet 43

1 Analyse préliminaire

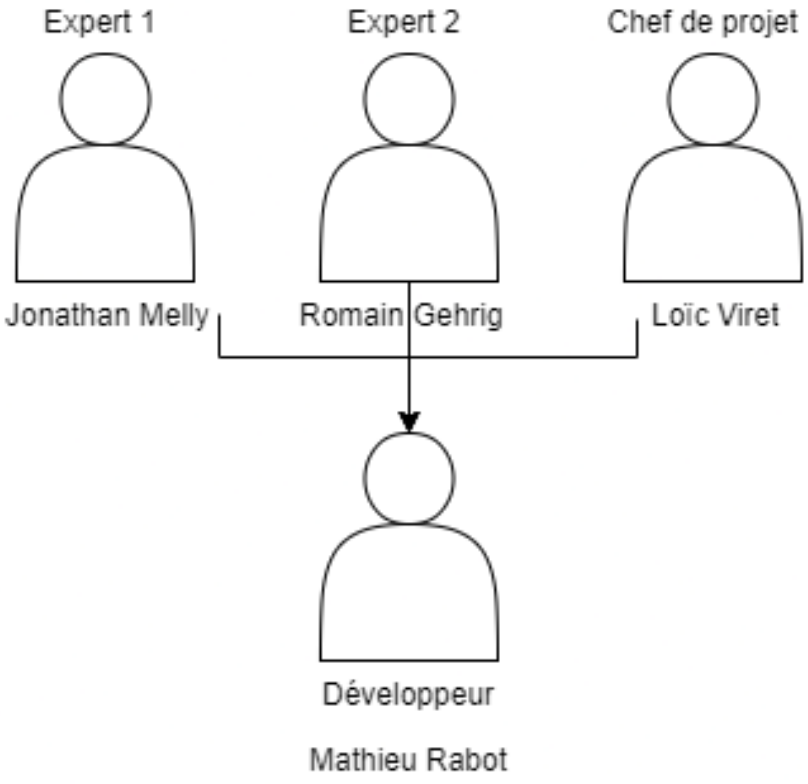
1.1 Introduction

Le projet est une idée que j'ai depuis quelques années maintenant, c'est de faire un J'ai récolté plusieurs d'idées au fur et à mesure du temps que j'ai mis dans un fichier et j'ai décidé de réaliser ce jeu durant mon **Travail Pratique Individuel**.

Ça me permettait d'enfin réaliser ce projet et me donne **de la motivation** à travailler dessus, j'ai aussi décidé d'apprendre **un nouveau langage** pour moi en même temps, ce qui était un peu une sorte d'obligation. Je m'oblige à réaliser mon jeu en risquant de rater mon TPI.

Mais ça ne m'empêche pas de prendre énormément de plaisir à travailler dessus, je trouve vraiment ça très intéressant, ce que j'apprends et ce que je peux réaliser.

1.2 Organigramme du projet



1.3 Objectifs

Le but du projet est de réaliser un jeu vidéo d'aventure en **Java**.

Les objectifs suivants sont tirés de mon **cahier des charges**, se seront des actions que mon code devra implémenter.

- La création de compte de l'utilisateur (pseudo + mot de passe)
- La création d'un système aléatoire d'obtention de personnage (avec pourcentage pour chaque niveau de rareté de personnage : rare, épique et légendaire)
- Mode histoire pour le jeu avec l'affrontement d'ennemis
    - Le mode histoire devra permettre d'accéder au combat suivant sur la piste des combats.
    - Les combats se dérouleront dans un écran spécifique
- Obtention d'équipements (augmentation de niveau qui font évoluer les attributs défensifs ou offensifs avec un ratio ou un tableau de rations prédéfinis).

Les éléments de jeux (personnages, équipements) devront être suffisamment présents sans être chronophages, un minimum de 5 par types sera suffisant.

Une attention toute particulière devra être faite concernant le développement en vue de son évolutivité future.

1.3.1 Différentes difficultés à venir

Pour les différentes difficultés à venir, je pense que le plus dur sera l'apprentissage de l'utilisation d'un ORM, comme je sais pas du tout ce que c'est et je sais pas du tout comment l'implémenter, ça va me prendre pas mal de temps à l'installer sur mon projet.

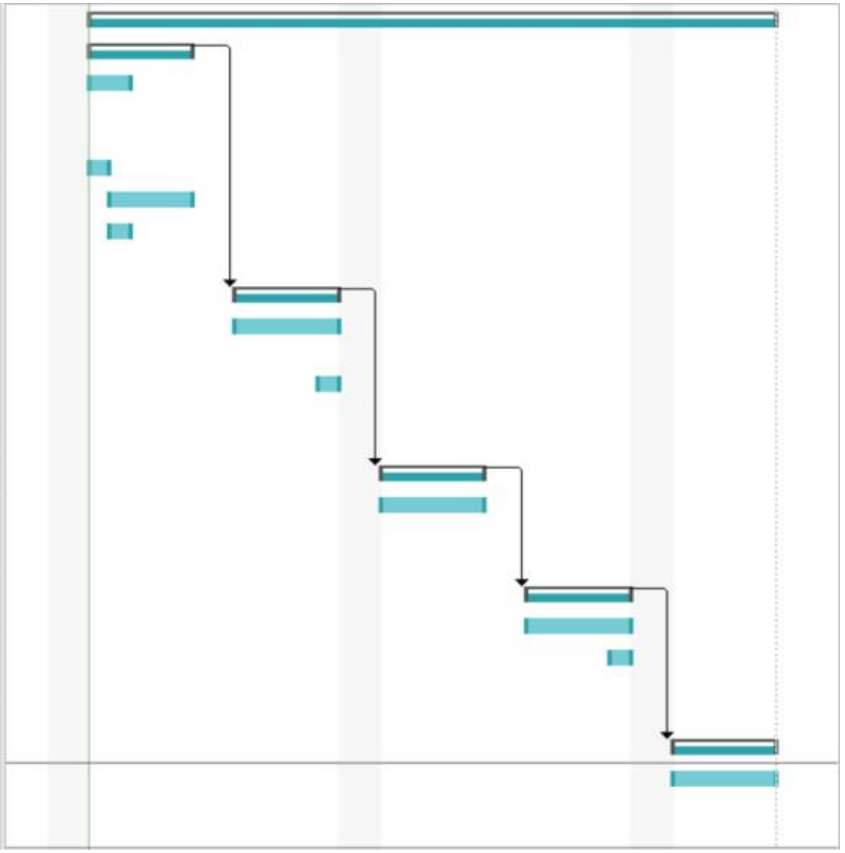
La gestion du temps aussi ça va être un problème, il y a énormément à faire en un mois et ça me stresse pas mal sachant que j'ai pris beaucoup de temps à faire mon rapport de projet pour le pré-TPI

1.3.2 Planification initiale

J’ai divisé ma planification en semaine, en ajoutant les différents jours ou semaine de vacances durant la durée du projet.

Et j’ai tout rediviser en catégorie des aspects que je vais devoir traiter dans mon projet avec les toutes les tâches ajouter par section.

Projet Code::TimeRewind	**25 jrs**	**Lun 03.05.21**	**Ven 04.06.21**	
Semaine 1	**5 jrs**	**Lun 03.05.21**	**Ven 07.05.21**	
Commencer la création des maquettes nécessaire représentant les différentes vues manquante du jeu	2 jrs	Lun 03.05.21	Mar 04.05.21	
Création de cette planification initiale	1 jr	Lun 03.05.21	Lun 03.05.21	
Implémentation de l'ORM Hibernate dans le projet	4 jrs	Mar 04.05.21	Ven 07.05.21	
Apprentissage de l'utilisation d'un ORM	1 jr	Mar 04.05.21	Mar 04.05.21	
Semaine 2	**5 jrs**	**Lun 10.05.21**	**Ven 14.05.21**	**2**
Création du système d'obtention des personnages par rareté	5 jrs	Lun 10.05.21	Ven 14.05.21	
Faire tester mon jeu par d'autre personne pour voir si les fonctionnalités fonctionnent	1 jr	Ven 14.05.21	Ven 14.05.21	
Semaine 3	**5 jrs**	**Lun 17.05.21**	**Ven 21.05.21**	**8**
Création du système d'équipement pour chaque personnage	5 jrs	Lun 17.05.21	Ven 21.05.21	
Semaine 4	**5 jrs**	**Lun 24.05.21**	**Ven 28.05.21**	**12**
Ajout du système d'amélioration des équipements	5 jrs	Lun 24.05.21	Ven 28.05.21	
Faire tester mon jeu par d'autre personne pour voir si les fonctionnalités fonctionnent	1 jr	Ven 28.05.21	Ven 28.05.21	
Semaine 5	**5 jrs**	**Lun 31.05.21**	**Ven 04.06.21**	**15**
Faire les derniers tests de mon jeu pour vérifier si toutes les fonctionnalités demandées sont bien fonctionnel	5 jrs	Lun 31.05.21	Ven 04.06.21	



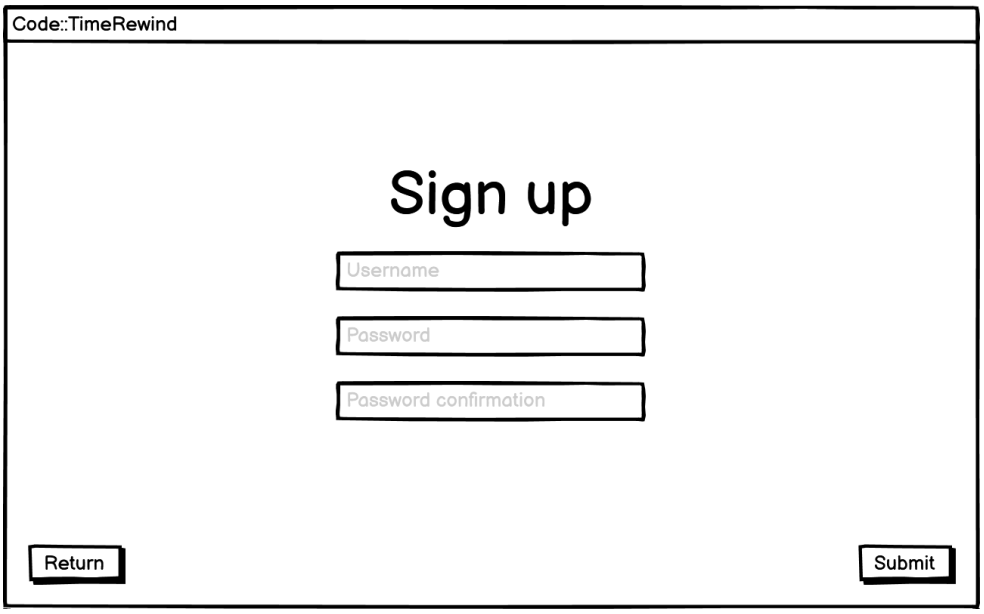
2 Analyse / Conception

2.1 Concept

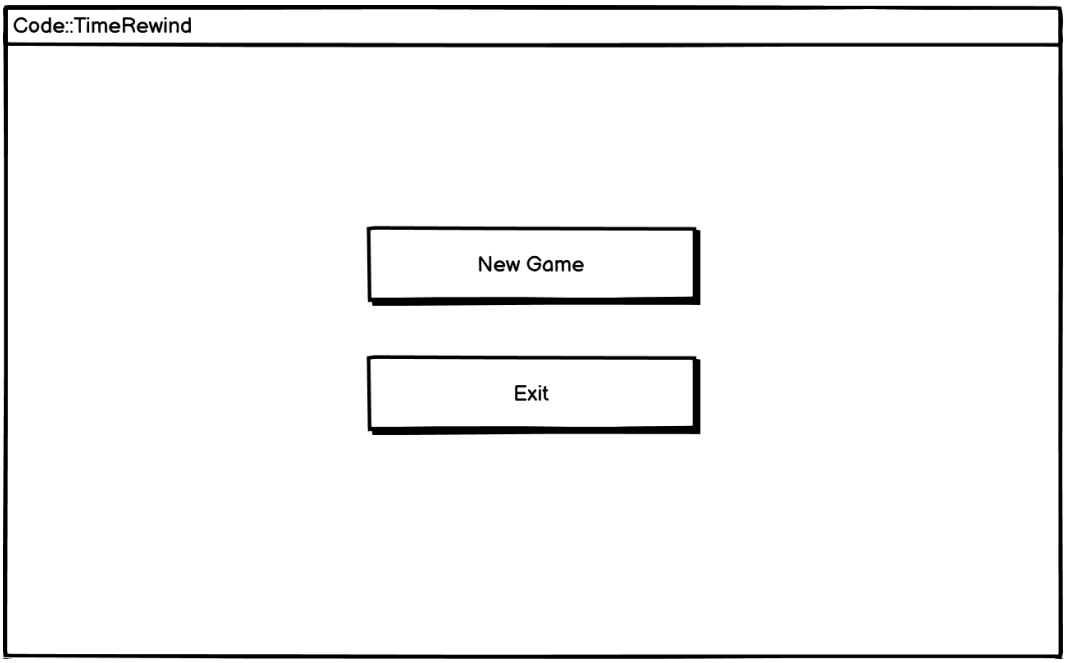
2.1.1 Gestion des maquettes

Au niveau de la conception, j'ai beaucoup réfléchi à la manière dont mon jeu va ressembler, c'est pour ça que j'ai dessiné plein de vue, il y en a certaine qui ne seront peut-être pas utilisé par manque de temps ou par changement d'avis durant le projet :

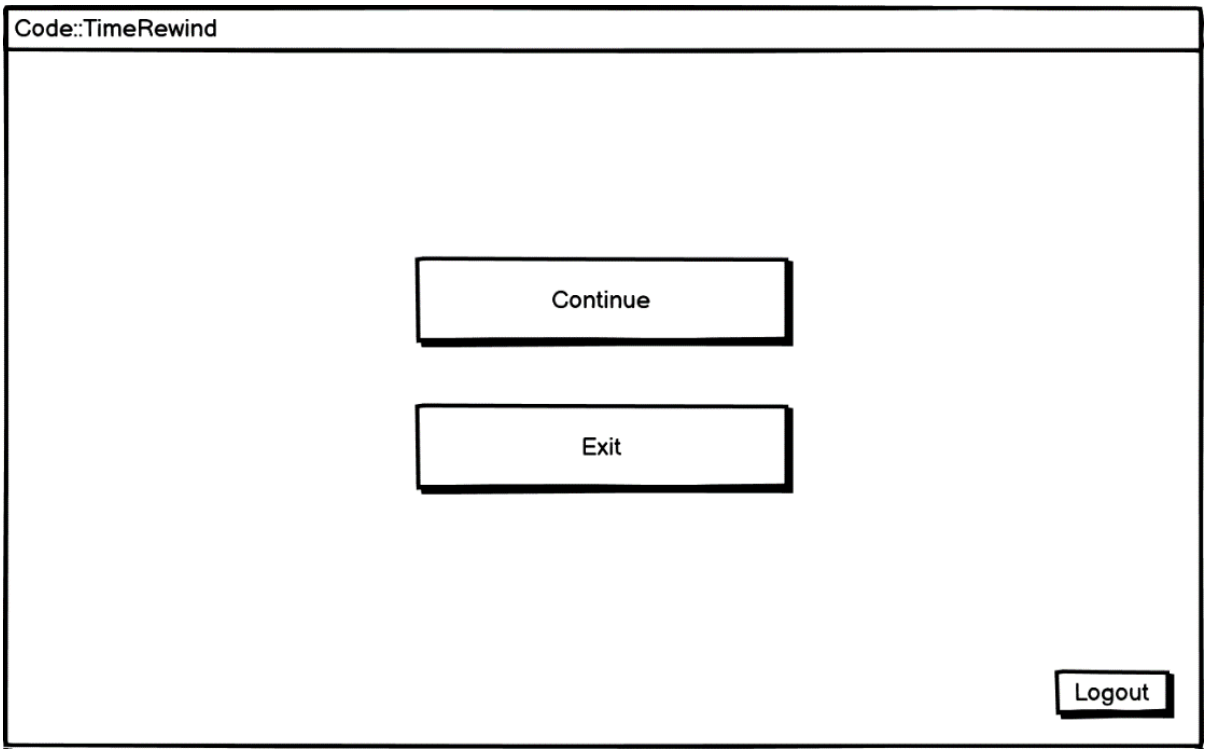
Création d'un compte dans le jeu :



Vue si aucun compte n'est créé dans la base de données :

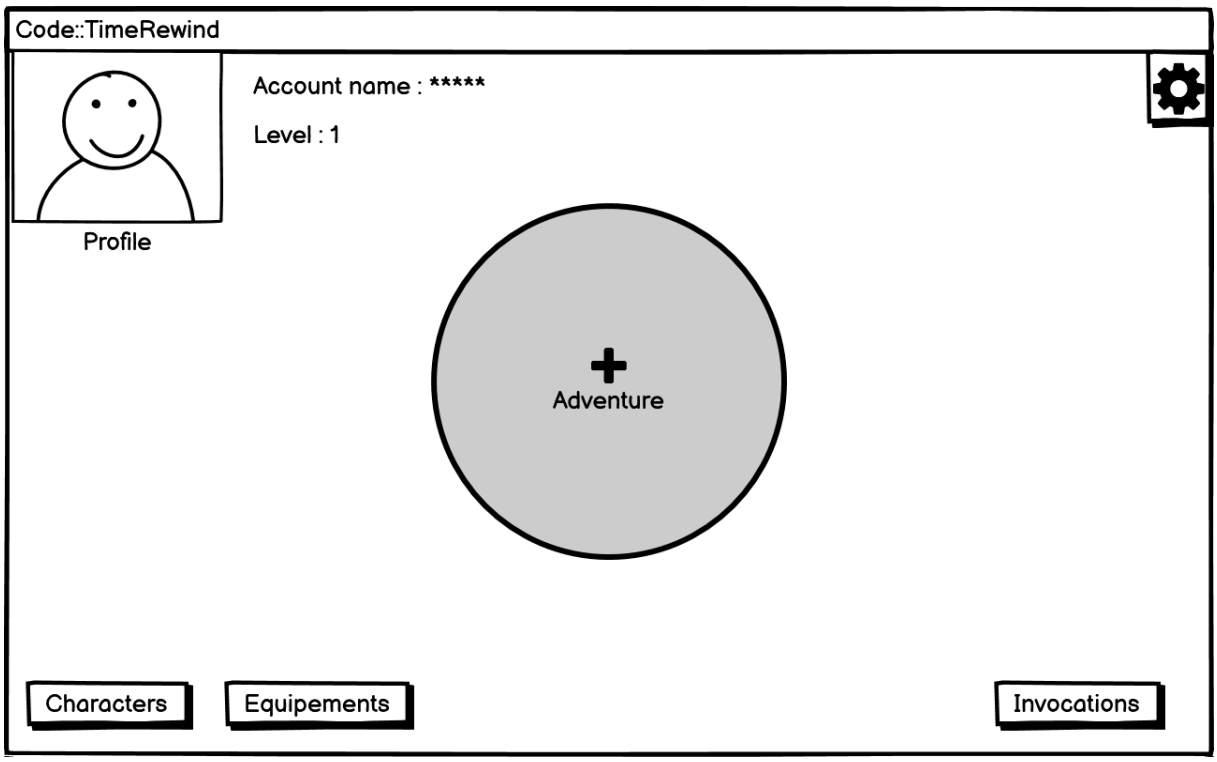


Vue si un compte existe déjà dans la base de données et propose à ne pas se connecter mais à directement jouer en cliquer sur « Continue ».

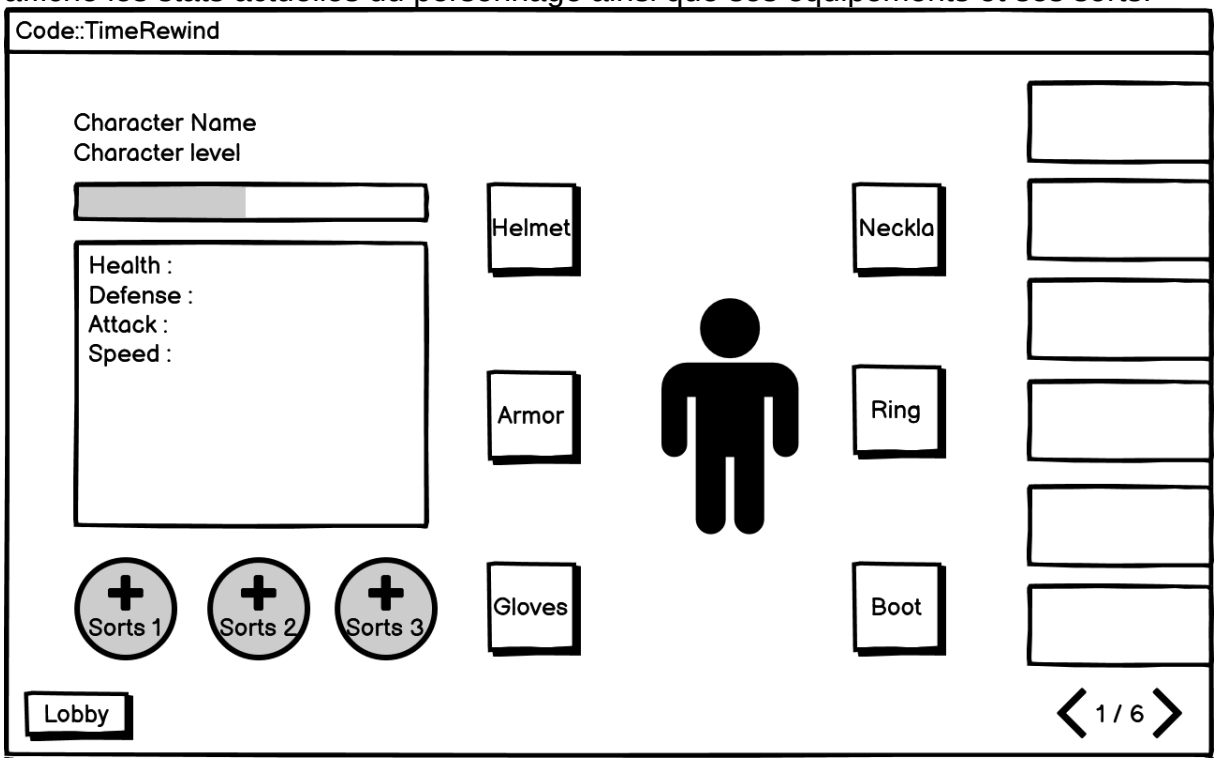


C'est la vue qu'on voit quand on a passé l'étape de la connexion.

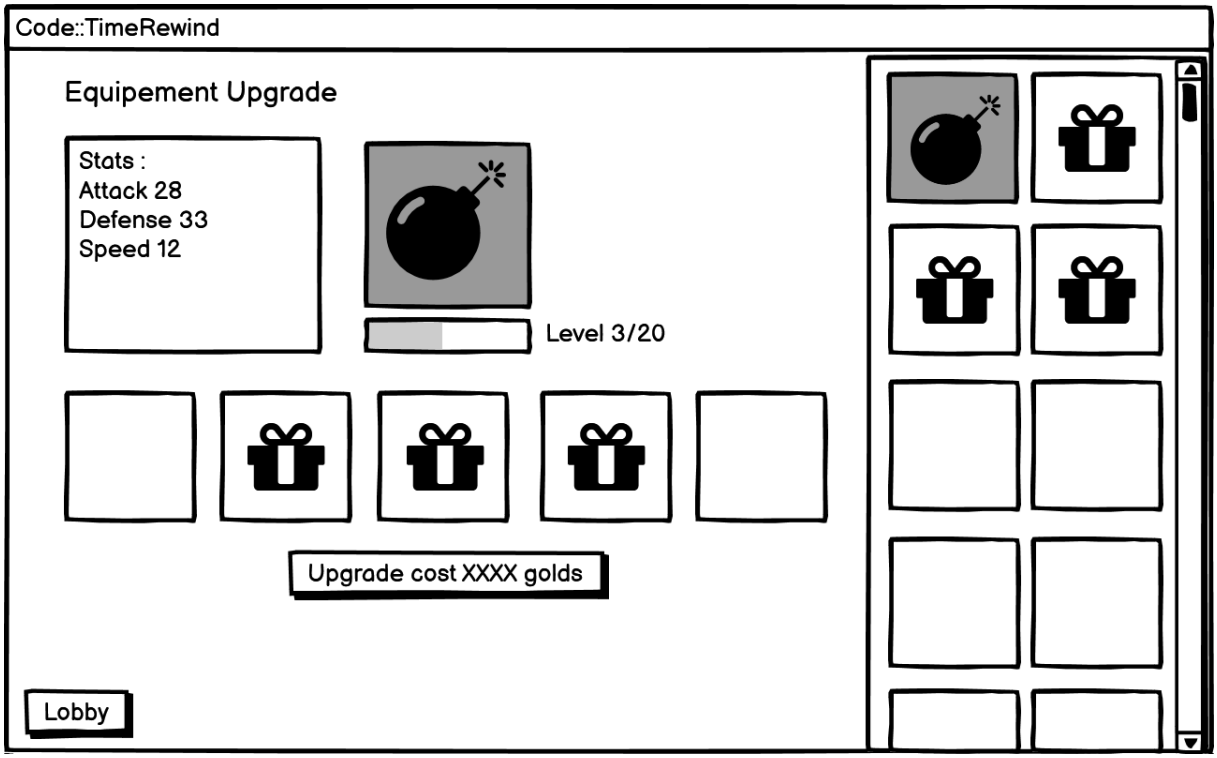
On arrive directement sur le lobby où on peut choisir si on veut jouer ou si on veut se balader dans les différentes autres vues du jeu.



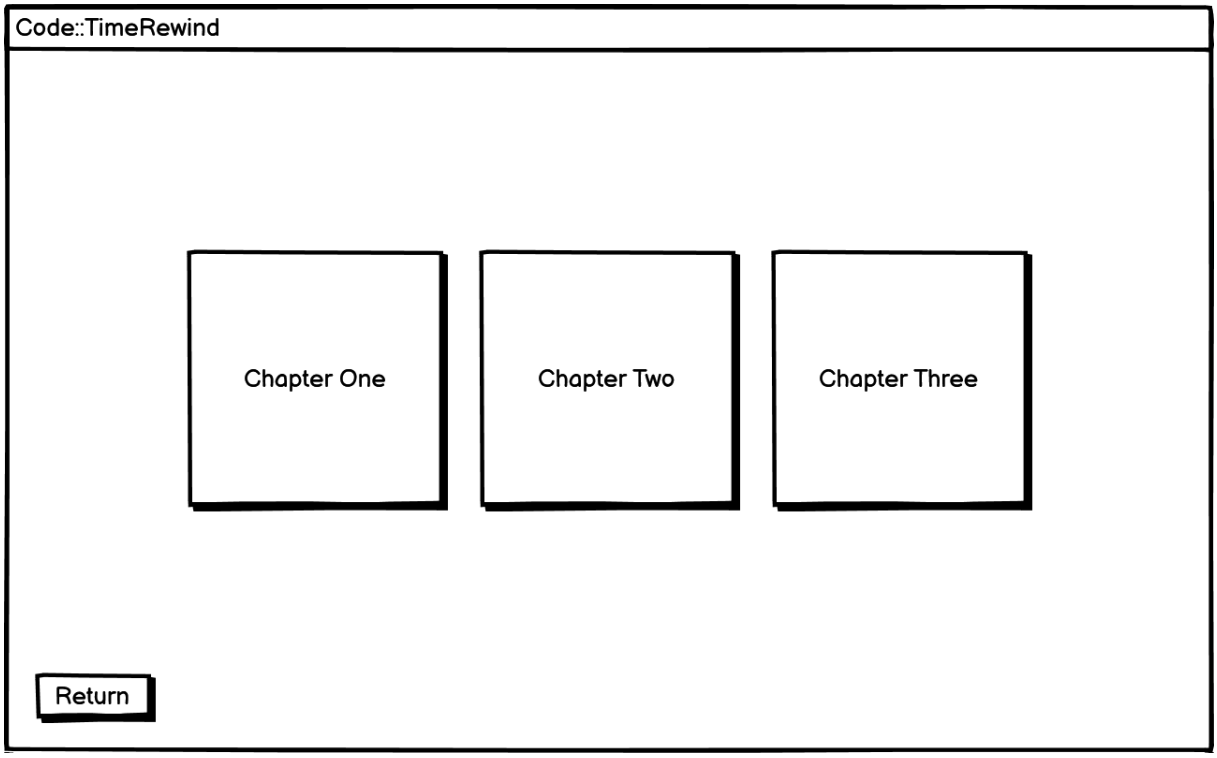
C’est la vue si on clique sur le bouton « Characters » dans la vue du lobby, elle affiche les stats actuelles du personnage ainsi que ses équipements et ses sorts.



C’est la vue si on clique sur le bouton « Equipements » dans la vue du lobby, elle affiche tous les équipements que le compte a, ainsi que les stats de l’équipement sélectionné.

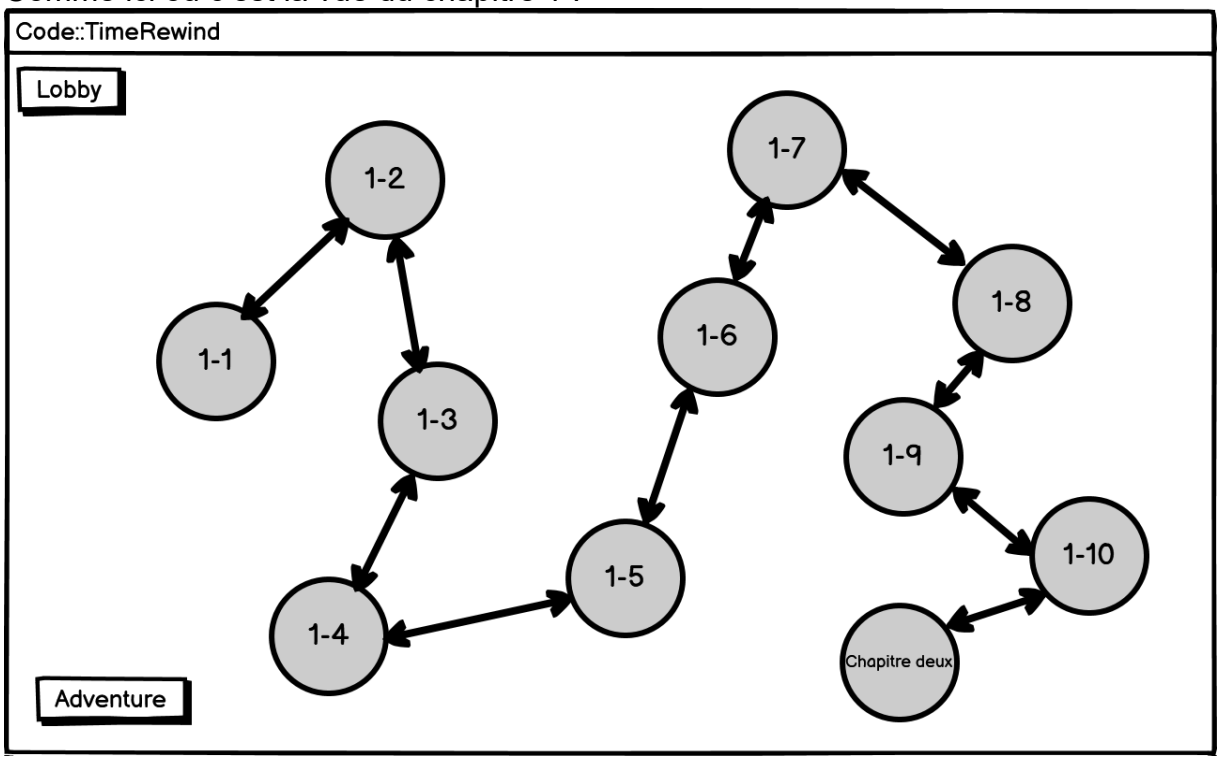


C’est la vue si on clique sur le bouton « Aventure » dans la vue du lobby, elle permet d’accéder aux différentes vues pour jouer au jeu. Elle accède à cette vue qui contient

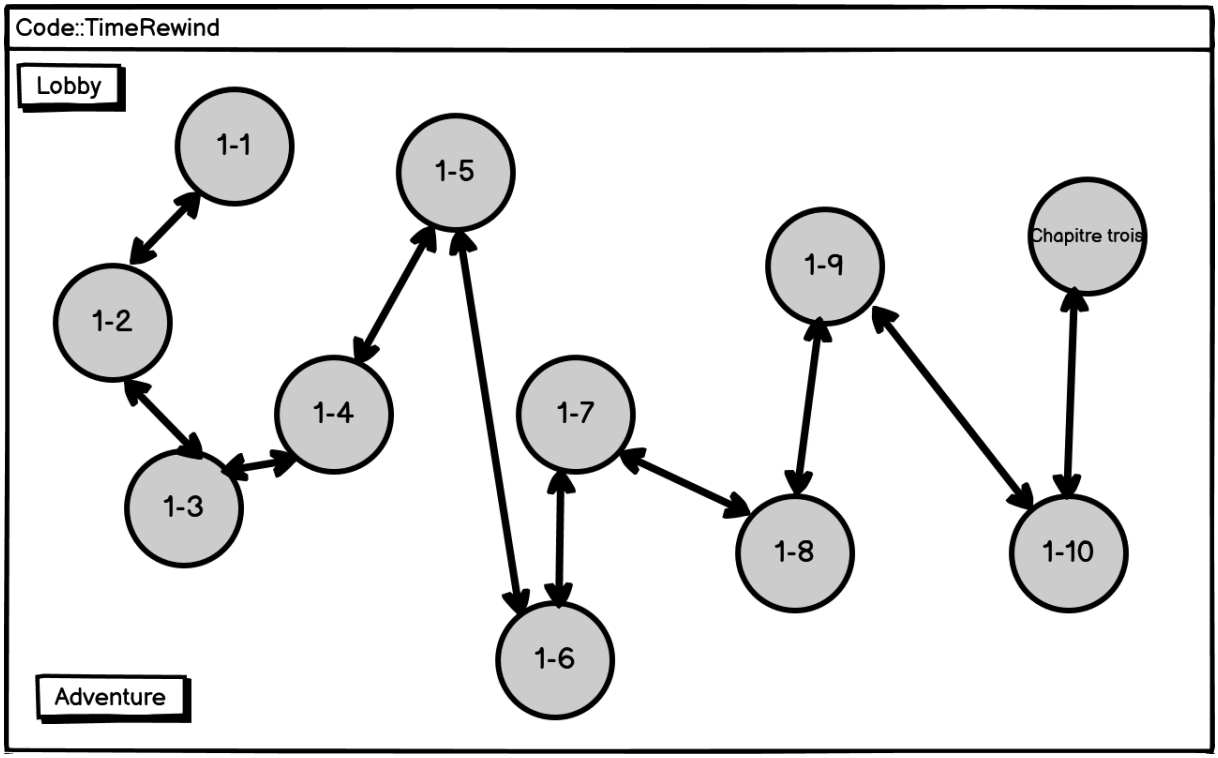


les différents chapitres du jeu. Pour l'instant il y en a que trois mais à l'avenir j'en rajouterais.

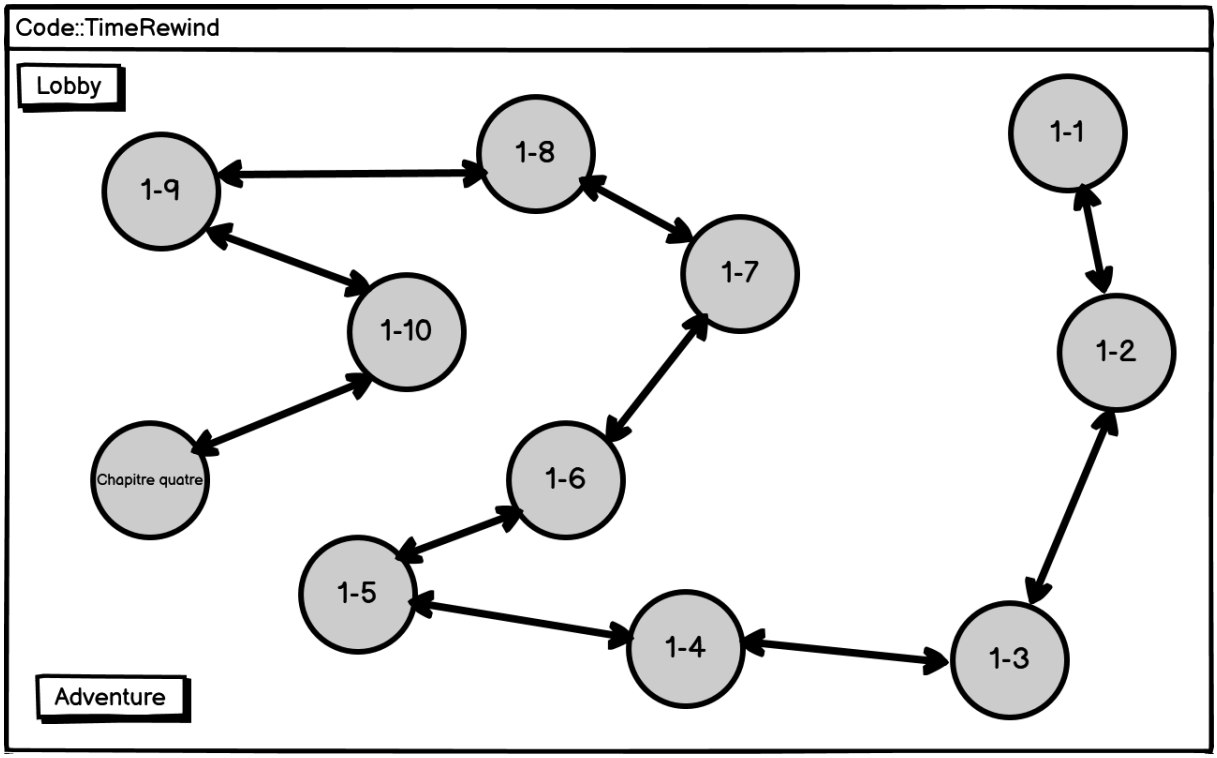
Après la vue des chapitres, nous avons la vue des niveaux par chapitres.
Chaque chapitre contiendra une dizaine de niveaux différents afficher comme ceci.
Comme ici ou c'est la vue du chapitre 1 :



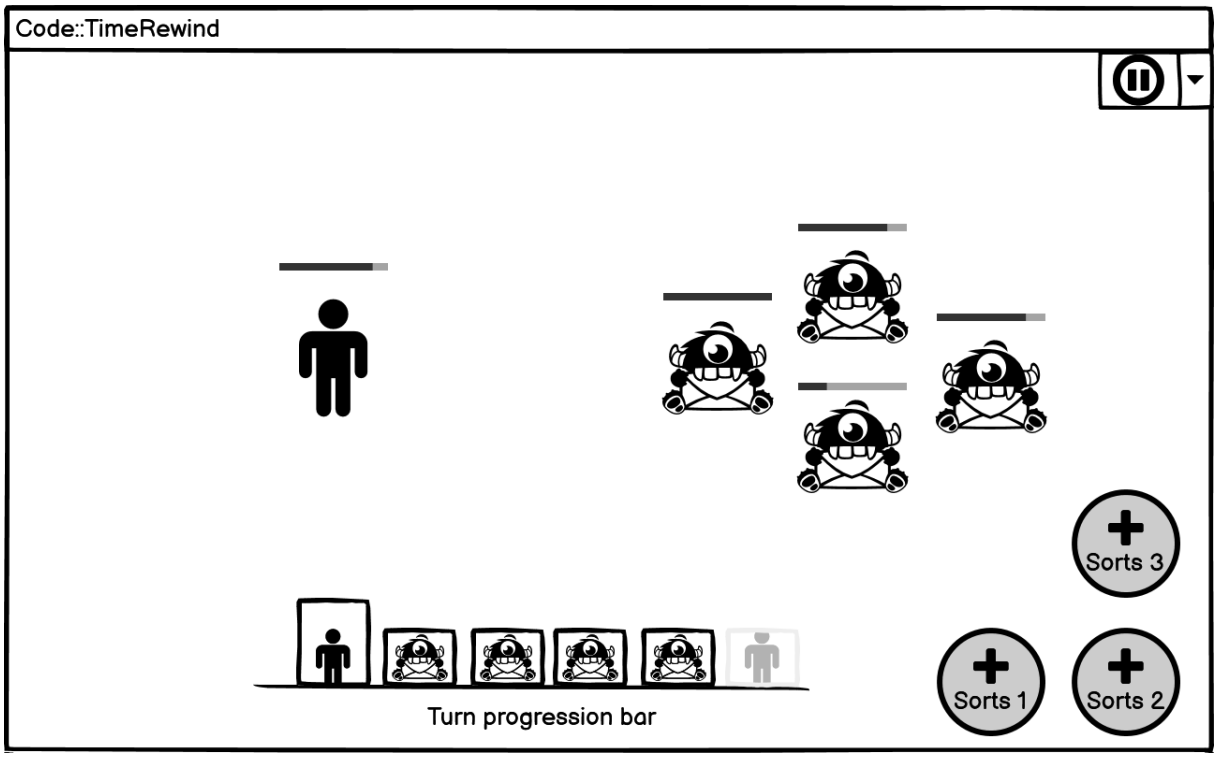
La vue du chapitre 2 :



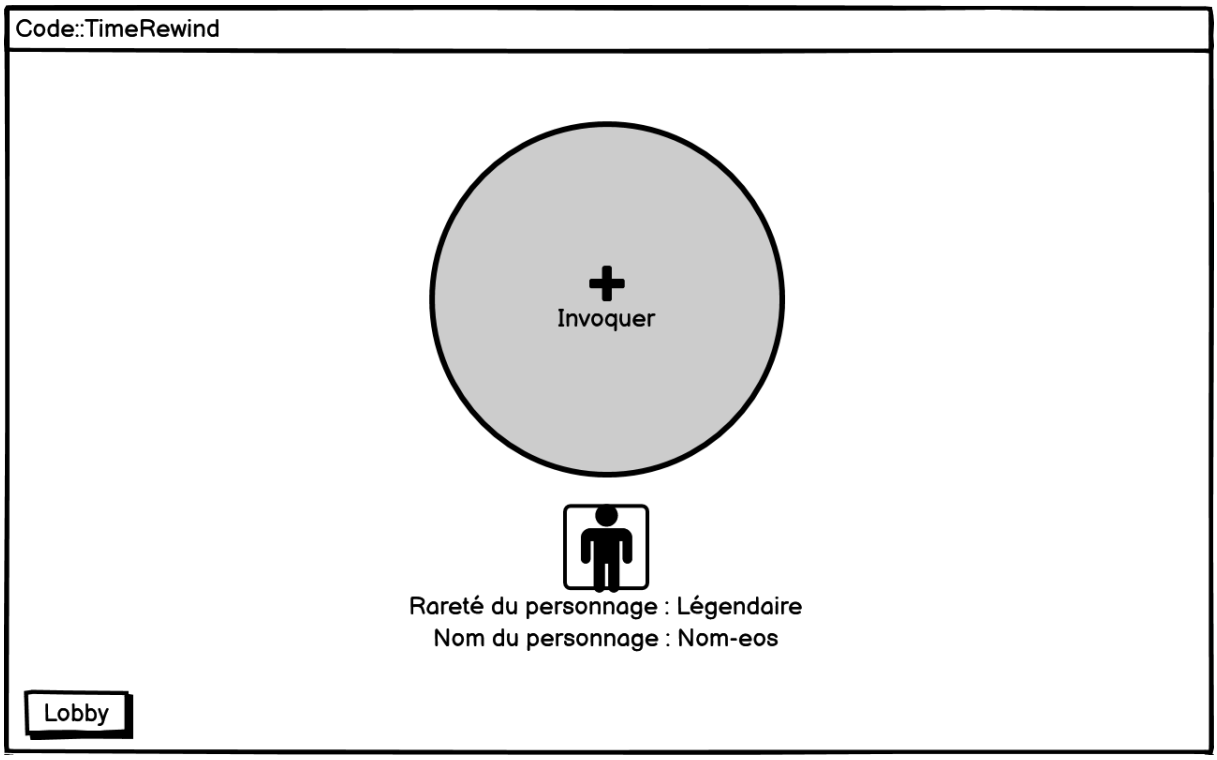
La vue du chapitre 3 :



Cette vue c'est le moment où on clique sur un niveau et qu'on commence un combat, la vue affiche les différentes entités avec leur barre de vie et leurs sorts



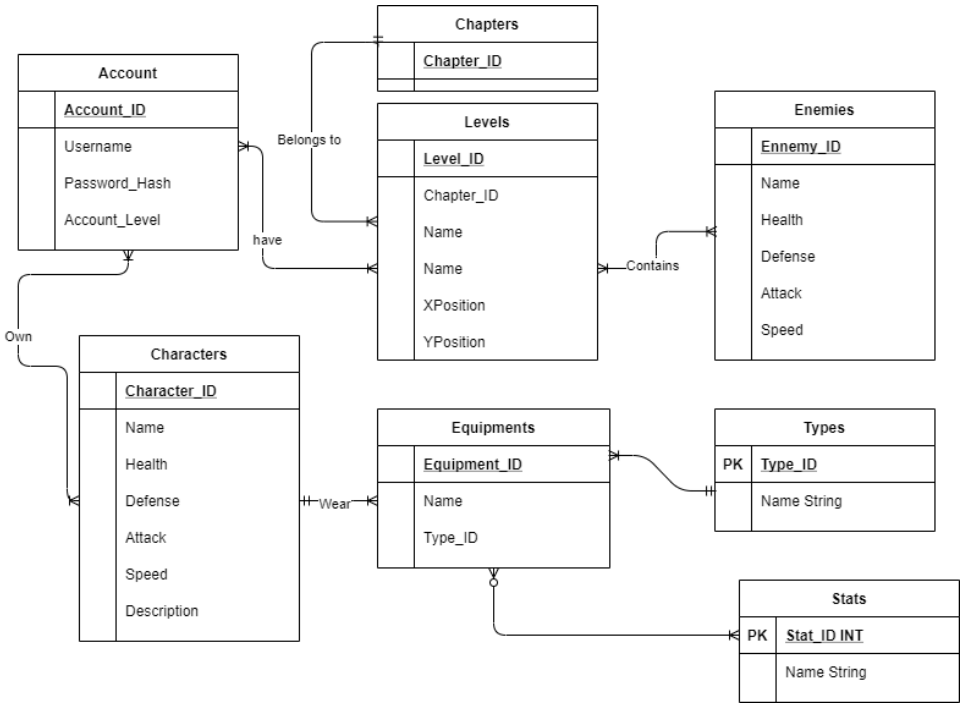
Cette vue permet d'invoquer des nouveaux personnages, avec un certain niveau de rareté, qui se joindront à notre équipe et qu'on pourra faire évoluer en jouant avec.



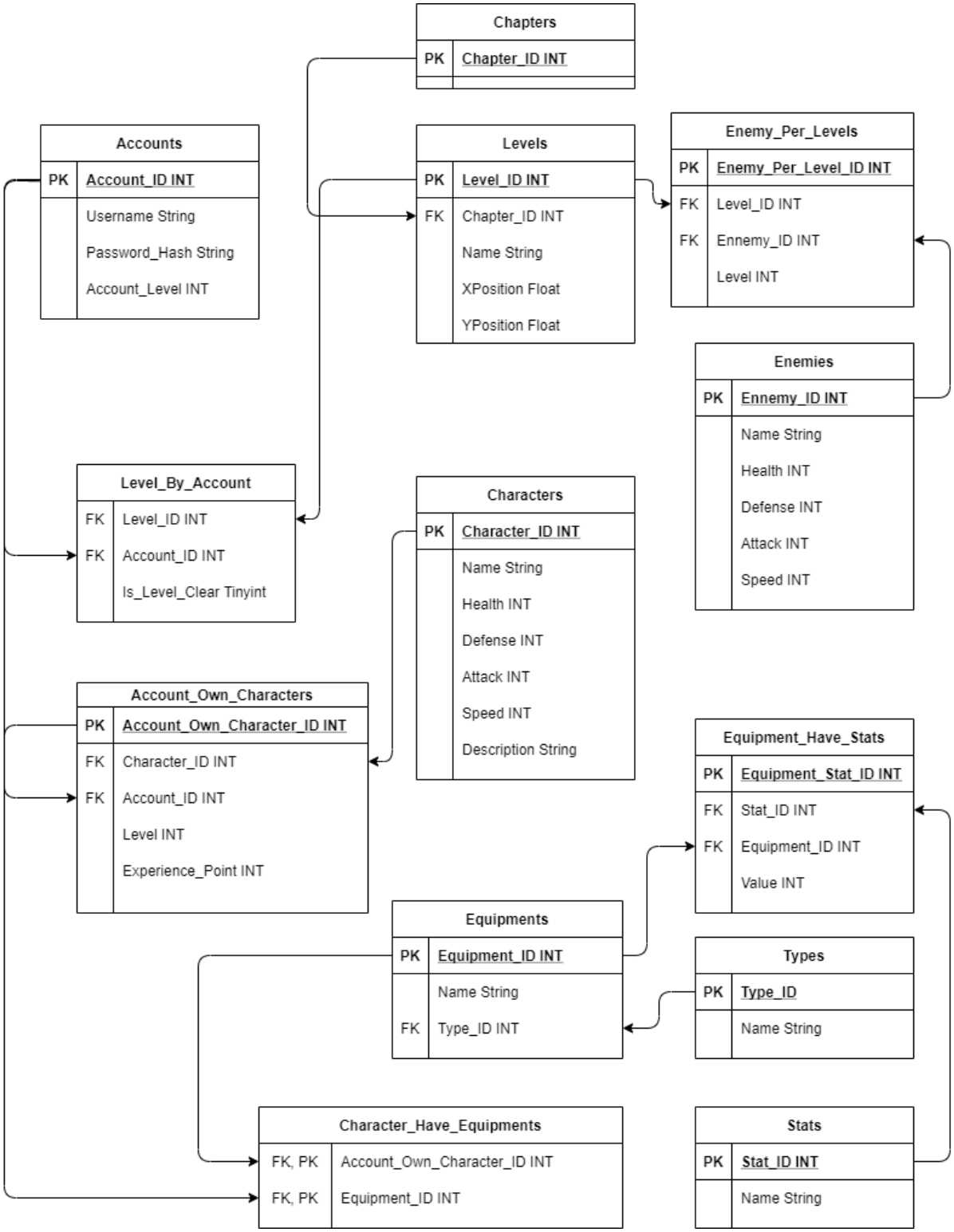
2.1.2 Gestion des données

Pour la gestion des données, j'ai créé un MCD et un MLD qui décrivent les relations entre les tables et la manière dont je l'imagine.

2.1.2.1 MCD



2.1.2.2 MLD


Chapters
PK Chapter_ID INT
Accounts
PK Account_ID INT
Username String
Password_Hash String
Account_Level INT
Levels
PK Level_ID INT
FK Chapter_ID INT
Name String
XPosition Float
YPosition Float
Enemy_Per_Levels
PK Enemy_Per_Level_ID INT
FK Level_ID INT
FK Ennemy_ID INT
Level INT
Enemies
PK Ennemy_ID INT
Name String
Health INT
Defense INT
Attack INT
Speed INT
Level_By_Account
FK Level_ID INT
FK Account_ID INT
Is_Level_Clear Tinyint
Characters
PK Character_ID INT
Name String
Health INT
Defense INT
Attack INT
Speed INT
Description String
Account_Own_Characters
PK Account_Own_Character_ID INT
FK Character_ID INT
FK Account_ID INT
Level INT
Experience_Point INT
Equipment_Have_Stats
PK Equipment_Stat_ID INT
FK Stat_ID INT
FK Equipment_ID INT
Value INT
Equipments
PK Equipment_ID INT
Name String
FK Type_ID INT
Types
PK Type_ID
Name String
Character_Have_Equipments
FK, PK Account_Own_Character_ID INT
FK, PK Equipment_ID INT
Stats
PK Stat_ID INT
Name String

2.2 Stratégie de test

Ma stratégie se repose sur les tests de mon jeu, je ferais en sorte d'avoir des versions stables de mon application pour les faire testés à différentes personnes qui pourront tester s'ils trouvent des bugs et me faire des retours.

J'aurais aussi plusieurs tests unitaires qui me permettrons de tester des fonctionnalités de mon logiciel et voir si l'ajout de nouvelle méthode ne casse pas tout le code.

2.3 Risques techniques

2.3.1 Apprendre l'utilisation d'un ORM (Hibernate)

Apprendre de nouvelle chose c'est toujours long et fastidieux, je pense que ça va me prendre beaucoup de temps à comprendre et implémenter cette nouvelle technologie

2.3.2 Gestion du temps

Comparé au pré-TPI, le TPI contient beaucoup moins de temps ce qui m'inquiète si je vais réussir à bien tout finir et atteindre les objectifs.

2.4 Planification

2.5 Dossier de conception

Dans ce chapitre, je vais détailler les différents logiciels que j'utilise, pourquoi je les utilise et qu'est-ce qu'ils apportent à mon projet, c'est pourquoi j'ai découpé ce chapitre en plusieurs sous-chapitres qui me permettent de rentrer plus dans les détails.

2.5.1 Choix du matériel :

Au niveau du matériel choisis, je dois obligatoirement faire mon travail en classe, donc utilisé la machine mise à disposition par le CPNV.

La machine contient cette configuration :

Processeur :	Intel i7-6400 3.40GHz 8 cœurs
Carte Graphique :	Intel HD Graphics 530
Mémoire :	16 Go
Stockage :	160 Go
Système d'exploitation :	Windows 10

2.5.2 Environnement de travail :

Mon environnement de travail en général se fera sur Eclipse qui est un IDE spécialement fait pour Java.

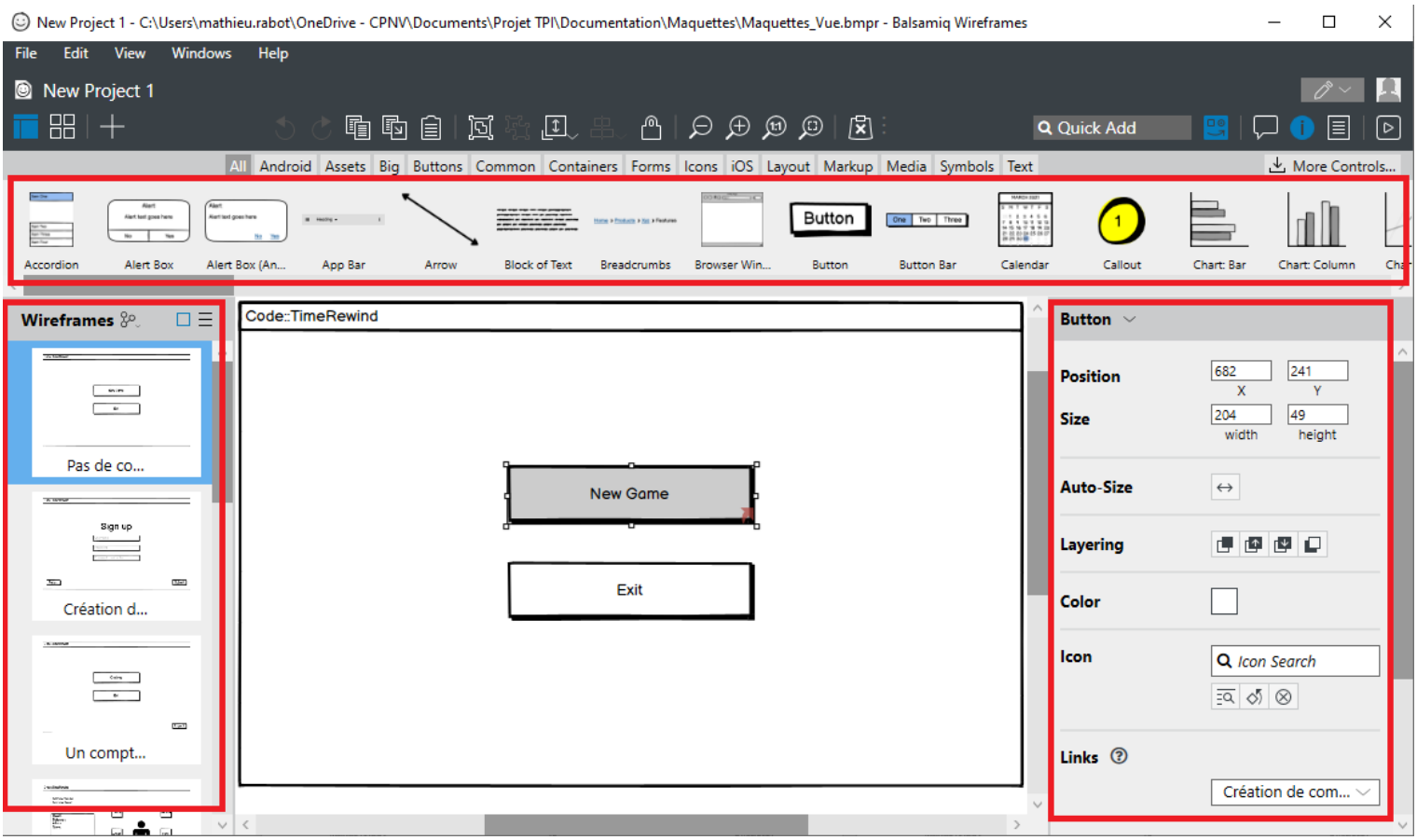
J'ai choisi cette IDE parce que Eclipse est gratuit et très complet dû au Marketplace intégré qui offre beaucoup de possibilité d'optimisation de la plateforme.

On m'a souvent conseillé de prendre IntelliJ comme plateforme de base pour coder en Java, mais la raison principale pour laquelle je veux utiliser Eclipse est que je vais continuer de programmer mon jeu après mon TPI, ce qui veut dire que je n'aurais plus accès a IntelliJ car je ne serais plus étudiant.

J'ai pris la dernière version stable de Eclipse (actuelle) qui est la 2020-12 (4.18)

2.5.3 Réalisation des maquettes :

Toute la gestion de mes maquettes, que j'ai présentées dans le point 2.1, a été faite sur le logiciel Balsamiq Wireframes comme présenté ci-dessous :



- Le premier rectangle rouge contient les différents modules qui composent les vues, se sont plein d'outils qu'on peut déplacer dans notre vue et qu'on peut ensuite paramétrer.
- Le rectangle de gauche contient toutes les vues qui composent notre projet, c'est l'endroit où on peut naviguer entre les différentes vues rapidement.
- Le rectangle de droite contient les différents paramètres qu'on peut modifier pour chaque module qu'on a déplacé dans notre vue, dans l'exemple ici, je peux configurer la position, la taille, la position du calque, la couleur, l'icône et le lien avec quel vue le bouton redirige.

2.5.4 Gestion des bases de données :

Pour la conception de ma base de données, j'ai décidé d'utiliser Draw.io qui est un logiciel très pratique pour faire de la modélisation en générale

Et pour l'implémentation de ma base de données, j'ai opté pour une base de données embarquée.

C'est un type de base de données très intéressant parce qu'elle permet de ne pas avoir besoin d'installer de service sur la machine du client ni d'avoir de système de base de données de base.

J'utilise Apache Derby, qui est la base de données embarquée d'Apache, toutes les requêtes créées sont écrites comme du SQL, ce qui permet de mettre à profit mes connaissances en requête et de pas avoir à en créer de nouvelle.

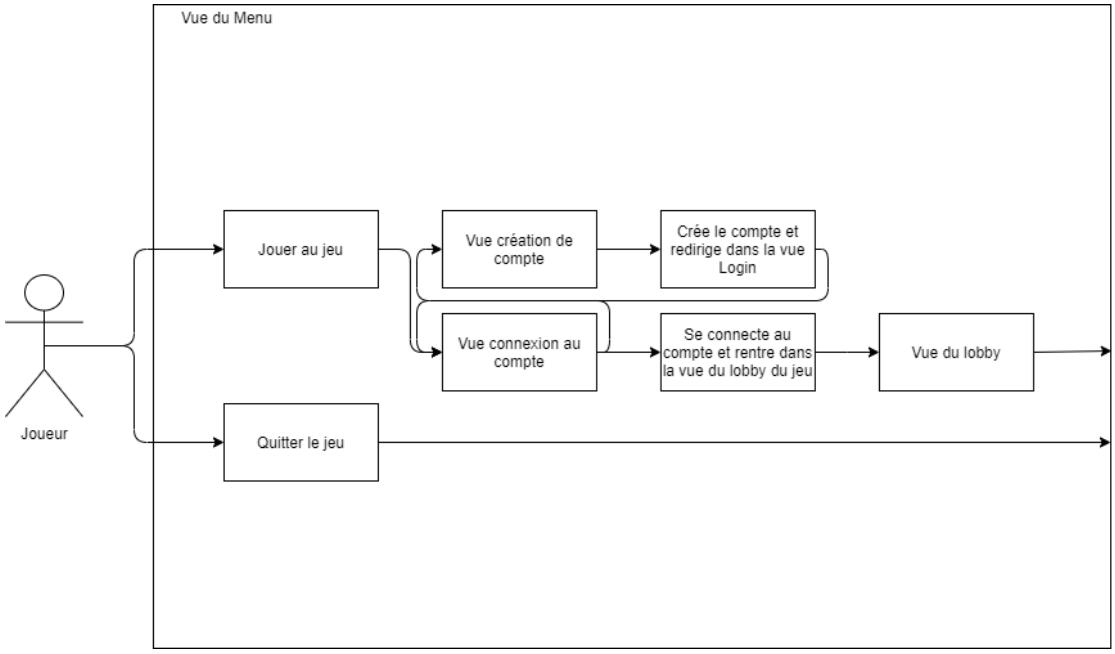
Et comme gestionnaire de base de données, donc pour interagir directement avec Derby, j'utilise DataGrip qui support les bases de données embarquées. Mais je peux aussi y accéder avec l'invite de commande de base de Windows via un utilitaire intégré à Derby qui s'appelle « ij ».

2.5.5 Diagramme projet :

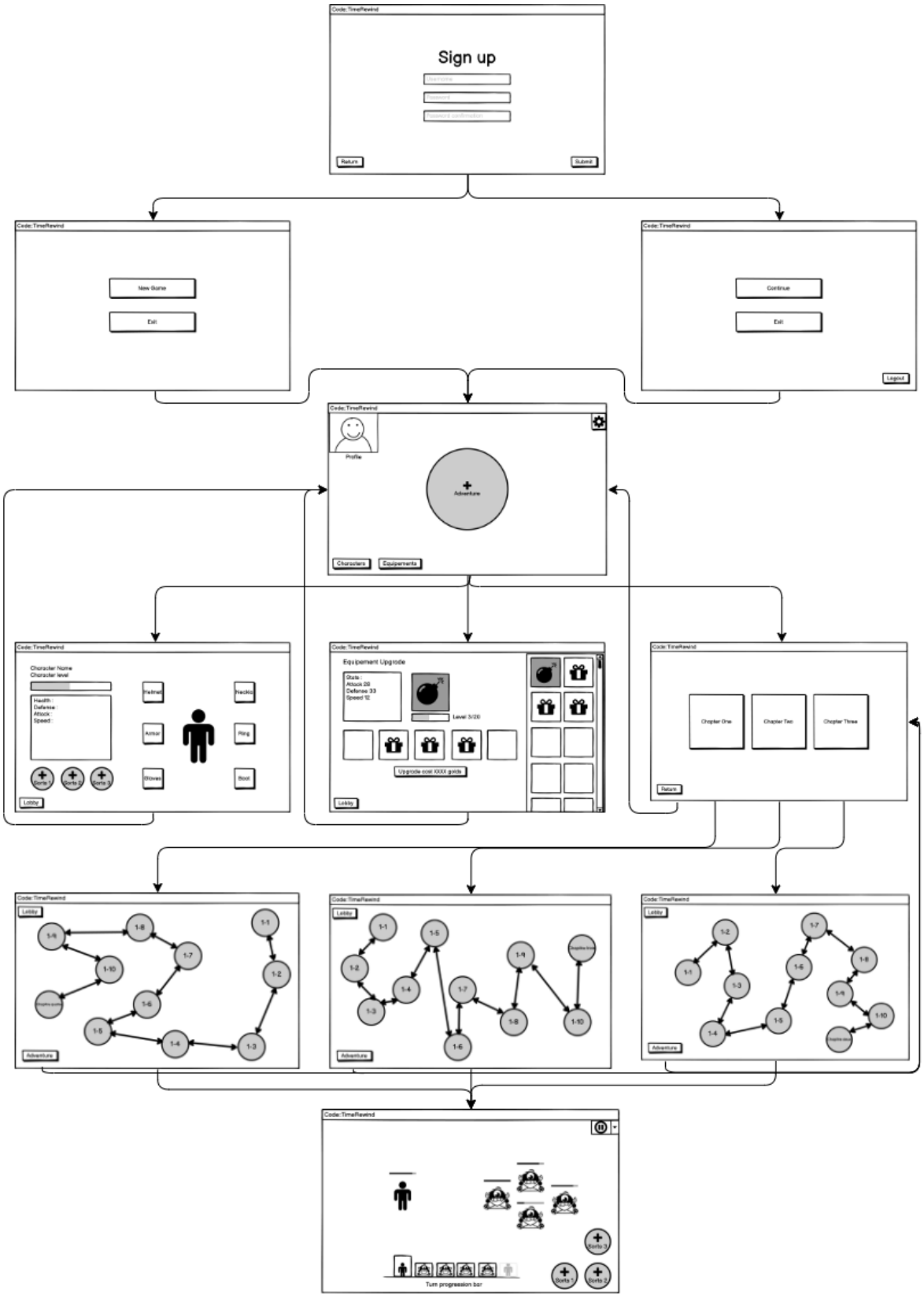
Ce diagramme représente les différents choix que feront le joueur quand il entrera dans le jeu, de base il se trouvera dans la vue du menu et il pourra décider de vouloir quitter le jeu ou de jouer, s'il décide de jouer, ça le redirige dans la vue de connexion ou il pourra choisir de se connecter ou de s'inscrire.

Après s'être inscrit, la vue change pour celle de la vue ou le joueur devra se connecter ce qui le fera rentrer dans la vue du lobby et pourra commencer à jouer.

Et s'il avait déjà un compte, il pourra directement se connecter pour accéder au jeu et continuer avec sa progression actuelle.



Ce diagramme représente l'architecteur / la disposition des maquettes et leur ordre dans le projet, on peut voir comment accéder à une certaine maquette et en sortir.



2.6 Convention de nommage

Au niveau des conventions de nommage, je vais clairement expliquer dans quel cas j'utilise du **CamelCase** ou du **PascalCase**

2.6.1 PascalCase

Le Pascal Case est une convention qui utilisée de cette manière Ex : **N**om**D**e**M**a**V**ariable.

Elle est définie par une **majuscule** en début de nom et a une suite de majuscule à chaque mot.

J'utilise pas souvent cette manière de nommé, les fois ou je l'utilise c'est quand :

- Je parle d'un fichier en particulier Ex : **D**atabase**M**anager.java
- Quand je nomme une classe Ex : Public class **D**atabase**M**anager{ }
- Quand je crée le constructeur de ses classes Ex : Public **D**atabase**M**anager(){}

2.6.2 CamelCase

Le Camel est une convention qui utilisée de cette manière Ex : nom**D**e**M**a**V**ariable.

Elle est définie par une **minuscule** en début de nom et a une suite de majuscule à chaque mot.

Cette convention je l'utilise pour tout le reste si j'utilise pas du PascalCase comme :

- Quand je crée des méthodes pour chaque classes Ex : public void insert**U**ser**I**n**D**atabase().
- Quand je crée des variables dans les classes ou dans les méthodes Ex : private int level**U**ser**A**ccount.

2.6.3 Autres conventions

Pour la création des **packages**, je l'écris uniquement en **minuscule même s'il y a plusieurs mots** Ex : model.databasemanager.

Pour la constitution des **constantes** dans le code, lorsque j'ai une variable qui ne changera jamais et qui garde la même valeur, je la mets entièrement en **majuscule** Ex : private static int **VALUE** = 10

2.6.4 Raisons

Ce petit sous-chapitre les raisons de pourquoi j'utilises ses conventions de nommage et pas d'autres.

1. C'est la manière dont j'ai appris a codé, lors des cours qu'on suivait et lorsque je cherchais des informations sur internet, j'ai vu ces conventions de nommage et je les ai appliquées.

2. Se sont les conventions de nommage que les utilisateurs Java utilise.
   Dans Eclipse, quand on respecte la convention de nommage pour la création des packages ou des fichiers c'est assez restrictif, cependant pour la création des variables c'est plus permissif. (Lien des conventions de nommage Java dans la bibliographie.)

3 Réalisation

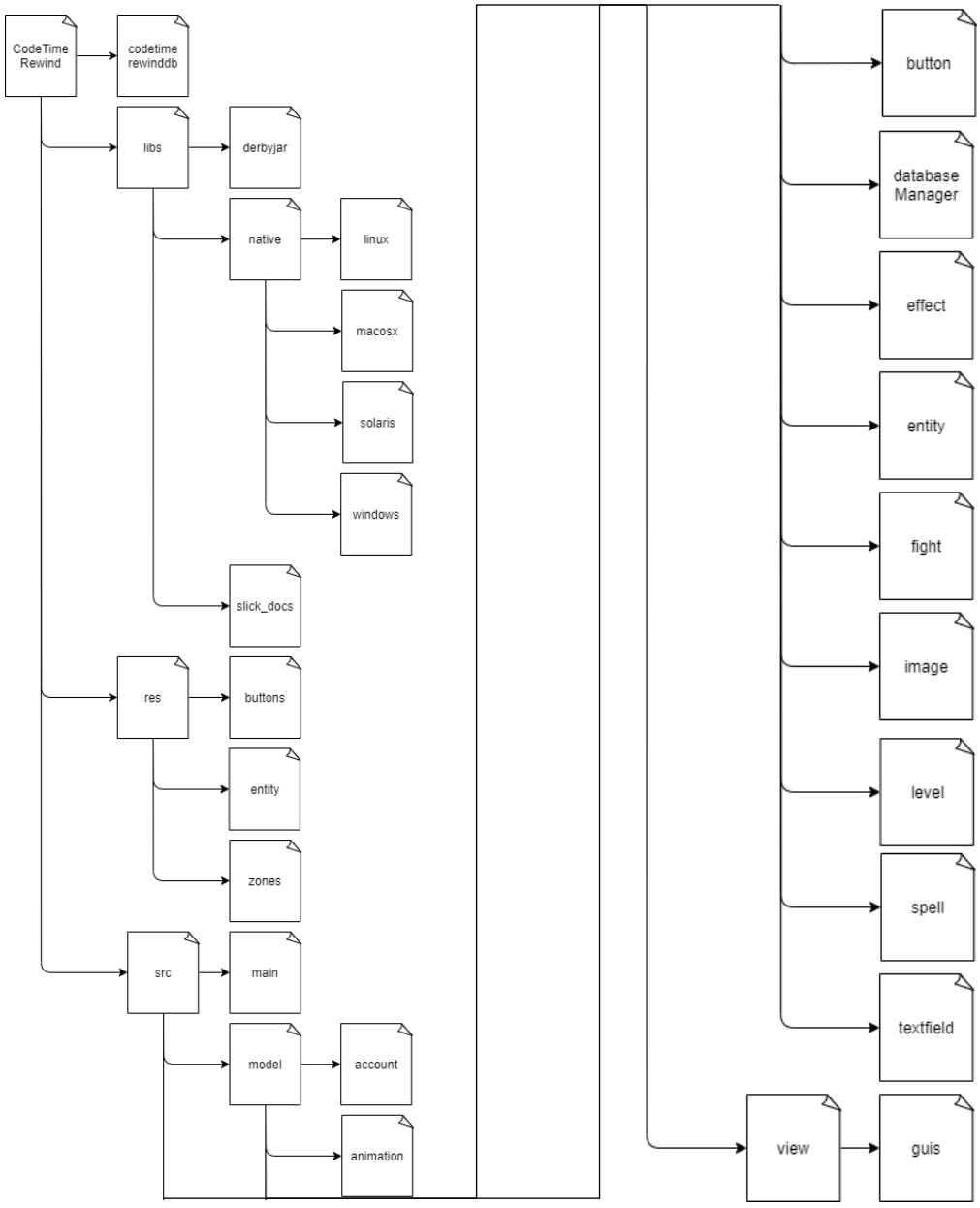
3.1 Dossier de réalisation

3.1.1 Répertoire du logiciel :

Mon Eclipse se trouve dans le dossier :
D:\Eclipe IDE\eclipse

L'entièreté du projet est créée dynamiquement, ce qui veut dire que ça importe peu l'endroit où l'application se trouve, tant que les différents autres dossiers sont avec dans le même dossier, l'application fonctionnera.

3.1.2 Liste des fichiers de mon projet :



3.1.2.1 Code-TimeRewind
Ce dossier est le point principal de mon projet, qui contient toutes les libraires ainsi que mes classes.

3.1.2.2 Codetimerewinddb
Ce dossier est le dossier crée automatiquement au lancement de mon application, il contient ma base de données embarquée.

3.1.2.3 Libs
Ce dossier contient toutes les librairies que j'utilise dans mon projet, l'explication en détail de chaque librairie se trouvera quelque chapitre en dessous.

3.1.2.4 DerbyJar
Ce dossier contient toutes mes librairies concernant ma base de données.

3.1.2.5 Native
Ce dossier contient les natives de mon projet, je définis sous quel OS je veux exécuter mon jeu et les .dll dans le dossier native fera la transcription.

3.1.2.6 Slick-docs
Ce dossier contient les différentes documentations pour la librairie Slick qui est ma librairie principale.

3.1.2.7 Res
Ce dossier contient toutes mes ressources nécessaires au fonctionnement de mon projet, il contient surtout les images de mes personnages, et des décors.

3.1.2.8 Res/Buttons
Ce dossier contient toutes les images pour mes différents boutons dans le jeu, il existe deux types de boutons principaux, le normal et celui qui est pressé qui est plus assombrie que l'autre pour simuler le bouton appuyer.

3.1.2.9 Res/Entity
Ce dossier contient toutes les images de mes différents personnages dans le jeu.

3.1.2.10 Zones
Ce dossier est plus générique parce qu'il contient des conteneurs que j'ai appelé des « zones » qui sont aussi des images mais qui servent beaucoup pour l'affichage de l'expérience ou encore pour mettre un fond aux sorts et aux statistiques.

3.1.2.11 Src
Ce dossier contient toutes mes classes et mon projet en générale.

3.1.2.12 Main
Ce dossier contient les classes principales de mon jeu, il contient le Main qui est le point d'entrée, ainsi que le Game qui est le gestionnaire qui fait la liaison entre toutes les différentes vues.

3.1.2.13 Model

Ce dossier contient les classes qui gère la liaison entre les vues et les données dans la base de données ou les données stockées en générale. (Il sera amélioré pour faire la liaison entre le futur Controller et les données, il y aura plus de lien direct avec les vues après.

3.1.2.14 Model/Account

Ce dossier contient la classe Account, qui est la classe qui va stocker les informations du compte actuel, tel que le mot de passe hashé ou le nom de compte, ainsi que la liste des personnages que le compte possède.

3.1.2.15 Model/Animation

Ce dossier contient les différentes animations en combat, elle permet de faire des combats plus dynamique et plus intéressant que si les persos ne pouvaient pas bouger.

3.1.2.16 Model/Button

Ce dossier contient des boutons pour chacune de mes vues, ça hérite des boutons qui se trouve dans la librairie de Slick2D, la différence c'est que dans ces boutons je définis leur position, leur taille et leur image par défaut.

3.1.2.17 Model/DatabaseManager

Ce dossier contient mon database manager, qui est ma classe qui fait le lien entre la base de données et mon jeu, ça permet d'exécuter différentes requêtes SQL et de mettre à jour la BDD.

3.1.2.18 Model/Effect

Ce dossier contient les différents effets que peut faire les sorts des personnages, tel que des effets actif ou passif.

3.1.2.19 Model/LivingEntity

Ce dossier contient les différentes LivingEntity dans le jeu, tel que les personnages du joueur et les ennemies, et permettent leur gestion.

3.1.2.20 Model/Fight

Ce dossier contient les informations stockées pour faire un combat ainsi que son contrôleur.

3.1.2.21 Model/Image

Ce dossier contient les classes des images qui sont des héritages de la classe Image de Slick2D, elle me permet de donner une taille, une position et de dessiner les images sur la vue.

3.1.2.22 Model/Stage

Ce dossier contient les différentes informations par rapport aux Stages, tel que le nombre d'ennemis dans le stage sa position ou encore dans quel chapitre il se trouve.

3.1.2.23 Model/Spell

Ce dossier contient les différents sorts des personnages, que ce soient des monstres ou des personnages, ils contiennent des sorts avec des effets et des temps de chargement.

3.1.2.24 Model/TextField

Ce dossier contient des text fields, qui sont des champs text que j'utilise pour faire mes bars dans le login et le register qui demande à l'utilisateur de rentrer son nom et mot de passe. Ça hérite de la classe du même nom de Slick2D.

3.1.2.25 View

Ce dossier contient mes différentes vues que l'utilisateur verra lorsqu'il lancera le programme et jouera.

3.1.2.26 Hibernate.cfg.xml

Ce fichier contient tous les paramètres a rentré pour que la liaison avec Hibernate se fasse tel que les classes mappées ou les choix lors de la création de la base de données.

```
<session-factory>
    <property name="connection.driver_class">
        org.apache.derby.jdbc.EmbeddedDriver
    </property>

    <!-- JDBC Database connection settings -->
    <property name="hibernate.connection.url">jdbc:derby:codetimerewinddb;create=true</property>

    <!-- JDBC connection pool settings ... using built-in test pool -->
    <property name="connection.pool_size">1</property>

    <!-- Enable Hibernate's automatic session context management -->
    <property name="current_session_context_class">thread</property>

    <!-- Select our Derby dialect -->
    <property name="hibernate.dialect">org.hibernate.dialect.DerbyTenSevenDialect</property>

    <property name="hbm2ddl.auto">update</property>



    <!-- Names the annotated entity class -->
    <mapping class="model.chapter.Chapter" />
    <mapping class="model.account.Account" />
    <mapping class="model.accountOwnCharacter.AccountOwnCharacter" />
    <mapping class="model.enemyPerStage.EnemyPerStage" />
    <mapping class="model.stage.Stage" />
    <mapping class="model.rarity.Rarity" />
    <mapping class="model.rarity.Legendary" />
    <mapping class="model.rarity.Epic" />
    <mapping class="model.rarity.Rare" />
    <mapping class="model.livingEntity.LivingEntity" />
    <mapping class="model.livingEntity.enemy.Enemy" />
    <mapping class="model.livingEntity.enemy.Zombie" />
    <mapping class="model.livingEntity.enemy.Skeleton" />
    <mapping class="model.livingEntity.enemy.Boar" />
    <mapping class="model.livingEntity.character.Character" />
    <mapping class="model.livingEntity.character.Nomeos" />
    <mapping class="model.livingEntity.character.Guiwi" />
    <mapping class="model.livingEntity.character.Xtreme" />
    <mapping class="model.stageByAccount.StageByAccount" />
</session-factory>
```

3.1.3 Version de mon produit :

La version actuelle de mon jeu est la 1.2 qui est une version stable mais qui comporte encore énormément de bug, lorsque je vais en résoudre plusieurs d'un coup, je vais faire passer le jeu à la version suivante.

3.1.4 Description des librairies utilisées :

3.1.4.1 JUnit
Cette libraire me permet de faire des tests unitaires dans mon projet, ça me permet de tester des fonctions et de vérifier si les tests fonctionnent, ça voudrait dire que mon code est fonctionnel.

3.1.4.2 Lombok
Cette librairie est très pratique, elle me permet de faire en sorte que mes accesseurs et mes setters dans chacune de mes classes soient déjà utilisable sans avoir besoin de les coder ex :

```
@NoArgsConstructor
@Getter
@Setter
public class GuiAdventure extends BasicGameState {
    private int stateId;
    private Image background;
    private Button lobbyButton;
    private Button adventureButton;
```

Pour mes attributs LobbyButton et AdventureButton, je n'aurais pas besoin de déclarer les getters et les setters grâce à Lombok.

3.1.4.3 Lwjgl et lwjgl_util
Ces librairies sont obligatoires pour pouvoir utiliser Slick2D, ça permet de générer certains graphismes et certaines options.

3.1.4.4 Native-linux, Native-mac et native-Windows
Ces libraires me permettent de définir sous quel Os je veux que mon programme s'exécute.

3.1.4.5 Slick
C'est la librairie principale de mon programme, c'est une librairie spécialement faite pour les jeux en 2D, elle gère la boucle du jeu, les graphismes, les sons, les vues etc…

3.1.4.6 Suite Spring
La suite Spring ce sont des librairies qui me permettent de gérer l'encryptions de mon mot de passe et son dé encryptions.

3.1.4.7 Derby
Les librairies Derby c'est celle qui permettent de faire fonctionner ma base de données embarquée en gérant les requêtes SQL et les liens avec les différents drivers la faisant fonctionner.

3.1.4.8 Hibernate
Les librairies Hibernate font le lien entre mon application et ma base de données en convertissant des objets en requête SQL au travers d'annotations rajoutées dans les classes.

```
@Entity
@Table(name = "Accounts")
public class Account {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Account_Id")
    private int id;

    @Column(name = "Account_Level")
    private int account_Level;
```

3.2 Choix effectué durant le projet

3.2.1 Choix de la création de la base donnée

Durant l'implémentation de ma base de données, j'ai dû choisir la manière dont je voulais la créer ainsi que ses tables.

J'avais le choix entre différents moyens.

- Générer les tables à la sauvegarde des entités.
- Créer un script SQL pour créer les tables au lancement de l'application.
- Créer une génération automatique de mes tables selon les entités mappées.

3.2.1.1 Première option

Pour cette première option, c'est une fonctionnalité de Hibernate qui permet que toute les classes qui sont **mappées** et qui sont notées d'un @Entity d'être directement crée dans la base de données lorsqu'on fait une insertion dans la base de données. Ex : Si nous voulons enregistrer l'user « Paul », on a pas besoin d'avoir la table crée, elle se crée automatiquement quand on fera une insertion de Paul.

```
@Entity
@Table(name = "Accounts")
public class Account {
```

3.2.1.2 Deuxième option

Cette deuxième option c'est plutôt simple, pouvoir créer un script SQL et modifié la configuration de Hibernate pour que le script s'exécute pour la création de la base de données.

3.2.1.3 Troisième option

Cette solution permet de créer directement les tables au lancement du programmes selon les classes qui sont mappées et annotées d'un @Entity cependant, par rapport à la première option, ça a pas besoin d'insérer des données pour que les tables se créent.

```
<property name="hbm2ddl.auto">update</property>
```

3.2.1.4 Résultat

J'ai choisi d'utiliser la **troisième option,** tout simplement parce que j'ai testé les deux autres avants et que j'ai eu plein de problème qui m'ont fait choisir cette option, la **première option** a pas fonctionné parce que la première commande que je fais c'est vérifier si l'utilisateur existe et les selects ne créent pas les tables donc il cherchait dans une table qui existait pas. La **seconde option**, qu'importe ce que j'écrivais comme requête SQL, mon drop table fonctionnait bien mais pas mon create table et ça m'a pris beaucoup de temps à débugger pour pas trouver la solution.

3.3 Description des tests effectués

3.3.1 Différents tests :

3.3.1.1 Connexion à la base de données

Ce test permet de vérifier si la connexion à la base de données est possible.

Résultat :
Si la connexion est possible, ça return l'état de la base de données et met fin au test.
Si la connexion est impossible, ça lance une « SlickException » et met fin au test.

3.3.1.2 Insertion des différents niveaux

Ce test permet de vérifier si on peut insérer les différents niveaux dans la base de données (les niveaux faisant énormément de jonction entre plusieurs tables).

Résultat :
Si l'insertion est réussie, ça return un booléen TRUE et ça met fin au test.
Si l'insertion a échouée, ça return un booléen FALSE et ça met fin au test.

3.3.1.3 Vérification d'un test de connexion à la base de données coté utilisateur

Ce test permet de vérifier si un utilisateur arrive bien, à se connecter avec son compte.

Résultat :
Si la connexion est réussie, ça return un booléen TRUE et ça met fin au test.
Si la connexion a échoué, ça return un booléen FALSE et ça met fin au test.

3.3.1.4 Insertion des différents personnages existants

Ce test permet de vérifier si on peut insérer les différents personnages dans la base de données (Les personnage ayant beaucoup de jonction entre plusieurs tables).

Résultat :
Si l'insertion est réussie, ça return un booléen TRUE et ça met fin au test.
Si l'insertion a échouée, ça return un booléen FALSE et ça met fin au test.

3.3.2 Tests effectués par des externes

Comme cité dans ma stratégie de test, je vais faire tester mon jeu par différentes personnes externe au projet pour voir si c'est intuitif pour eux, si le manuel d'utilisation est bien écrit et compréhensible et s'ils arrivent à faire tourner le jeu et a y joué.

3.3.2.1 Date : 18.05.2021 :

Mathieu Rabot	
Machine utilisée :	**Ordinateur personnel** **Processeur** : i7-1165G7 11th Gen **Carte graphique** : Intel Iris Xe Graphics **Ram** : 16Go
Résultat attendu par le développeur :	Le mode Aventure a crash quand il a sélectionné son chapitre
Avis du testeur :	Faudrait que j'arrive à trouver le temps pour bosser sur la view « Character » des personnages sinon on peut pas voir les personnages qu'on a acquis.

Guilherme De Oliveira Calhau	
Machine utilisée :	**Ordinateur du CPNV** **Processeur** : i7-6700 **Carte graphique** : Intel HD Graphics 530 **Ram** : 16Go
Résultat attendu par le développeur :	Le mode Aventure a crash quand il a sélectionné son chapitre
Avis du testeur :	Pour l'instant le jeu marche pas mais il est beau. « L'aventure ne peut pas être lancée »

Sophia Laloux	
Machine utilisée :	**Ordinateur personnel** **Processeur** : i7-10700F 10th Gen **Carte graphique** : GTX 1050 TI 4Go **Ram** : 32Go
Résultat attendu par le développeur :	Le mode Aventure a crash quand elle a sélectionné son chapitre
Avis du testeur :	Un peu de latence mais les fonctionnalités fonctionnent, dommage du manque d'animation dans Summon.

3.4 Erreurs restantes

Ce chapitre définis les différentes erreurs qui reste dans mon projet qui fait que mon jeu n'est pas encore totalement jouable, ça contient aussi les améliorations futures que je vais devoir faire pour résoudre le problème.

3.5 Liste des documents fournis

Comme demandé dans mon dans mon cahier des charges, les documents fournis et autres livrable sont les suivants :

- Une planification initiale (disponible plus haut dans le rapport)
- Un rapport de projet
- Un journal de travail (disponible plus bas dans le rapport)
- Le code source du jeu (fournis via un dépôt GitHub)
- L'exécutable/l'installateur du produit fini

4 Conclusions

4.1 Objectifs atteints ?

4.2 Points positifs / négatifs

4.3 Difficultés particulières

4.4 Suite pour le projet

5 Annexes

5.1 Aide reçu lors du projet

Ce chapitre va regrouper toutes les aides externes que j'ai reçu lors du projet et en quoi elles m'ont aidées.

04.05.2021 - Aide de Mr. Viret par rapport à une question posée sur la planification initiale si elle doit contenir le rapport de projet.

06.05.2021 – Aide de Mr. Viret pour une incompréhension du cahier des charges et sur les ORMs.

17.05.2021 – Aide de Guilherme Calhau et Mauro Dos Santos pour la création de personnage pour mon jeu (ils ont créé un personnage chacun).

20.05.2021 – Aide de Mr. Viret concernant des bugs par rapport à Hibernate et sur la suite de mon projet, définir l'axe dans lequel je vais devoir continuer.

5.2 Sources – Bibliographie – Acquisition des connaissances

Lien du site :

https://hibernate.org/orm/

Date de consultation : 04 mai 2021

Utilisation :

Utiliser pour pouvoir comprendre ce qu'était qu'un ORM et ce que Hibernate faisait en particulier en tant qu'ORM en regardant la documentation.

Lien du site :

https://www.youtube.com/playlist?list=PLEAQNNR8IlB7fNkRsUgzrR346i-UqE5CG

Date de consultation : 04 mai 2021

Utilisation :

Playlist de vidéo pour introduire Hibernate et son utilisation au cours d'un petit projet, vidéo très intéressante que je regarde en boucle pour bien comprendre et réutiliser ses informations.

Lien du site :

http://remy-manu.no-ip.biz/Java/Tutoriels/J2EE/ORM.html

Date de consultation : 04 mai 2021

Utilisation :

Enorme site contenant plein d'information sur Hibernate et l'utilisation de ses annotations, cependant il est légèrement illisible et c'est dur de trouver les informations qu'on veut sans index.

Lien du site :

https://www.youtube.com/watch?v=Zj1mRWfhx_Y
Date de consultation : 06 mai 2021

Utilisation :

Vidéo YouTube qui explique l'héritage avec Hibernate qui a répondu à quelque question que j'avais et que je regarde assez souvent.

Lien du site :

https://www.youtube.com/watch?v=_vhskxEihz4&t=1575s

Date de consultation : 06 mai 2021

Utilisation :

Vidéo YouTube qui explique l'utilisation des tables de jonctions avec Hibernate que je regarde aussi très souvent.

Lien du site :

https://www.loribel.com/java/normes/nommage.html#:~:text=Les%20noms%20des%20classes%20doivent,des%20noms%20simples%20et%20descriptifs

Date de consultation : 18 mai 2021

Utilisation :

Convention de nommage Java pour vérifier si mon projet était bien en ordre.

5.3 Journal de travail

Jour	Semaine	Heures	Type	Description	Liens		
03.mai	1	8h30 - 9h00	Analyse	Entretien avec les deux **experts** pour expliquer le début du projet et son déroulement	-	-	-
03.mai	1	9h00 - 9h30	Documentation	Création de la **planification initial** et début de son remplissage pour la première semaine	-	-	-
03.mai	1	9h30- 9h35	Documentation	Création du **Journal de travail**	-	-	-
03.mai	1	9h35- 9h55	Documentation	Remplissage des trois premières tâches du journal de travail	-	-	-
03.mai	1	9h55 - 10h20	Documentation	Suite de la création de la planification initiale en ajoutant les différentes semaines et dates	-	-	-
03.mai	1	10h20 - 11h10	Documentation	Finition de la planification initiale	-	-	-
03.mai	1	11h10 - 11h30	Documentation	Reprise du **rapport de projet** du pré-TPI et mise à jour des **objectifs** pour les faire correspondre au nouveau cahier des charges	-	-	-
03.mai	1	11h30 - 12h15	Documentation	Mise à jour de la planification et des difficultés à venir dans le **rapport de projet** 1.2.1 et 1.2.2	-	-	-
03.mai	1	13h30 - 15h	Implémentation	Recommencer les changements d'architecture que j'avais effectué avant le TPI	-	-	-
03.mai	1	15h15 - 16h00	Implémentation	Recommencer les changements d'architecture que j'avais effectué avant le TPI	-	-	-
04.mai	1	9h50 - 10h45	Analyse	Recherche ce que c'est que Hibernate et qu'est-ce qu'un **ORM**	Lien.	-	-
04.mai	1	10h45 - 11h30	Implémentation	Tentative d'implémentation de Hibernate dans mon projet	Lien.		-
04.mai	1	? - ?	Documentation	Documenter l'utilisation d'un ORM et Hibernate	-	-	-

04.mai	1	11h30 - 12h15	Analyse	Recherche ce que c'est que Hibernate et qu'est-ce qu'un ORM	Lien.	-	-
04.mai	1	13h30 - 16h	Implémentation	Recommencer les changements d'architecture que j'avais effectué avant le TPI une deuxième fois	-	-	-
06.mai	1	8h00 - 9h35	Analyse	Essayer de comprendre comment **Hibernate** fonctionne	Lien. Inheritance	Lien.	Lien. Join Table
06.mai	1	9h35 - 9h45	Analyse	Session avec Mr. Viret pour discuter du cahier des charges et de Hibernate	-	-	-
06.mai	1	9h50 - 10h45	Implémentation	Restructuration de mon architecture de code sur Eclipse	-	-	-
06.mai	1	10h45 - 12h15	Implémentation	Essayer d'implémenter **Hibernate** dans mon projet	-	-	-
06.mai	1	13h30 - 17h	Implémentation	Essayer d'**implémenter Hibernate** dans mon projet en changeant la version de mon Eclipse et en implémentant la base de données dans Eclipse	-	-	-
07.mai	1	8h00 - 9h50	Implémentation	**Débugger** mon environnement de travail du a l'implémentation de Hibernate	-	-	-
07.mai	1	9h50 - 10h45	Implémentation	Changement du databaseManager pour correspondre à Hibernate et a ses annotations	-	-	-
07.mai	1	10h45 - 12h15	Implémentation	Changement des classes Level en Stage et création des relations entre les tables Hibernate	-	-	-
07.mai	1	13h30 - 15h	Documentation	Documenter l'utilisation d'un ORM et Hibernate et préparer mon cahier des charges pour le premier rendu du projet	-	-	-
10.mai	2	08h00 - 9h35	Implémentation	Continuer sur les changements de mon databaseManager et de mon model pour faire fonctionner Hibernate	Lien One To Many	Lien Create Query	-
10.mai	2	9h50 - 10h45	Implémentation	Continuer sur les changements de mon databaseManager et de mon model pour faire fonctionner Hibernate	-	-	-

10.mai	2	10h45 - 12h15	Implémentation	Créer les scriptes SQL de la création de la base de données et sa suppression	-	-	-
10.mai	2	13h30 - 14h15	Implémentation	Abandonner l'idée des scriptes et changer mon fichier de configuration Hibernate pour générer les tables automatiquement selon les entités mappées.	Lien Hibernate hbm2ddl	-	-
10.mai	2	14h15 - 15h00	Documentation	Documenter mon changement de stratégie pour ma base de données dans le rapport de projet. Chapitre 3.2	-	-	-
11.mai	2	9h50 - 10h30	Implémentation	Finir de changer mon databaseManager pour le faire fonctionner avec Hibernate et tester de faire se connecter et s'enregistrer un user	-	-	-
11.mai	2	10h30 - 11h00	Analyse	Changer les maquettes pour ajouter la nouvelle view d'invocation d'héros	-	-	-
11.mai	2	10h30 - 11h00	Documentation	Intégré les changements de maquette dans la gestion des maquettes Chapitre 2.1.1	-	-	-
11.mai	2	11h00 - 12h15	Documentation	Modifier le MCD et le MLD pour correspondre à Hibernate et ajouter la nouvelle table des raretés des personnes dedans (Rare, Epic, Legendary)	-	-	-
11.mai	2	13h30 - 14h30	Implémentation	Changer le databaseManager pour inclure le nouveau système de rareté sur les personnages et commencé à créer la vue dans le code en implémentant le GuiSummon et le SummonController.	-	-	-
11.mai	2	14h30 - 15h00	Analyse	Créer les nouveaux boutons que l'ajout de l'invocation demande (Un pour invoquer et l'autre pour se déplacer vers la vue Summon).	-	-	-
11.mai	2	15h15 - 16h00	Implémentation	Début de la création du système d'obtention de personnage dans mon SummonController.	-	-	-

17.mai	3	8h00 - 9h00	Documentation	Commenter mon choix pour la création de la base de données dans le Chapitre 3.2, ajouter une explication de ce qu'est le fichier de configuration de Hibernate 3.1.2.26 et ajouter l'explication des classes mappées dans le Glossaire.	-	-	-
17.mai	3	9h00 - 9h35	Implémentation	Ajouter les sprites pour les nouveaux persos Guiwi et Xtreme	-	-	-
17.mai	3	9h50 - 10h55	Implémentation	Continuer de créer le système d'obtention de personnage et afficher les sprites des personnages obtenus.	-	-	-
17.mai	3	10h55 - 11h30	Implémentation	Créer l'ajout des personnages obtenus dans la base de données.	-	-	-
17.mai	3	11h30 - 12h15	Implémentation	Finir le système d'obtention de personnage et leur enregistrement dans la base de données.	-	-	-
17.mai	3	13h30 - 14h30	Tests	Essayer de réparer l'exportation de mon projet	-	-	-
17.mai	3	14h30 - 15h00	Analyse	Entretien avec le deuxième expert pour parler de l'état actuel du projet	-	-	-
17.mai	3	15h00 - 16h00	Tests	Essayer de réparer l'exportation de mon projet	-	-	-
18.mai	3	9h50 - 10h30	Tests	Lancer la build de mon projet et le faire tester sur une autre machine (ça fonctionne)	-	-	-
18.mai	3	10h30 - 11h15	Documentation	Mettre à jour le rapport de projet pour le manuel d'utilisation et les archives du projet, ainsi que créer les wikis dans GitHub.	Lien wiki UserGuide	Lien wiki ExportationProcess	-
18.mai	3	11h15 - 12h15	Documentation	Ajouter les tests effectués par des externes dans mon rapport de projet et spécifiant les personnes, leurs machines et le résultat de leur test. 3.3.2.1	-	-	-

18.mai	3	13h30 - 15h00	Implémentation	Commencer à débugger la view "Characters" pour préparer le terrain au système d'équipement qu'il y aura dans cette vue	-	-	-
18.mai	3	15h00 - 15h05	Documentation	Changement de la maquette "Characters" pour introduire les multiples personnages et la changer dans la documentation	-	-	-
18.mai	3	15h20 - 16h05	Documentation	Créer une section pour la convention de nommage dans le rapport de projet et pourquoi j'utilise certaine convention de nommage	Lien convention de nommage Java	-	-
20.mai	3	8h00 - 8h45	Documentation	Essayer de débugger la view Character pour préparer l'arrivée des équipements	-	-	-
20.mai	3	9h20 - 9h35	Analyse	Entretien avec le chef de projet sur l'avancement de mon projet, regarder quelques bugs et définir ce qu'il me reste à faire	-	-	-
20.mai	3	9h50 - 11h00	Implémentation	Essayer de débugger la view Adventure pour préparer à la fin des combats	-	-	-
20.mai	3	11h00 - 12h15	Implémentation	Essayer de débugger la view Adventure pour préparer à la fin des combats	-	-	-
20.mai	3	13h30 - 17h00	Implémentation	Essayer de débugger la view Adventure pour préparer à la fin des combats	-	-	-
21.mai	3	08h00 - 9h35	Implémentation	Essayer de débugger les combats pour que les personnages puissent s'affronter et donc finir le combat	-	-	-
21.mai	3	9h50 - 12h15	Implémentation	Essayer de débugger les combats pour que les personnages puissent s'affronter et donc finir le combat	-	-	-
21.mai	3	13h30 - 14h10	Implémentation	Finir de débugger les combats pour que la vue de fin des combats arrive.	-	-	-

21.mai	3	14h10 - 15h00	Documentation	Préparer mon livrable de la semaine en mettant a jour mon rapport de projet et mon journal de travail.	-	-	-

5.4 Manuel d'Installation

Pour accéder au manuel d’installation du jeu, il faut juste cliquer sur ce Lien.

5.5 Archives du projet

Pour accéder à mon projet directement prêt à être utilisé, donc à son installeur, il faut accéder à ce **lien** et télécharger la dernière release disponible.
Il faut seulement télécharger le fichier .zip du projet qui contient un dossier « Res » qui sont les ressources du projet comme les images et un fichier Java qui est le jeu en lui-même.