Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Pôle TIC - Filière Télécommunication

Projet de semestre 5

2017-2018

Virtual Memory

Rapport du projet

Étudiants :	Dufresne Loïc
	Stulz Nicolas
Professeurs :	Bruegger Pascal
	Kilchoer François
Proposé par :	Gaëtan Collaud, chez Softcom

Hes·so FRIBOURG FREIBURG
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Fribourg, 1 février 2018

Remerciements

Nous remercions toutes les personnes qui nous ont aidés à réaliser ce projet de semestre 5.

Nous adressons, tout particulièrement, nos sincères remerciements à nos superviseurs, Monsieur Pascal Bruegger et Monsieur François Kilchoer, et à notre mandant, Monsieur Gaétan Collaud, qui nous ont guidés et conseillés tout au long de la réalisation de ce projet, et aussi pour leurs disponibilités. Merci beaucoup !

Nous tenons aussi à remercier chaleureusement, Madame Houda Chabbi, pour ces précieux conseils qui nous ont beaucoup aidés pour la partie de l'analyse des bases de données et Monsieur François Buntschu pour la mise en place de notre machine virtuelle permettant le déploiement de notre site Web.

Un grand merci aux personnes qui ont testé le Virtual Memory lors de la phase des tests utilisateurs, vos remarques nous ont été d'une grande aide pour la suite du projet.

Table des matières

Historique du document

Version	Étudiants	Changements	Date
1.0	Loïc Dufresne	Création du document	11.10.2017
2.0	Nicolas Stulz	Passage à LaTeX	23.10.2017
2.0	Loïc Dufresne - Nicolas Stulz	Premier rendu du document	13.11.2017
2.1	Loïc Dufresne - Nicolas Stulz	Rendu de la partie analyse du document	15.11.2017
2.2	Loïc Dufresne - Nicolas Stulz	Rendu de la partie conception du document	6.12.2017
2.3	Loïc Dufresne - Nicolas Stulz	Rendu de la partie implémentation du document	17.01.2018
2.4	Loïc Dufresne - Nicolas Stulz	Rendu finale du document	01.02.2018

1 Introduction

1.1 Contexte

De nos jours, il est de plus en plus fréquent lors d'un entretien d'embauche, que les RH fassent passer des tests à leurs recrues en posant des questions sous forment de petit quizz en relation au job proposé.

C'est pourquoi notre mandataire nous demande de réaliser une sorte de quizz à choix multiples complètement paramétrable afin d'en faire un outil professionnel d'aide au recrutement. Pourquoi ne pas aussi en faire un outil d'apprentissage ou tout simplement un jeu pour s'amuser entre amis.

Le but de ce projet est de créer le template de ce quizz, facilement paramétrable afin qu'il puisse être adapté pour toute sorte de domaines.

Le mode permettant de jouer au quizz sera facile à administrer et fonctionnel pour toutes sortes de contextes (jeux, cours d'histoire, ressources humaines, etc.). Le mode permettant d'administrer ce quizz permettra de créer de nouvelles relations et modifications de ces relations afin d'alimenter le jeu à tout moment. La sécurité du site Web n'est à ce jour pas indispensable.

1.2 Objectifs

Les objectifs principaux à atteindre pour ce projet sont :

- Mettre en place un serveur Web permettant l'utilisation de « Spring Boot » pour réaliser une API REST
- Analyser et choisir les technologies afin de réaliser ce projet
- Réaliser une base de données permettant d'accueillir les données de l'application Web
- Réaliser une application Web responsive avec Angular CLI contenant un mode permettant de jouer au quizz et un mode permettant d'administrer le quizz
- Réaliser un design simple pour l'application Web

Les objectifs optionnels du projet :

- Réaliser un système de login pour le mode permettant d'administrer le quizz
- Réaliser un système de login pour les clients/joueurs du quizz
- Réaliser un système de sauvegarde des résultats des clients/joueurs du quizz
- Améliorer l'expérience utilisateur

2 Analyse

Ce chapitre traite de l'analyse de notre projet. Cette analyse permettra au mandataire ou aux personnes qui reprendront le projet de bien comprendre son but et de quelle manière il va être mis sur pieds. Nous allons concevoir ce projet de A à Z, aucune analyse, conception ou implémentation antérieurs n'a été réalisé.

Comme expliqué précédemment, notre projet sera représenté par un jeu, un "Virtual Memory", nous décrirons le fonctionnement de ce jeu et quelles technologies seront mises en place pour le réaliser. Certaines technologies devront être analysées afin de choisir la meilleure manière de mettre en place notre application, d'autres technologies seront imposées par le mandant. Nous ferons aussi des recherches sur les applications existantes reproduisant de près ou de loin le fonctionnement de notre jeu.

Dans l'état de l'art, nous décrirons les applications similaires ou identiques à ce que nous voulons réaliser. Les technologies que nous pourrons utiliser afin de réaliser notre projet seront citées, en plus de celles imposées afin d'avoir une bonne vue d'ensemble. Au niveau de la base de données, les différentes techniques afin de la créer seront nommées.

2.1 État de l'art

D'après nos recherches, dans la plupart des quizz, il y a une question, des fausses réponses et la bonne réponse codée en dur. Les réponses peuvent apparaître dans un ordre différent, mais ce sont très vite toujours les mêmes réponses. Nous avons classé les quizz en 2 grands groupes. L'un regroupe les quizz avec des images ou du texte, et l'autre regroupe les templates permettant de créer des quizz.

2.1.1 Groupe 1: Jeux

Nous traitons ici les différents quizz permettant de jouer en ligne. Nous regardons les différents modes de jeu, le gameplay, comment sont utilisées les images. Les images font partie de la question ou sont-elles des réponses? Est-ce que les questions ou les réponses sont mélangées entre texte et images?

Topquizz sur `https://www.topquizz.com/` propose pas mal de thèmes comme Musique, Géographie, Sport, etc. Il y a même un mode jeu de Blind Test, c'est à dire que le joueur entant une chanson et doit répondre, soit qui la chante, soit quel est le titre de la chanson parmi 4 réponses sous forme de texte. Il y a aussi un mode qui demande au joueur de trouver "Quelle marque est-ce? (avec le logo sous forme d'image) et il doit sélectionner la réponse sous forme du nom de la marque parmi 4 réponses possibles. Le site nous propose aussi une liste de quizz en fonction des plus populaires, des plus récents, au hasard, les plus faciles ou difficiles.

Ce site offre la possibilité de créer des quizz, mais pour cela nous devons nous enregistrer.

Nous avons joué plusieurs fois au même quizz et ce sont toujours les mêmes questions dans le même ordre qui sortent.

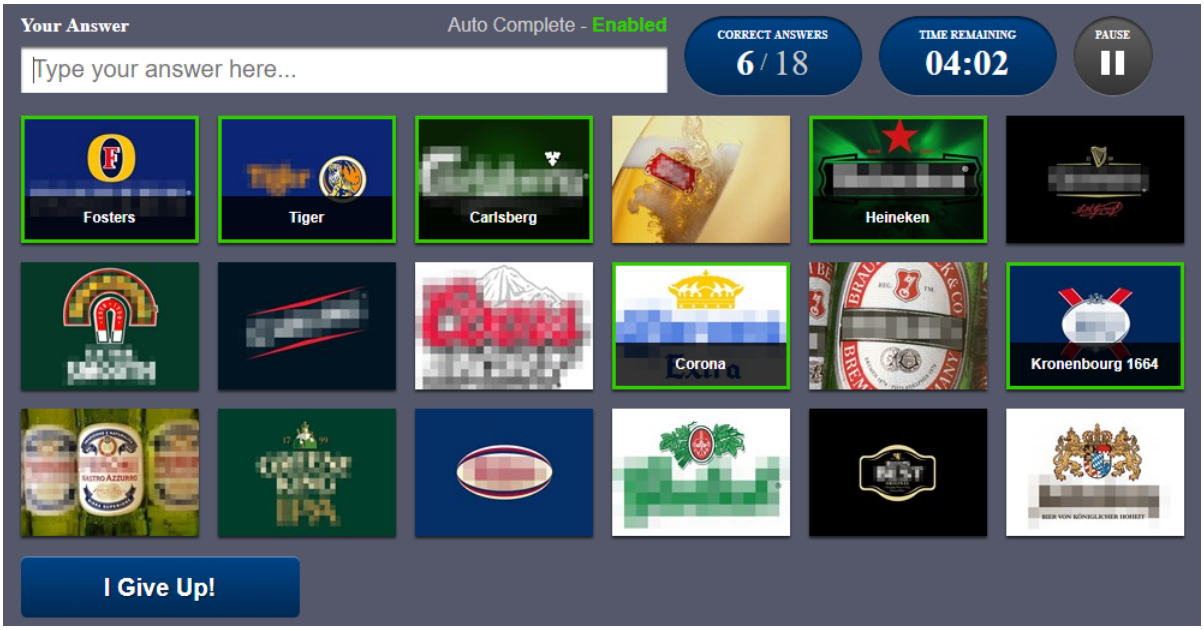


Figure 1 – Déroulement d'un jeu sur `http://www.quizfactor.com`

Nous avons sélectionné un exemple intéressant qui est `www.quizfactor.com` que nous pouvons voir dans la figure 1. Ce site propose des quizz avec des images comme le montre la figure 1. Dans ce mode de jeu, nous devons trouver le nom des images, comme ici avec des marques de bières. Nous pouvons choisir une catégorie par exemple "Animals and Nature" ou "Food and Drinks". De plus, ce quizz offre différents modes de jeux tels que des vrais-faux, et des questions-réponses à choix multiples. Il y a parfois la possibilité de choisir un mode de difficulté. Toutes les questions sont les mêmes à chaque fois que l'on rejoue, il n'y a apparemment pas de générations aléatoires. Nous ne connaissons pas le langage utilisé par le serveur. Du côté client, nous voyons l'utilisation de JavaScript.

2.1.2 Groupe 2: Créateur de quizz

Nous traitons ici les applications en ligne permettant de créer des quizz sans offrir la possibilité d'y jouer.

`https://www.quiz-maker.com/` permet de créer des quizz simplement pour un prix entre 15 et 99 USD par mois. Le prix varie entre le nombre d'utilisateurs et le nombre de quizz créer essentiellement. Il permet de créer des quiz à choix multiples ou à un seul choix. L'utilisateur doit créer la question en dur et proposer différentes réponses en indiquant la bonne.

`https://www.onlinequizcreator.com/fr/` permet de créer des quizz pour un prix entre 18 et 89 euros par mois ou 910 euros par année. Il propose une version gratuite très limitée telle que 15 questions par quizz, 100 jeux jouables par mois, type de questions: choix multiples, texte, libre, à trous. La formule la plus chère offre beaucoup plus de possibilités comme des quizz au design responsive et un nombre illimité de quizz. Ce créateur de quizz est utilisé par des sociétés comme Dell, Heineken ou encore Toyota.

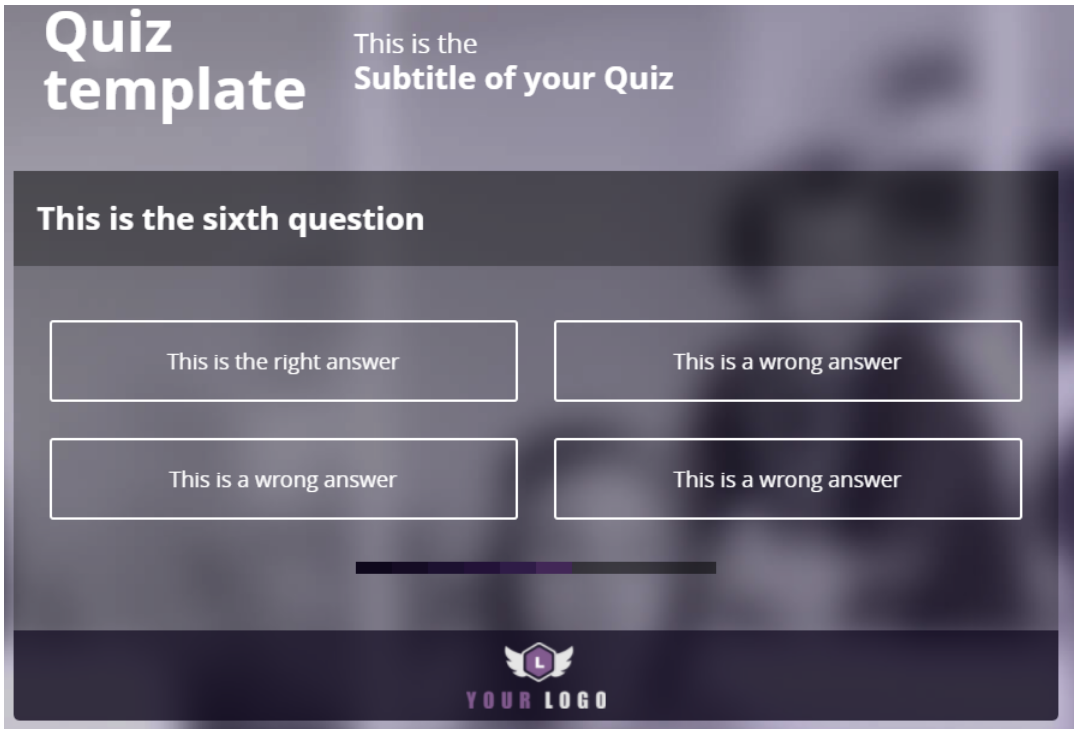


Figure 2 – Template de quizz sur `https://www.dot.vu`

Ce cite propose de créer des expériences interactives, en effet ils vendent divers templates qui permettent de créer un quizz, de le publier et de traiter les résultats. Parmi les quizz possibles, on peut utiliser des images comme réponses. Par contre, les questions et réponses sont toujours fixes, donc pas de côté aléatoire. L'utilisation de cette plateforme coûte entre 120 et 800 euros par mois.

Même si nous ne nous occupons pas de l'aspect économique, les exemples ci-dessus, prouvent que notre application a un potentiel économique certain.

2.1.3 Notre application

Notre application se différencie de différentes façons, la première étant qu'au niveau jeu, les questions seront générées aléatoirement en fonction de nos données stockées.

La deuxième étant qu'au niveau de l'administration, l'administrateur peut ajouter des entités ainsi que des attributs à ces entités, une image ou pas. Une fois plusieurs entités stockées, l'administrateur, en créant un nouveau quizz, sélectionne la question du quizz et les réponses vont être choisies aléatoirement.

2.2 Technologies

Afin de mieux comprendre, le fonctionnement et les différentes technologies présentent dans notre projet, voici un schéma comprenant les technologies imposées par le mandant :

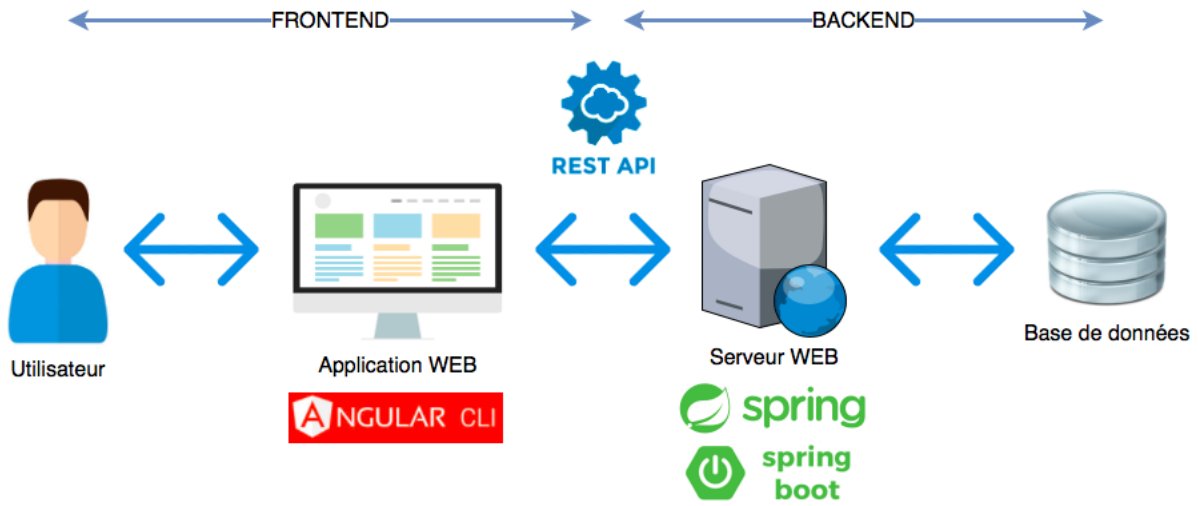


Figure 3 – Schéma des différentes technologies de notre projet

Le schéma précédent est divisé en deux parties :

- **Le Frontend :** Désigne la partie du projet visible à l'écran, avec laquelle l'utilisateur peut interagir. Cette partie est divisée en deux domaines, le design et le développement. Le design est conçu grâce à des maquettes et développé grâce à des langages comme HTML, CSS ou encore JavaScript.
- **Le Backend :** Désigne la partie invisible pour l'utilisateur, la plus grosse partie du projet. Cette partie est divisée en trois parties, le serveur, l'application et la base de données. Le serveur où les pages de l'application sont hébergées et accessible à tout moment, l'application hébergée sur le serveur développée avec des langages de programmation dynamique comme PHP, Python, SQL, et la base de données où sont stockées les données de l'application.

Les données de notre jeu seront stockées dans une base de données, le type de base de données reste à définir, nous analyserons différents types de base de données (relationnel, orienté documents, orientés objets...) afin de peser le pour et le contre et faire un choix.

Le framework Spring Boot nous facilitera la configuration d'un projet Spring afin de permettre l'accès aux données à notre application. L'outil Angular CLI nous permettra de créer une application en réalisant toutes les tâches les plus courantes de développement. La notion d'API REST est aussi présente d'un notre projet, nous y reviendrons.

2.3 Contraintes

Comme cité précédemment, le projet comporte quelques contraintes, en effet certaines technologies ou notions sont imposées pour la mise en place de l'infrastructure, comme le framework Spring Boot ou l'outil Angular CLI qui aide au développement avec le framework Angular.

2.3.1 Spring Boot

Spring Boot est un framework Java qui nous facilitera la configuration d'un projet Spring et la création rapide de la structure du projet afin que le projet soit prêt à être déployé en environnement de production. Il apporte de nombreuses fonctionnalités comme des aspects Web, sécurité où l'accès aux données, les aspects annexes comme la configuration, les tests, le déploiement sont automatisés et les problématiques techniques sont prises en charge au maximum par Spring Boot, afin que le développeur puisse se concentrer sur le coeur du projet. Au niveau du déploiement applicatif, Spring Boot offre la possibilité d'intégrer directement un serveur Tomcat afin de faire tourner l'application. Spring Boot n'a pas pour objectif d'apporter de nouvelles fonctionnalités ou solutions dans ce domaine, mais simplifie l'utilisation de technologies déjà existantes.

Ce qui nous intéresse le plus dans ce framework est le côté accès aux données, Spring Boot permet d'intégrer une base de données à notre projet très simplement.

Si Java est supporté par un OS SpringBoot l'est aussi.

2.3.2 Angular CLI

Angular CLI est un outil en ligne de commande qui aide au développement avec le framework Angular. Il est proposé et développé par Google et permet de réaliser les tâches de développement les plus courantes, il permet de générer un projet déjà configuré. Angular CLI permet de créer une application "from scratch", de créer l'architecture d'un projet, de lancer l'application dans le serveur Web, de lancer des tests complets, de fournir des plugins.

Angular est compatible avec les différents browsers suivants selon le support d'Angular [1]: Chrome (latest), Firefox(latest), Edge 13 et plus, IE 9 et plus, Safari 7 et plus, Android 4.1 et plus et IE Mobile 11.

La version 1 du framework Angular ou parfois appelée AngularJS permet de coder en JavaScript. Alors qu'à partir de la version 2, nous devons coder en TypeScript. Ce dernier est un langage développé par Microsoft qui permet d'améliorer et de sécuriser le code JavaScript. Ce langage est compilé en JavaScript et est donc supporté par tous les navigateurs récents. Angular4 est en fait la version 2.4 et non pas 4.

2.3.3 Responsive Web design

Une application Reb responsive permet d'adapter le contenu d'un site Web en fonction de la taille de l'écran sur lequel nous y sommes connectés. La figure 4 est un très bon exemple visuel d'une application responsive.

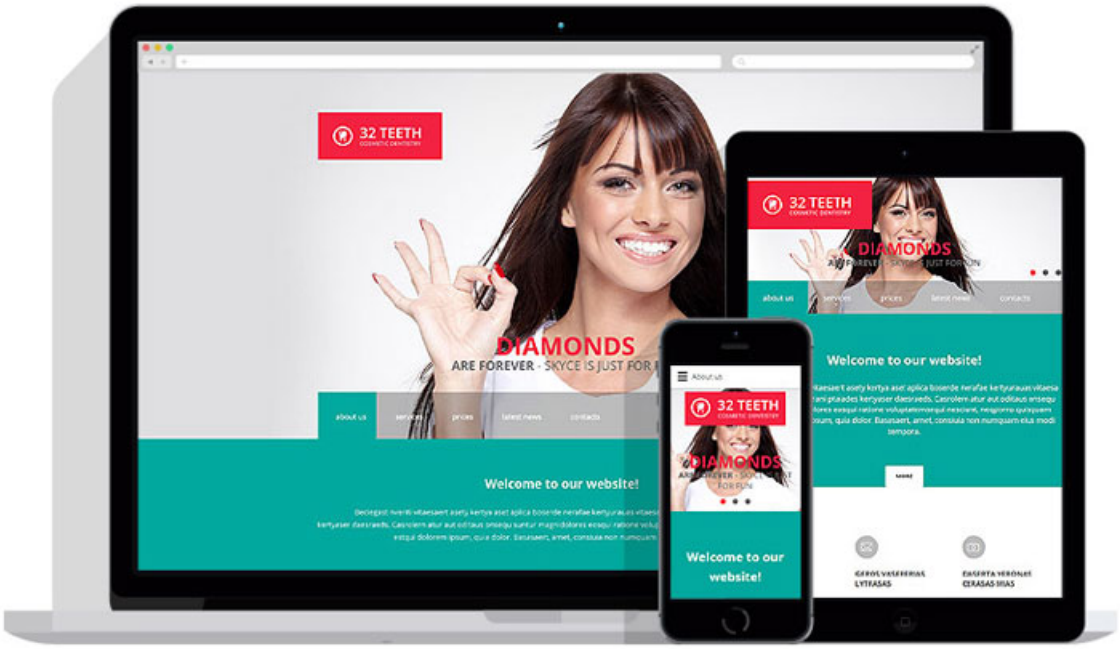


Figure 4 – Exemple de responsive design. [2]

À la demande du mandant, la partie de notre application Web permettant de jouer au quizz devra être responsive, en revanche, la partie permettant d'administrer le quizz ne sera pas forcément responsive, car l'administration se fera généralement sur un poste fixe.

Le Responsive Web design est une approche de la conception Web, il vise à optimiser l'expérience utilisateur lors de la lecture ou de la navigation de l'utilisateur sur les pages des sites Web. Une page Web n'aura pas les mêmes dimensions sur un ordinateur fixe ou un Smartphone. Le but du responsive est d'éviter à l'utilisateur de redimensionner ou de recarder une page Web lorsque ce dernier navigue. L'introduction du responsive Web design est quelque chose de primordial aujourd'hui, car de plus en plus d'utilisateurs naviguent sur Smartphone et tablette au détriment d'un ordinateur fixe ou portable.

Il existe différents frameworks aidant à la réalisation d'un Frontend responsive.

- **Bootstrap :** est le framework CSS de Twitter qui est maintenant OpenSource. Il offre des éléments préconfigurés comme des formulaires HTML, boutons, listes, une police... Et bien sûr la possibilité de mettre en place un design responsive.
- **Foundation :** est un framework CSS créé par la société Zurb. Il permet de faire du responsive design plus détaillé que Bootstrap, mais son utilisation est aussi un brin plus complexe.

Ces frameworks permettent de gérer cette partie responsive lors du développement d'un projet de site Web ou d'application Web, leur utilité étant de repositionner des éléments:

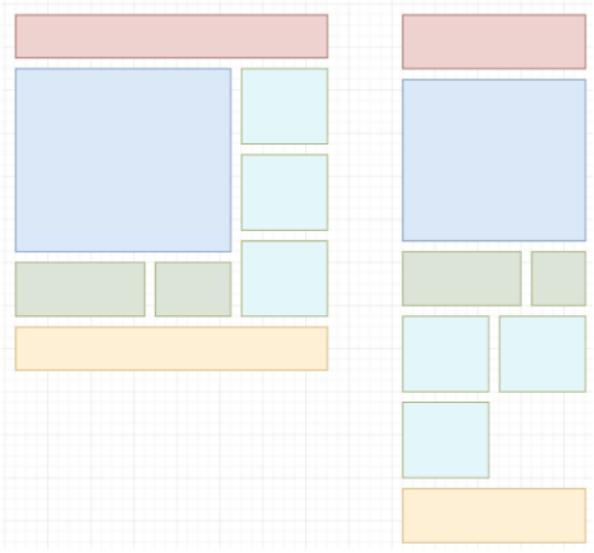


Figure 5 – Repositionnement des éléments avec les framework responsive

Le repositionnement de ces différents éléments sur la page dépend du type de support, voici les différents supports que ces outils de responsive Web design pourront gérer automatiquement:

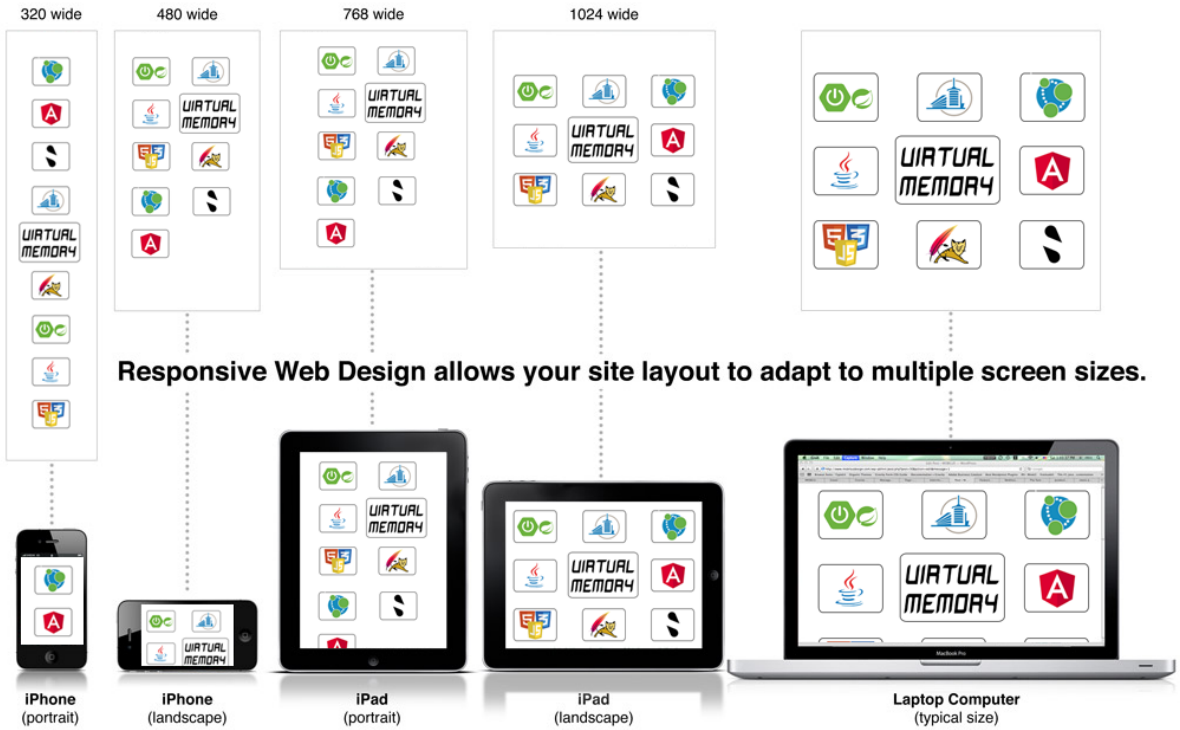


Figure 6 – Responsive Web design et notre projet [3]

Pour nous aider et automatiser le plus de tâches par rapport au responsive Web design, nous utiliserons le framework Bootstrap. Nous avons choisi cet outil, car il prend en charge les dernières versions stables des principaux navigateurs Web sur Desktop et Mobile:

	Chrome	Firefox	Safari	Android Browser & WebView	Microsoft Edge
Android	Supported	Supported	N/A	Android v5.0+ supported	N/A
iOS	Supported	Supported	Supported	N/A	N/A
Windows 10 Mobile	N/A	N/A	N/A	N/A	Supported

	Chrome	Firefox	Internet Explorer	Microsoft Edge	Opera	Safari
Mac	Supported	Supported	N/A	N/A	Supported	Supported
Windows	Supported	Supported	Supported, IE10+	Supported	Supported	Not supported

Figure 7 – Liste des navigateurs supportant Bootstrap [4]

Il est important de noter que les navigateurs proxy comme Opera Mini, le mode Turbo d'Opera Mobile, UC Browser Mini, Amazon Silk ne sont pas pris en charge par Bootstrap.

Comme nous allons offrir un design simple dans un premier temps, Bootstrap nous offre la possibilité de faire du responsive plus facilement. Bootstrap nous offre un CSS déjà très élégant. C'est aussi pourquoi nous choisissons ce framework. De plus, la dernière version de Bootstrap (la version 4) s'intègre à notre projet en une ligne de commande avec NPM qui est le gestionnaire de paquets officiel de Node.js.

2.3.4 API REST

"Une API REST ou parfois appelée RESTful est un style d'architecture permettant de construire des applications (Web, Intranet, Web Service). Il s'agit d'un ensemble de conventions et de bonnes pratiques à respecter et non d'une technologie à part entière. L'architecture REST utilise les spécifications originelles du protocole HTTP, plutôt que de réinventer une sur-couche (comme le font SOAP ou XML-RPC par exemple)." Selon un expert Web, Nicolas Hachet sur son blog. [5]

Selon lui, une API REST doit suivre 5 règles:

1. l'URI comme identifiant des ressources
2. les verbes HTTP comme identifiant des opérations (POST, GET, PUT, DELETE)
3. les réponses HTTP comme représentation des ressources (HTML, XML, CSV, JSON)
4. les liens comme relation entre ressources
5. un paramètre comme jeton d'authentification

Concrètement, selon l'exemple ci-dessous par de *Vinay Sahni* [6], nous devons utilisé le plus possible la même URL, mais d'ajouter un ID lorsque l'on veut un objet bien précis. Et utiliser les différentes méthodes du protocole HTTP sur la même URL.

```
GET /tickets - Retrieves a list of tickets
GET /tickets/12 - Retrieves a specific ticket
POST /tickets - Creates a new ticket
PUT /tickets/12 - Updates ticket #12
PATCH /tickets/12 - Partially updates ticket #12
DELETE /tickets/12 - Deletes ticket #12"
```

Une alternative à l'API Rest

GraphQL est une alternative à l'API Rest. GraphQL a été développé par Facebook et est libre d'usage depuis 2015. Ce langage d'interrogation permet d'obtenir les informations voulues sans forcément requêter tous les champs d'un objet ni même de requêter plusieurs tables d'une base de donnée SQL. GraphQL va probablement remplacer petit à petit les API Rest.

2.4 Les différentes bases de données

Cette analyse sur les différents types de bases de données nous aidera à choisir qu'elle est la meilleure façon de réaliser notre base de données dans le cadre de notre projet. Nous nous sommes intéressés aux types de bases de données suivants:

- Orienté texte
- Orienté documents
- Hiérarchiques
- Relationnels
- Orienté objets
- Orienté Graph

2.4.1 Orienté texte

Une base de données orientée texte est aussi appelée base de données dans un fichier plat, elle est composée d'un simple fichier, comme son nom l'indique, en format .txt ou .ini, qui représente une table en général, *"Un fichier plat est un fichier texte ou du texte combiné avec un fichier binaire contenant généralement un seul enregistrement par ligne."*[7] Les différents champs du fichier peuvent être séparés avec les formats CSV et DSV et à l'aide de séparateurs comme une virgule.

L'inconvénient d'une base de données orientée texte est sa gestion. En effet lorsque l'on veut ajouter ou mettre à jour des informations d'une ligne du fichier, il faut ré-entrer toutes les données dans un nouvel enregistrement, ce qui peut devenir vite complexe en fonction du nombre d'informations que nous pouvons avoir par ligne ou enregistrement.

Pour cette partie de l'analyse des bases des données, nous nous arrêterons là, parce que vu notre projet, nous pouvons dire d'avance que ce type de base de données ne nous conviendra pas vraiment, car nous aurons plusieurs entités à modifier constamment et ces modifications devront se faire le plus simplement possible pour l'administrateur du quizz.

2.4.2 Orienté documents

Une base de données orientée documents est un type de bases de données destiné aux applications ayant une fonction de gestion des documents. Les cas d'utilisation courant d'une base de données orientée document :

- *"Persistance de profils utilisateurs, de regroupement de contenus destinés à une même page, de données de sessions*
- *Cache d'informations sous une forme structurée*
- *Stockage de volumes très importants de données pour lesquelles la modélisation relationnelle aurait entraîné une limitation des possibilités de partitionnement et de réplication"*[8]

Mongo DB

Dans le monde des bases de données orientées documents, Mongo DB est ce qui se fait de mieux.

Mongo DB est la première base de données NoSQL, c'est une base de données orientée multi-plateforme et OpenSource, elle est basée sur un modèle de données de type document de hautes performances avec une haute disponibilité et une évolutivité facile. Mongo DB travaille sur le concept de la collecte et du document qui lui confère une grande souplesse d'utilisation.

Elle permet de faire évoluer le schéma de la base de données très simplement et à tout moment. Sa structure est très simple et explicite, car les relations dans la base de données sont réduites au maximum. Au niveau de ces fonctionnalités, Mongo DB est très intéressant, car un système de redondance est mis en place, les serveurs de base de données sont dupliqués et répartis géographiquement pour gérer une montée en charge du trafic, pour mieux gérer les pannes, augmenter les performances et répartir les données. Les requêtes dynamiques sur des documents sont supportées par Mongo DB et ce processus est autant puissant que les requêtes SQL. Les documents peuvent être différents de l'un à l'autre, leur taille, leur nombre de champs contenus peuvent différer.

Mongo DB est adéquate pour la gestion des Big Data (la gestion des données d'utilisateurs, les données mobiles et sociales...). De plus, le soutien de Mongo DB est apparemment très professionnel.

RethinkDB

À la demande de notre mandant, nous devons réaliser une étude approfondie de RethinkDB, alors qu'est-ce c'est?

*"RethinkDB pushes JSON to your apps **in realtime**.*
When your app polls for data, it becomes slow, unscalable, and cumbersome to maintain. RethinkDB is the open-source, scalable database that makes building realtime apps dramatically easier."[9]

RethinkDB est une base de données orientée documents et distribuée. Ce SGBD est utilisé par des clients tels que la NASA, Wise.io, etc. pour des raisons aussi diverses que la simplicité des requêtes à la simplification des services en temps réel.

RethinkDB possède des drivers de connexions en JavaScript, Python, Ruby et Java. RethinkDB offre un backend OpenSource nommé *"horizon"* [10].

RethinkDB est compatible avec les systèmes d'exploitation suivants, voire figure 8:

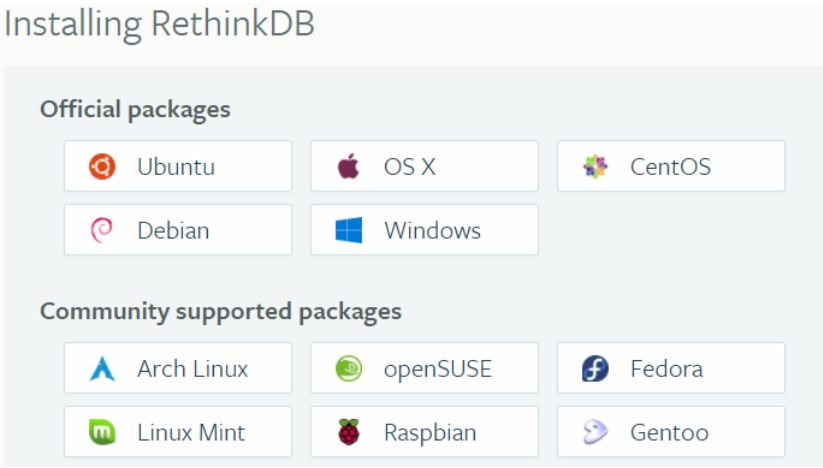


Figure 8 – Systèmes d'exploitation compatibles avec RethinkDB[11]

Le temps réel

"Avec les bases de données temps réel, il est possible de "s'abonner" à une requête et d'être tenu informé à chaque modification, sans avoir à faire une requête. Cela permet donc la création d'application comme le travail collaboratif de Google Doc, Trello, ... bien plus simplement."[12]

RethinkDB nous permet donc de nous abonner à une requête. C'est-à-dire que nous n'avons plus besoin de demander la ressource à chaque fois, mais juste s'abonner à cette requête. Demander une ressource qui n'a pas changé génère du trafic inutile. En nous abonnant à une requête, nous recevrons les nouvelles données à chaque fois qu'elles changent. Les figures 9 et 10 nous montrent la différence entre un échange requête/réponse traditionnel et un abonnement à une donnée.

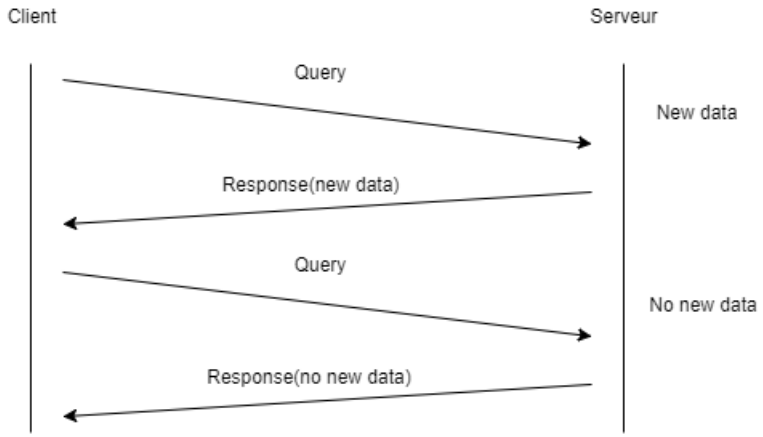


Figure 9 – Échange requête/réponse traditionnel

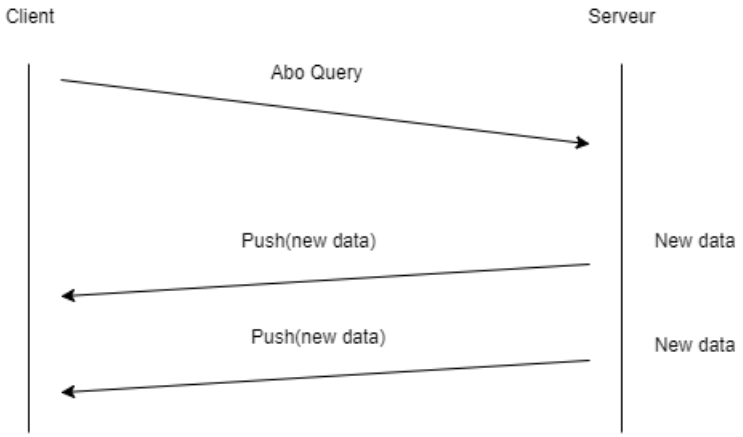


Figure 10 – Échange avec abonnement

La requête suivante provenant du site RethinkDB [9] nous montre comment nous abonner à une requête dans le cas de l'affichage des 3 meilleurs scores d'un jeu. La méthode change() est un listener qui contrôle si une valeur change et met-à jour la requête.

```
r.table('game').orderBy('score').limit(3).changes()
```

ReQL

RethinkDB a son propre langage de requêtes nommé ReQL. Nous allons comparer des requêtes entre le langage SQL que nous connaissons déjà avec des requêtes ReQL. Les requêtes suivantes proviennent du guide *"Why learn RethinkDB?"* [13].

Voici une requête simple en SQL:

```
SELECT * FROM users WHERE name="Vinh" ORDER BY id DESC LIMIT 10,100
```

Voici la même requête en ReQL:

```
r.table('users').getAll('vinh', {index: 'name'}).order_by(r.desc(id)).limit(10)
```

Nous pouvons constater que le principe des requêtes est plus ou moins le même. En effet, les méthodes utilisées ont le même nom que les mots clés SQL.

Voici une requête en SQL plus complexe:

```
SELECT *
FROM foods as f
INNER JOIN compounds_foods as c ON c.food_id=f.id
WHERE f.id IN (10, 20)
ORDER By f.id DESC, c.id ASC
```

Voici la même requête en ReQL:

```
r.db("food")
    .table("foodbase")
    .filter(function (food) {
      return r.expr([10, 20{reql}{]).contains(food("id"))
    })
    .eqJoin("id", r.db("foodbase").table("compound_foods"), {index: "food_id"})
```

Le langage ReQL permet aussi les jointures.

Exemple d'utilisation

Les exemples de code suivant proviennent du guide *"Ten-minute guide with RethinkDB and Java"*[14].

```
// instance a RethinkDB object named r
public static final RethinkDB r = RethinkDB.r;
// open a connection, the default port is 28015
Connection conn = r.connection().hostname("localhost").port(28015).connect();
// create a new db named "test" and create a table named "authors"
r.db("test").tableCreate("authors").run(conn);
// insert a new document in the table named "authors"
r.table("authors").insert(r.array(
    r.hashMap("name", "William Adama")
     .with("tv_show", "Battlestar Galactica")
     .with("posts", r.array(
        r.hashMap("title", "Decommissioning speech")
         .with("content", "The Cylon War is long over..."),
        r.hashMap("title", "We are at war")
         .with("content", "Moments ago, this ship received..."),
        r.hashMap("title", "The new Earth")
         .with("content", "The discoveries of the past few days...")
        )
    )
)).run(conn);
// retrieve all documents from the table authors
Cursor cursor = r.table("authors").run(conn);
for (Object doc : cursor) {
    System.out.println(doc);
}
// try to retrieve the document where the name attribute is set to William Adama
Cursor cursor = r.table("authors").filter(row -> row.g("name").eq("William
     Adama")).run(conn);
for (Object doc : cursor) {
    System.out.println(doc);
}
// retrieve all authors who have more than two posts
Cursor cursor = r.table("authors").filter(row ->
     row.g("posts").count().gt(2)).run(conn);
for (Object doc : cursor) {
    System.out.println(doc);
}
// delete every document with less than three posts
r.table("authors").filter(row ->
     row.g("posts").count().lt(3)).delete().run(conn);
```

L'utilisation de ReQL nous semble par très complexe du moment ou les requêtes restent simples.

2.4.3 Hiérarchique

"Une base de données hiérarchique est une base de données dont le système de gestion lie les enregistrements dans une structure arborescente où chaque enregistrement n'a qu'un seul possesseur."[15]

Une base de données de type hiérarchique est considérée comme un ensemble de plusieurs arbres, chaque arbre est indépendant et possède une racine unique, ce type de base de données permet de représenter le monde réel, car le monde réel est souvent organisé hiérarchiquement.

Ce type de base de données peut être facilement étendu, très évolutif. La facilité de gestion, de bonnes performances et de la redondance sont garanties avec les bases de données de type hiérarchique.

2.4.4 Relationnel

"Une base de données relationnelle est une collection de données organisées sous la forme de tables définies de façon formelle, à partir desquelles les données sont accessibles et assemblées sans avoir à réorganiser les tables de la base de données."[16]

Ce type de base de données est ce qu'il y a de meilleur pour l'organisation logique des données. Les données sont bien rangées, claires et peuvent être utilisées avec aisance. La technologie utilisée pour les bases de données relationnelles est stable et sûre comme par exemple SQL qui est un langage standard et normalisé. Des requêtes complexes peuvent être mises en place avec ce système.

Avec ce modèle relationnel SQL, c'est relativement simple, les tables contenues dans la base de données ont des enregistrements et ces tables peuvent avoir des relations, ce qui permet de lier très facilement des enregistrements d'une table à l'autre.

Un des principaux inconvénients de ce type de base de données est la modification de l'architecture de base de la base de données qui peut être complexe. Au niveau des performances ce type de base de données est très puissant, mais lors d'un très grand volume de données, les performances peuvent vite baisser.

2.4.5 Orienté objets

Les SGBDOO servent à stocker des objets. Ils sont souvent spécialisés pour certains langages ce qui fait que ce type de base de données est très peu utilisé. Nous allons voir un exemple qui est toujours utilisé.

"db4o (DataBase For Objects) est un système de gestion de base de données orientée objets Open Source pour des applications Java et .Net."[17] Son langage est db4o-sql.

Un exemple avec db4o en Java provenant d'un forum [18].

```
public class Ch2 {
    public static void main(String[] args) {
        ObjectContainer bdd = Db4oEmbedded.openFile(Db4oEmbedded.newConfiguration(),
             "pilote.db4o");
        try {
            // Creation d un pilote.
            Pilote pilote1 = new Pilote("Michael Schumacher", 100);
            // Ajout dans la bdd.
            bdd.store(pilote1);
            System.out.println("Le pilote " + pilote1 + " a ete enregistre.");
        }
        finally {
            bdd.close();
        }
    }
    public static void listerResultats(List<?> resultats) {
        System.out.println("Number of results : " + resultats.size());
        for(Object o : resultats) {
            System.out.println(o);
        }
    }
}
```

DB4o stock très simplement les objets tels quels dans la base de données.

Les bases de données objets ont été développées au début 2000, mais leur petit succès fait qu'elles sont rarement utilisées aujourd'hui.

2.4.6 Orienté Graph

Dans nos recherches sur les bases de données, nous avons découvert les bases de données orientées Graph. Voici une introduction sur ce type de bases de données, expliqué par Guillaume Serries : *"Les bases de données Graph, qui utilisent des structures graphiques pour répondre à des requêtes, sont devenues une évidence pour beaucoup depuis que les réseaux sociaux comme Facebook et Twitter en ont fait le cœur de leur gestion de base de données. [...] Avec des données de qualité, les bases de données Graph permettent de modéliser les données et de les stocker de la manière dont nous pensons et raisonnons dans le monde réel."* [19]

Neo4j Neo4j est la base de données orientée Graph la plus employée et se positionne à la 21ème place du classement des bases de données les plus utilisées. Son langage de requêtes se nomme "Cypher". Voici comment créer une base de données, un exemple de ce que nous pouvons réaliser avec cette base de données orientée Graph `http://console.neo4j.org`

```
create (Neo:Crew {name:'Neo'}), (Morpheus:Crew {name: 'Morpheus'}), (Trinity:Crew
    {name: 'Trinity'}), (Cypher:Crew:Matrix {name: 'Cypher'}), (Smith:Matrix {name:
    'Agent Smith'}), (Architect:Matrix {name:'The Architect'}),
(Neo)-[:KNOWS]->(Morpheus), (Neo)-[:LOVES]->(Trinity),
    (Morpheus)-[:KNOWS]->(Trinity),
(Morpheus)-[:KNOWS]->(Cypher), (Cypher)-[:KNOWS]->(Smith),
    (Smith)-[:CODED_BY]->(Architect)
```

Le résultat des quelques lignes Cypher ci-dessus est visible dans la figure 11.

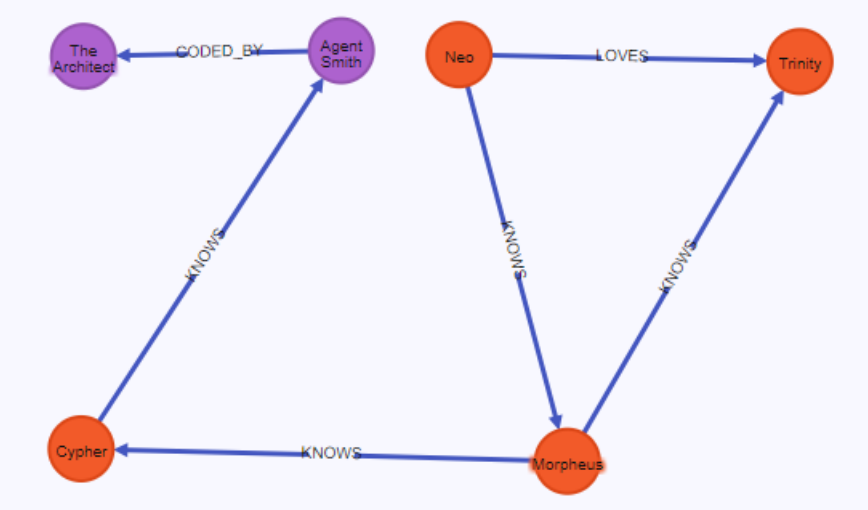
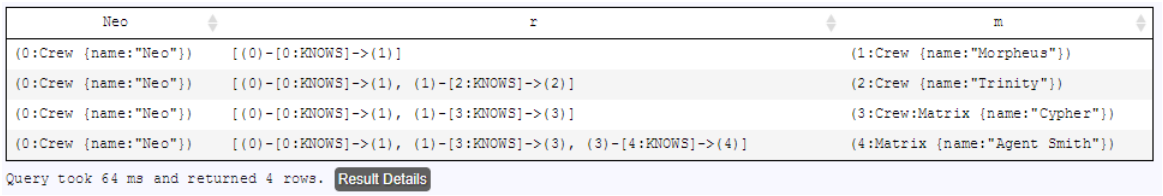


Figure 11 – Graph du contenu "exemple" de la Neo4j Console en ligne

```
match (n:Crew)-[r:KNOWS*]-(m) where n.name='Neo' return n as Neo,r,m
```

La requête suivante donne le résultat à voir dans la figure 12.



Neo	r	m
(0:Crew {name:"Neo"})	[(0)-[0:KNOWS]->(1)]	(1:Crew {name:"Morpheus"})
(0:Crew {name:"Neo"})	[(0)-[0:KNOWS]->(1), (1)-[2:KNOWS]->(2)]	(2:Crew {name:"Trinity"})
(0:Crew {name:"Neo"})	[(0)-[0:KNOWS]->(1), (1)-[3:KNOWS]->(3)]	(3:Crew:Matrix {name:"Cypher"})
(0:Crew {name:"Neo"})	[(0)-[0:KNOWS]->(1), (1)-[3:KNOWS]->(3), (3)-[4:KNOWS]->(4)]	(4:Matrix {name:"Agent Smith"})

Query took 64 ms and returned 4 rows. Result Details

Figure 12 – Résultat de la requête ci-dessus dans la Neo4j Console en ligne

Les bases de données orientées Graph comme Neo4j permettent de créer n'importe quel objet et de les relier entre eux en créant n'importe quelle relation.

2.5 Qu'est ce que nous allons stocker ?

D'après nos analyses, nous pouvons maintenant lister les technologies dont nous allons nous servir pour réaliser notre projet. Angular CLI et Springboot étant déjà imposés, la seule technologie dont nous avions à choisir était celle à mettre en place pour la base de données. Bien entendu, nous devons savoir précisément, ce que nous allons stocker.

Cette partie entamera déjà un peu la conception de notre projet, nous avons du penser techniquement comment stocker les données de notre projet.

Notre projet sera basé sur des questions et des réponses, ces questions et ces réponses seront vues comme des entités, une entité comportera des informations comme au minimum son titre (texte, image...) et son sujet qui déterminera le domaine dans lequel elle jouera un rôle (art, sport, politique...), ces entités seront reliées entre elles avec des liens.

2.6 Choix des technologies

Le choix du type de base de données sera d'abord de trancher entre des bases de données SQL ou NoSQL, nous avons trouvé un article pour le moins intéressant sur ce sujet, traitant des contraintes de SQL cet article démontre comment NoSQL se démarque et résout ces contraintes. Bien sûr, SQL est extrêmement pratique dans le monde de l'informatique et des données, mais certaines contraintes font qu'il est parfois compliqué pour l'outil de trouver une information dont il a besoin, surtout quand des centaines de personnes veulent accéder à cette même information, ce qui est très coûteux au niveau des ressources.

Afin de palier à ces contraintes, NoSQL a été pensé différemment, comme le cite "GEEKO" dans son article : *"Plutôt que de s'appuyer sur des contraintes lourdes et consommatrices de ressources, on va se concentrer sur l'accès direct à l'information, car c'est finalement ce que recherche l'utilisateur. [...] NoSQL est donc un modèle de BDD adapté au web, où l'utilisateur et l'accès à l'information sont mis en avant."*[20] Nous pouvons voir que NoSQL préfère mettre en avant l'accès à l'information avant toutes autres choses. Le résultat de cette manière de faire pour NoSQL est une scalabilité immédiate ! En effet une montée en charge du trafic ne va pas avoir de conséquences sur les performances qui ont la capacité à rester constantes.

Bien entendu, cela ne veut pas dire que tout le monde doit se mettre au NoSQL. Il est clair que ces bases de données ont des avantages réels fassent au SQL, mais il faut prendre en compte plusieurs critères et cela dépend des cas. Comme le cite "GEEKO" : *"MySQL est loin d'être mort, il continue d'évoluer et justement dans le sens de la scalabilité/rapidité d'accès en lecture. De plus il suffit très bien dans certains cas où l'utilisateur ne recherche pas la vitesse à tout prix."*[20] Il faut aussi prendre en compte que NoSQL demande un niveau de conception en plus, la structure du stockage de l'information doit être pensée minutieusement avant de mettre en place la structure.

Le mot de la fin est qu'avant de choisir entre une des deux technologies, il faut savoir ce que l'on veut faire précisément, comme "GEEKO" le cite très bien pour conclure dans son article : *"Pour résumer, il faut savoir ce qu'on veut faire : est-ce que l'accès à une information doit être rapide ? Si oui, il vaut mieux privilégier NoSQL. Est-ce que notre client a besoin de stocker de grandes quantités de données pour les réutiliser de temps à autre ? À ce moment-là, MySQL semble plus adapté."*[20]

Actuellement, il n'est pas vraiment facile de trancher entre quel projet irait mieux avec SQL ou quel projet irait mieux avec NoSQL. SQL semble mieux adapté pour les projets ou la structure de données peut être identifié à l'avance ou quand l'intégrité des données est essentielle. NoSQL semble mieux adapté pour les projets où l'organisation de données est indépendante, indéterminée et évolutive ou quand l'organisation de la base de données demande un forte agilité et gérant beaucoup de données structurées, semi-structurées, non structurées et polymorphes.

Dans un premier temps, nous pensions nous pencher sur du SQL, donc sur un type de bases de données relationnelles, pour plusieurs raisons :

- Le SQL est un langage standardisé, normalisé et mature que nous connaissons bien
- Les données que nous allons stocker est bien assimilé à des tables et des champs, nos entités seraient nos tables et nos informations sur les entités seraient les champs. Ce qui nous permettrait de stocker les données de manière claire et structurée, de plus nos entités sont liées entre elles ce qui se fait relativement facilement avec une table de lien
- Une grande quantité d'informations sera stockée dans notre base de données, ce qui doit être le plus structuré possible et accessible
- Notre structure de données sera pensée et identifiée à l'avance ce qui nous évitera des modifications dans la structure de la base de données

Pour résumer la situation, nous pensions partir dans la simplicité avec une technologie que nous connaissons bien, mais après une grande réflexion, nous nous sommes dit que ce n'était peut-être pas la meilleure situation. En effet, la clé de notre projet n'est pas nos entités, mais plutôt ce qu'il y a entre nos entités, c'est à dire les liens entre elles. Afin de gérer au mieux ces liens, le type de base de données le plus optimal est la base de données Graph.

Ce qui a retenu notre attention pour ce type de base de données est la phrase suivante citée dans le point 2.3.6 : *"Une base de données Graph permet de modéliser les données et de les stocker de la manière dont nous pensons et raisonnons dans le monde réel"*. Cette citation, nous a fait nous pencher encore plus sur ce type de base de données.

Une base de données orientée Graph, ne travaille pas comme les autres types de bases de données NoSQL. Les autres types de bases de données NoSQL travaillent et se focalisent sur le traitement de la donnée afin d'en tirer une valeur, or les bases de données orientées Graph travaillent et se focalisent plus sur les jointures, les connexions, les liens, ce qu'il y a entre les données, ce qui se rapporte à la clé de notre projet. Concernant les bases de données relationnelles ou SQL, il est vrai que cette qualité de travail sur les jointures qu'a une base de données orientée Graph, n'est pas reproduite. Il est vrai qu'avec SQL, les jointures entre les données seront vite difficiles à gérer, ce qui à un impacte sur les performances qui seront plus faibles et donc de la latence. Le fait d'augmenter le nombre de relations avec SQL, rend la gestion de la base de données très vite complexe, ce qui n'est pas le cas avec une base orientée Graph, la gestion et l'évolutivité se font beaucoup plus simplement.

Plusieurs grands noms utilisent des bases de données orientées Graph comme Google, Facebook, Twitter, eBay, Meetic. Lors d'une recherche Google, des pages provenant des 4 coins du monde sont affichées instantanément, tout se joue sur des liens, c'est la clé. Si nous prenons l'exemple de eBay, comme le cite Guillaume Serries dans un article : *"C'est avec cette technologie également que eBay peut optimiser ses livraisons en Californie, promise en moins de deux heures. "7 lignes de code Cypher ont suffi pour mettre en place un système où le livreur peut récupérer un nouveau colis et le livrer alors même qu'il effectue déjà une livraison" mentionne le responsable de Neo4j. "Cela permet de faire deux à trois tournées au lieu d'une seule dans le même temps."* [19] Ces algorithmes seraient extrêmement difficiles à mettre en place avec une base de données relationnelle. Pour Meetic, Twitter ou Facebook tout se joue aussi sur la relation, les liens entre les personnes, les entités.

Concrètement, avec une base de données orientée Graph, nous allons gagner en:

- **Performance**: Lorsque les données, les noeuds et les liens grandissent, les performances restent les mêmes avec ce type de base de données, comme le cite si bien Guillaume Serries : *"Alors que les performances des requêtes se dégradent rapidement dans les bases de données relationnelles avec la croissance des jointures, la performance des bases de données Graph restent identiques quand le jeu de données s'accroît."* [19]
- **Flexibilité**: Faire évoluer la base de données est simple et sans inconvénients: *"Les nouvelles relations où les nœuds peuvent être ajoutés sans interférence avec les requêtes et les applications existantes."* [19]
- **Agilité**: Guillaume Serries nous parle aussi de ce point dans son article: *"Le modèle de données semi-structuré ou déstructuré des bases de données Graph permet de faire évoluer la base en cohérence avec les nouvelles activités d'une entreprise par exemple."* [19]
- **Scalabilité**: Ce point est très important pour nous, les performances doivent rester constantes.

Bien entendu, les bases de données orientées Graph peuvent aussi avoir des inconvénients. Ces bases de données ne sont pas recommandées pour stocker des Big Data. En effet, une des limites se trouve au niveau du partitionnement de la base de données en fonction de sa densité. Dans un environnement distribué, les très grosses bases de données orientées Graph peuvent être réparties sur plusieurs serveurs, ce qui peut diminuer les performances. De plus, lorsqu'il faut calculer de très grandes agrégations de données, les performances peuvent aussi diminuer.

Nous avons décidé de faire un tableau récapitulatif des besoins de notre projet avec différents types de bases de données. Nous aurons une vue plus explicite afin de trancher sur quelles technologies nous pourrons partir, le choix se fera en fonction des coûts que nous attribuerons:

Fonctions	NoSQL					SQL
	Graph	Texte	Documents	Objets	Hiérarchique	
Complexité de requêtage	Au niveau de la couche application					Au niveau du schéma
	?	?	?	?	?	?
Montée en charge (scalabilité)	15	15	15	15	15	5
Temps de réponse pour gros volumes de données	20	20	20	20	20	0
Schéma dynamique/évolutif	15	0	15	15	15	5
Module de communication	API (plus compliqué à mettre en place)					15
	10	10	10	10	10	(Driver SQL)
Langage d'interrogation	API REST (HTTP) (plus compliqué à mettre en place)					15
	10	10	10	10	10	(SQL)
Insertion de nouvelles données	15	0 (Tout réinsérer)	15	15	15	15
Relation/jointure	20 (Seul Graph dans le NoSQL gère les relations/pas de tables de relation)	0	0	0	0	15 (Table de relation)
Utilisation pour notre projet	20 (Réseaux sociaux, relations entre entités)	-	5 (Profil utilisateur d'une application Web)	-	-	15 (Organisation logique des données)

Dans notre tableau comparatif, nous voyons que les bases de données SQL sont bonnes pour certaines fonctions, mais au niveau de la scalabilité, le temps de réponse et l'évolutivité ce n'est pas optimal. Pourtant ces fonctions doivent être performantes dans notre projet. Si nous nous penchons du côté NoSQL, le type de base de données qui répond le plus à ces demandes est le base de données orientée Graph, de plus, le base de données orientée Graph est ce qui se fait de mieux au niveau des relations/jointures, ce qui est la clé de notre projet.

Pour conclure cette partie sur le choix de la base de données, dans un premier temps, nous étions focalisés sur un type de base de données SQL, car nous connaissons bien la technologie et que cette technologie est aussi ce qu'il y a de plus utilisé:

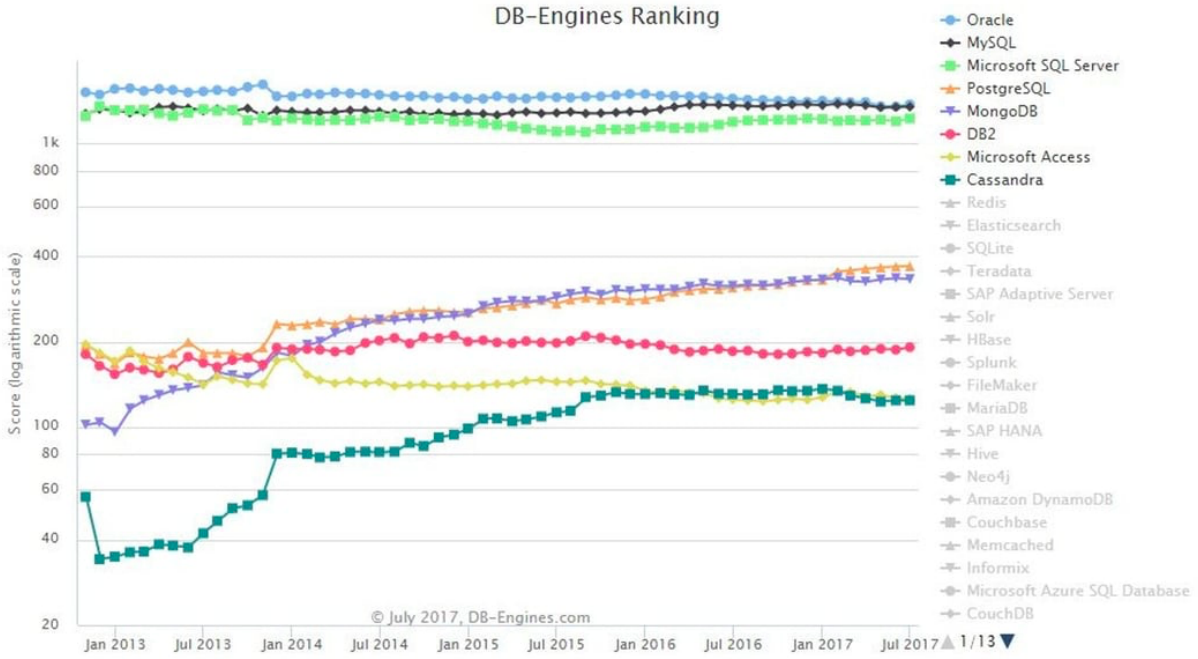


Figure 13 – Utilisation des différents types de bases de données [21]

Mais, après plus de recherches, nous avons vu que pour notre projet, ce n'était pas ce qu'il y avait de plus adéquat. Nous avons donc décidé de nous tourner vers une base de données orientée Graph afin de stocker nos données et comme analysé précédemment, nous utiliserons plus particulièrement Neo4j et son langage Cypher comme base de données orientée graphe. Nous ne connaissions pas du tout ce type de bases de données, il est aussi intéressant de se lancer dans une technologie que nous ne connaissons pas vraiment, c'est un joli challenge à relever, voilà aussi un point qui nous fait partir sur ce type de bases de données et non le SQL, le challenge.

Les bases de données Graph sont composées de noeuds et de relations entre ces noeuds, un noeud possède un nom et n propriétés, une relation est toujours unidirectionnelle et possède un nom et n propriétés. Dans ce type de bases de données, les noeuds sont aussi importants que les relations.

Ci-dessous un petit aperçu d'un test que nous avons effectué avec quelques requêtes Cypher sur le site Web Neo4j `http://console.neo4j.org` :

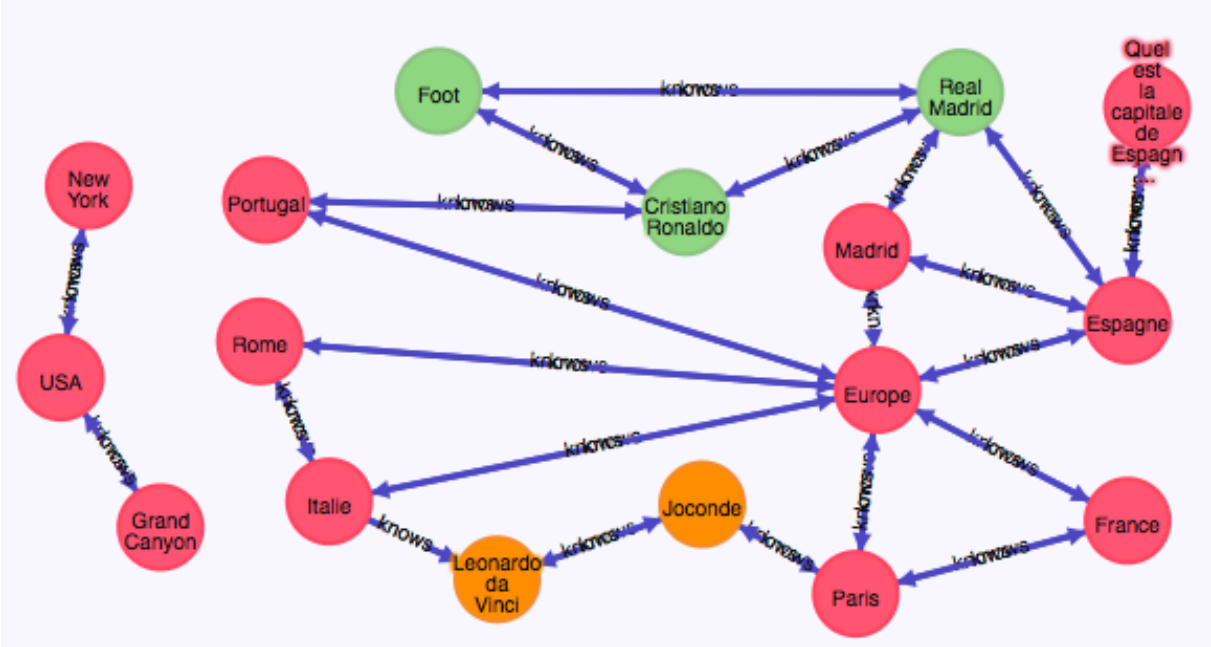


Figure 14 – Requêtes Cypher dans la console du site Web Neo4j se rapportant à l'idée de notre projet

Sur ce schéma, nous pouvons voir nos différentes entités (noeuds) se rapportant à l'idée de notre projet, réparties dans divers sujets (propriétés) comme la géographie, le sport et l'art. Nous pouvons voir que tout se joue au niveau des liens entre ces entités. Des liens entre différents sujets peuvent être créés ainsi que des liens entre ces différentes entités, ce qui est très intéressant.

L'utilisation de Neo4j est soumise à différentes licences.

La "Community Edition" est Open Source sous licence GPLv3. Cette édition est complète et performante, et peut être utilisé gratuitement dans un projet si la base de données est hébergée dans une organisation ou chez une personne privée.

La "Enterprise Edition" à une licence Enterprise et est donc payante. Et offre divers service en plus que la "Community Edition", comme l'hébergement et du support pour une base de données.

2.7 Outils de développement

Ayant lu, une bonne partit de la documentation de Spring, seulement deux IDE sont recommandés par ces derniers, STS et IntelliJ IDEA.

2.7.1 Spring Tool Suite

"The Spring Tool Suite is an Eclipse-based development environment that is customized for developing Spring applications. It provides a ready-to-use environment to implement, debug, run, and deploy your Spring applications, including integrations for Pivotal tc Server, Pivotal Cloud Foundry, Git, Maven, AspectJ, and more." [22]

Cet IDE est un logiciel gratuit fait pour coder avec Spring basé sur le l'IDE Eclipse que nous avons appris à connaître dans notre formation.

2.7.2 IntelliJ IDEA

"IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains" [23], voilà comment cet IDE est présenté par Jetbrains sur leur site web.

IntelliJ IDEA est un IDE permettant de coder pour la JVM offrant de nombreuses facilitations pour le programmeur. Cet IDE produit par JetBrains est payant (environ 150 CHF) la première année, mais est offert aux étudiants.

De plus, cet IDEA comprend les langages Web comme qu'Angular CLI, HTML, etc... *"IntelliJ IDEA includes all features of WebStorm"*. [24], selon un développeur de chez JetBrains. WebStorm est un IDE permettant de coder avec divers langages Web aussi créé par JetBrains.

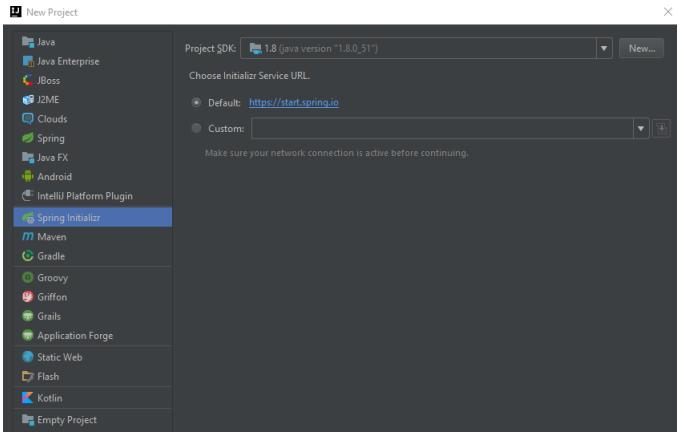


Figure 15 – Création d'un nouveau projet avec Spring Initializer sur ItelliJ IDEA 2017 2.5

IntelliJ IDEA nous permet à l'initialisation d'un projet SpringBoot, de choisir des dépendances directement pour divers types de bases de données SQL tel que MySQL, PostgreSQL, etc.. et NOSQL comme MongoDB, Neo4j.

2.7.3 Notre choix

Notre choix se porte sur IntelliJ IDEA qui est beaucoup plus complet que STS. En effet, il va nous permettre de développer le Backend et le Frontend de notre application. Et nous permet de commencer à développer sans aucun plug-in additionnel.

2.8 Conclusion de l'analyse

Nous pouvons donc vous présenter le schéma complet de notre projet avec les technologies imposées et choisies par nos propres soins:

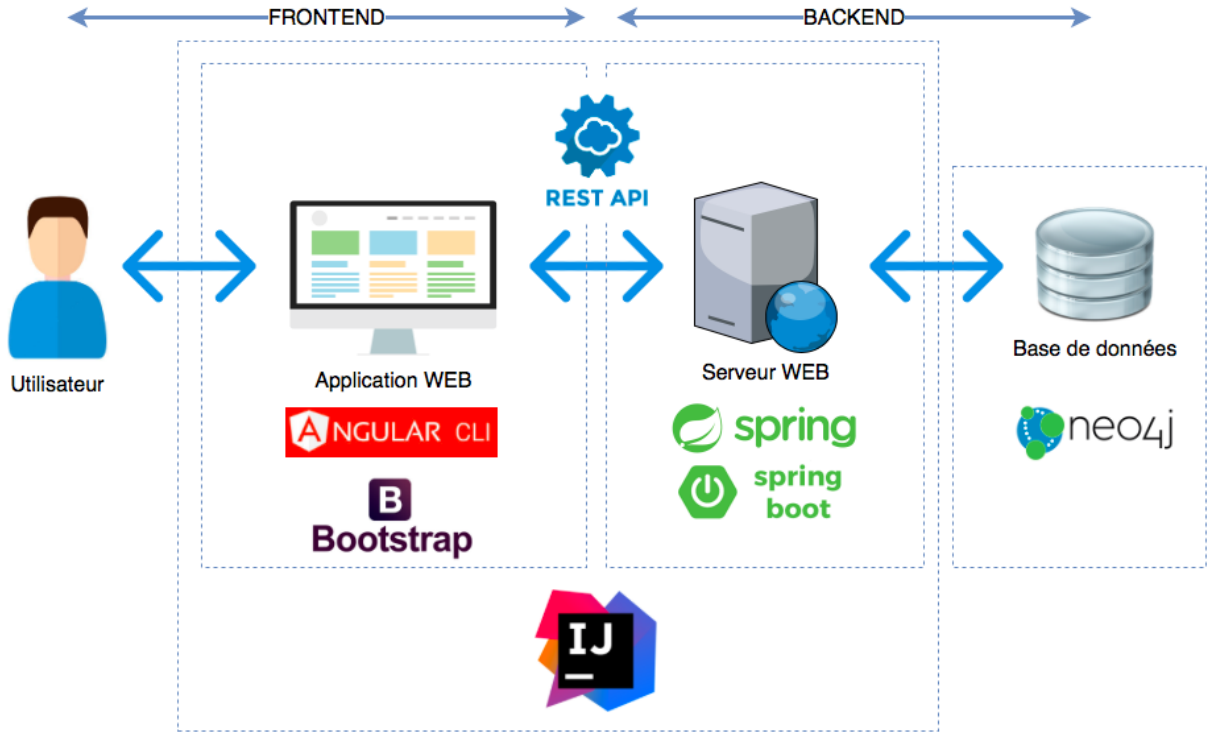


Figure 16 – Schéma des différentes technologies de notre projet

Nos technologies et nos outils de travail étant définis et choisis, comme le prouve le schéma ci-dessous, nous pouvons clore cette partie d'analyse du projet et passer à la partie conception de notre projet.

3 Conception

Ce chapitre traite de la conception de notre projet. Cette conception permettra au mandant ou aux personnes qui reprendront le projet de bien comprendre son fonctionnement et la manière dont il a été pensé.

La conception présentera les différentes maquettes de notre projet "Virtual Memory", que ce soit la partie jeu ou la partie administration. Ces deux parties seront aussi conceptualisées grâce à différents Use Case et diagrammes de séquences afin de bien comprendre comment implémenter le projet. Nous penserons aussi la base de données de notre projet avec différents schémas et maquettes. Dans ce chapitre, nous avons aussi créé un logo représentant le Virtual Memory.

3.1 Logo

Nous avons décidé de créer un logo qui représente notre projet "Virtual Memory", car nous pensons que c'est une chose importante que d'avoir une identité visuelle pour notre application.

Nous voulons un logo simple, mais tape-à-l'oeil, un logo qui rappelle le fait qu'il s'agit d'un memory ou d'un quizz. Après plusieurs recherches et de la réflexion, nous avons pensé que mettre en avant une représentation de cartes dans le logo rappellera le sens du projet comme cité précédemment. Nous voulons aussi jouer avec des couleurs vives ce qui permet d'éveiller les sens du joueur.

Avec tous ces éléments, nous avons pu mettre en avant un croquis préliminaire :

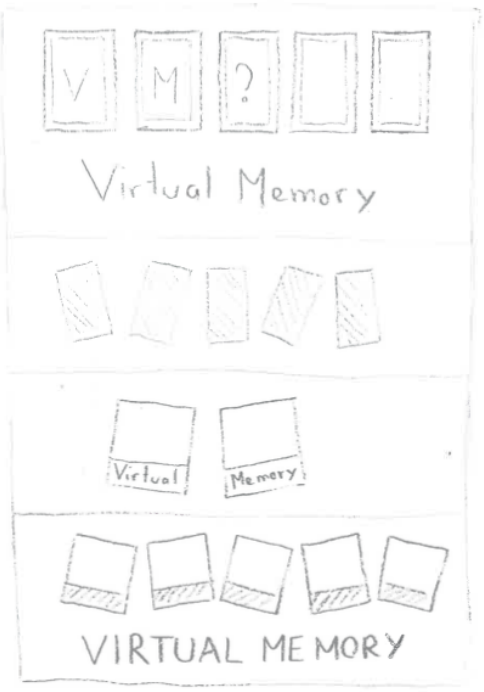


Figure 17 – Croquis du logo

Le dernier modèle met bien en avant la notion de cartes rappelant un quizz ou un memory, en décalant les cartes avec une certaine rotation, cela donne un côté déstructuré à notre croquis. Il ne manque plus que les couleurs qui tapent à l'oeil et nous allons aussi travailler la police de l'écriture afin d'avoir une écriture simple, nette et précise, mais tout de même originale.

Au final, nous sommes arrivés à ce logo :

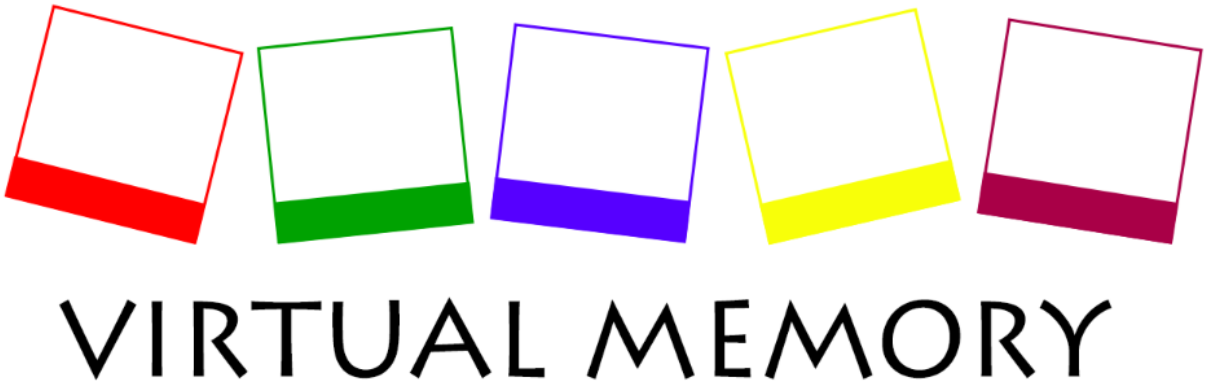


Figure 18 – Logo final représentant notre projet "Virtual Memory"

Ce logo nous semble original et rappelant le but de notre projet. Bien entendu, nous avons voulu amener un petit côté ludique à l'application, mais ce logo peut être changé ou modifié selon les goûts du mandant.

3.2 Fonctionnement de l'application

Nous allons définir globalement le fonctionnement de l'application comme expliqué lors des séances avant de passer à une phase plus détaillée avec les Use Case.

Notre application est basée sur des entités. Un quizz est composé de x entités, une entité principale (question) et x entités secondaires, dont x réponses correctes et x réponses fausses. Une entité peut être composée d'un texte ou d'une image et elle ne définit pas une question ou une réponse, une question peut être réponse et une réponse peut être question.

Comme défini dans le cahier des charges, notre application est composée de deux parties :

- **La partie jeu:** Le joueur choisi le ou les quizz avec lesquels il veut jouer, il lance la partie et répond aux différents quizz qu'il a choisis.
- **La partie administration:** L'administrateur a le choix entre deux modes d'administration:
  - Création des entités et les liaisons avec les autres entités.
  - Création des quizz en sélectionnant une entité principale.

3.3 Interface graphique

Pour ce thème qui traire des interfaces graphiques, nous traiterons de la partie responsive de notre application Web pour commencer. Ensuite, nous pourrons créer nos différentes maquettes.

3.3.1 Bootstrap

Nous avons choisi la framework Bootstrap, dans l'analyse de notre projet, pour traiter du côté responsive de notre application Web.

Bootstrap travaille avec une grille de 12 colonnes par défaut, ce qui est intéressant, c'est que cette grille est entièrement paramétrable, extensible avec d'autres colonnes dans la grille ou dans les colonnes elles-mêmes, ou le nombre de colonnes peut aussi être réduit :

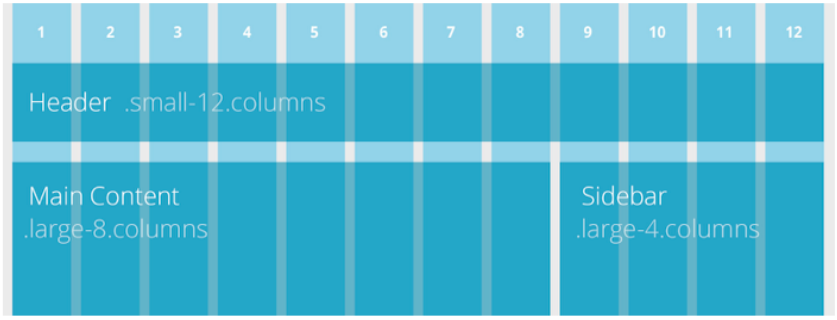


Figure 19 – Représentation de la notion de grille et colonnes dans Bootstrap [25]

Bootstrap propose aussi une bonne documentation, des templates de composants classiques (tableaux, formulaires, boutons) ou plus inédits (barres de navigation, menus, carrousels, galeries d'images) du Web déjà configuré et prêt à être intégré et employé à un projet.

Les différentes classes mises à disposition par le framework Bootstrap permettent de gérer la taille des pages en fonction des pixels des écrans. Nous avons juste à utiliser les différentes classes présentées ci-dessous, nous n'avons pas à nous soucier du type d'appareil (Smartphones, tablettes ou ordinateur...) ni de calculer leurs dimensions, les classes automatisent ce processus, et nous pouvons "jouer" simplement avec les classes proposées et les colonnes sur la grille. Ci-dessous, les différentes classes, parmi tant d'autres, à disposition pour "jouer" avec la grille et ces colonnes en fonction des pixels de l'écran :

	Extra small <576px	**Small** ≥576px	**Medium** ≥768px	**Large** ≥992px	**Extra large** ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`
# of columns	12				

Figure 20 – Représentation de la notion de grille et colonnes dans Bootstrap [26]

Comme nous montre le site Web officiel de Bootstrap `https://getbootstrap.com`, il est simple de gérer cette grille et ces colonnes avec les différentes classes :

```
<!-- Stack the columns on mobile by making one full-width and the other half-width -->
<div class="row">
  <div class="col-12 col-md-8">.col-12 .col-md-8</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>

<!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on desktop -->
<div class="row">
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>

<!-- Columns are always 50% wide, on mobile and desktop -->
<div class="row">
  <div class="col-6">.col-6</div>
  <div class="col-6">.col-6</div>
</div>
```

Voici le résultat du code ci-dessous, sur la première figure, sur un écran de type "medium" paramétré avec la classe ".col-md" et sur la deuxième figure, sur un écran de type "extra small" paramétré avec la classe ".col" :

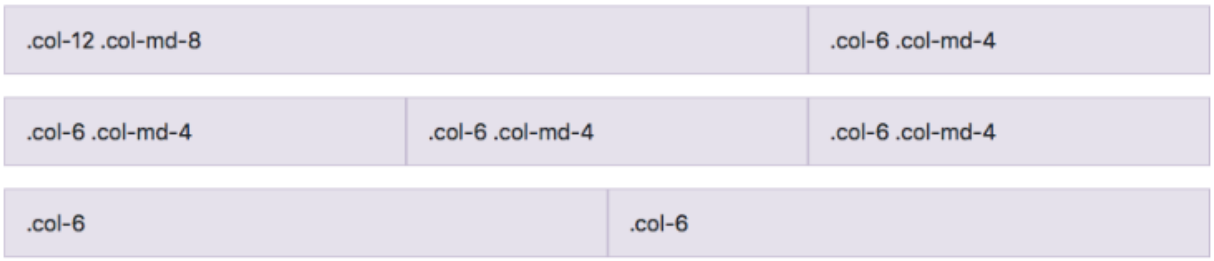


Figure 21 – Représentation de la notion de grilles et colonnes dans Bootstrap [26]

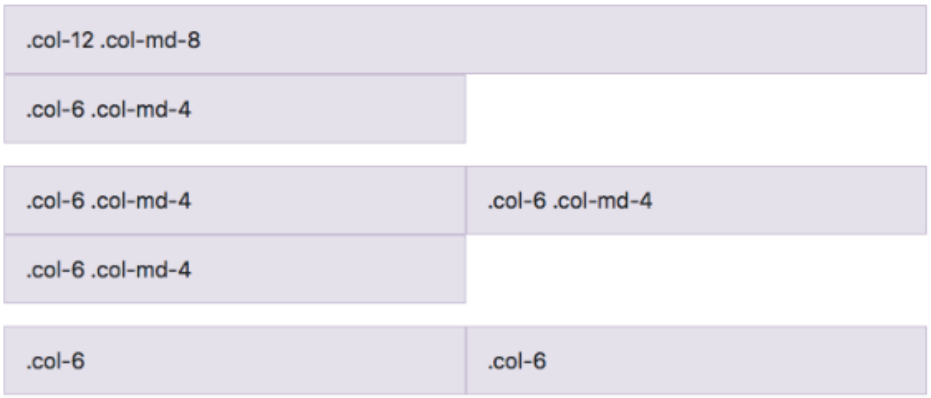


Figure 22 – Représentation de la notion de grilles et colonnes dans Bootstrap [26]

Nous pouvons noter que lorsqu'un élément est défini avec 12 colonnes, il prend tout l'écran, nous retrouvons la notion de grille.

3.3.2 Maquettes

Les maquettes que nous allons vous présenter on été pensées d'une manière responsive pour les différents types d'écrans. Elles représentent la partie jeu et la partie administration de notre quizz.

Les interfaces doivent être simples d'utilisation et compréhensibles rapidement par l'utilisateur. Comme discuté avec notre mandant, moins l'utilisateur fait d'actions, par exemple des cliques, mieux notre application sera.

Accueil

La page d'accueil oriente l'utilisateur entre le mode jeu et le mode administration. Le logo de "Virtual Memory" est mis en avant sur cette page et un footer simple est disposé en bas de page. Cette page se veut responsive, c'est pourquoi nous présentons la maquette en format Desktop et en format Smartphone :

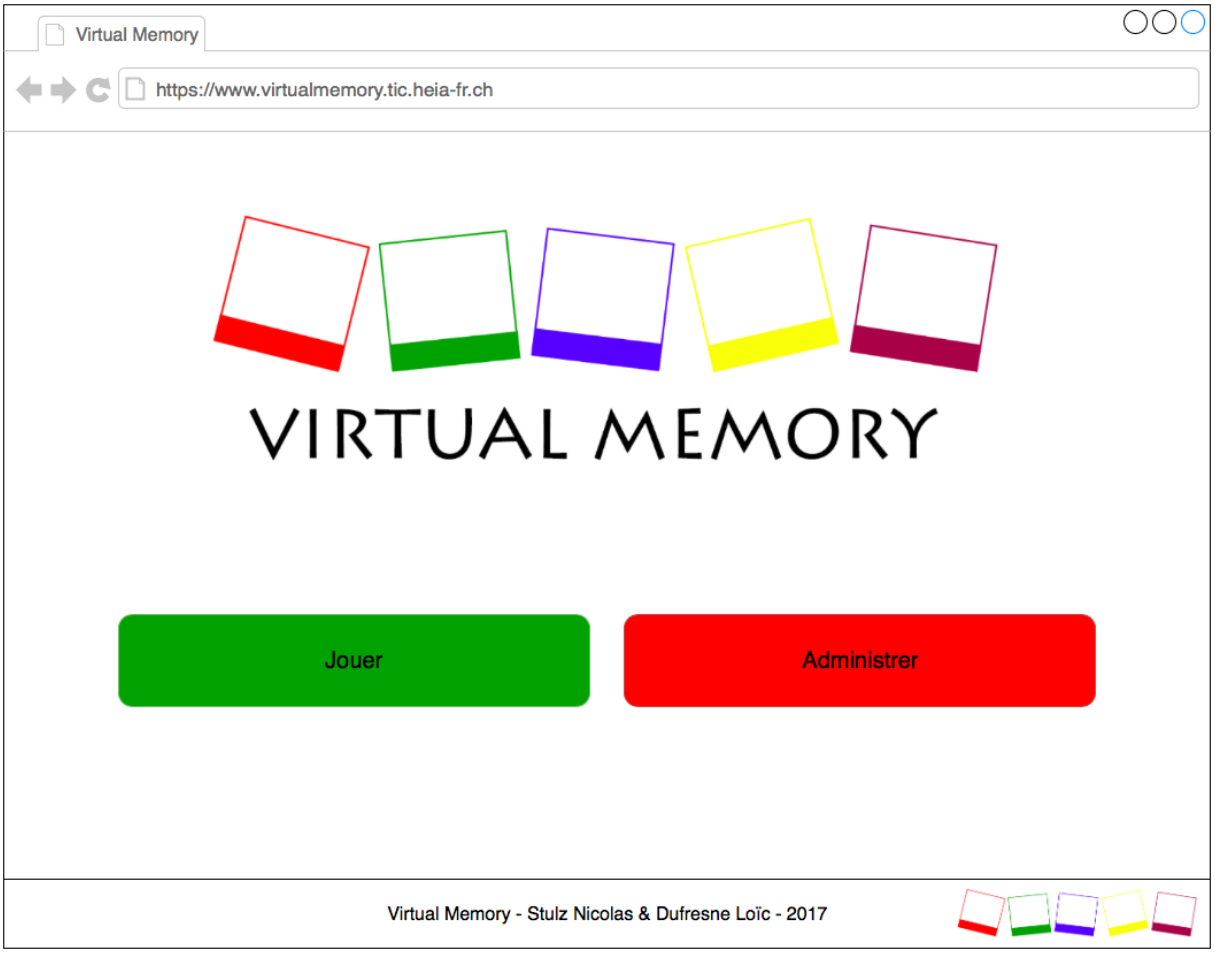


Figure 23 – Maquette de la page d'accueil de notre application sur un Desktop

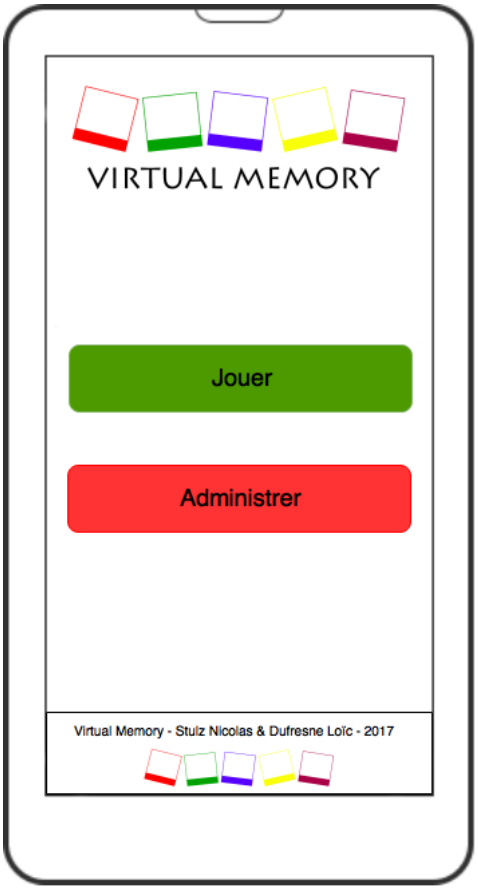


Figure 24 – Maquette de la page d'accueil de notre application sur un Smartphone (mode portrait)

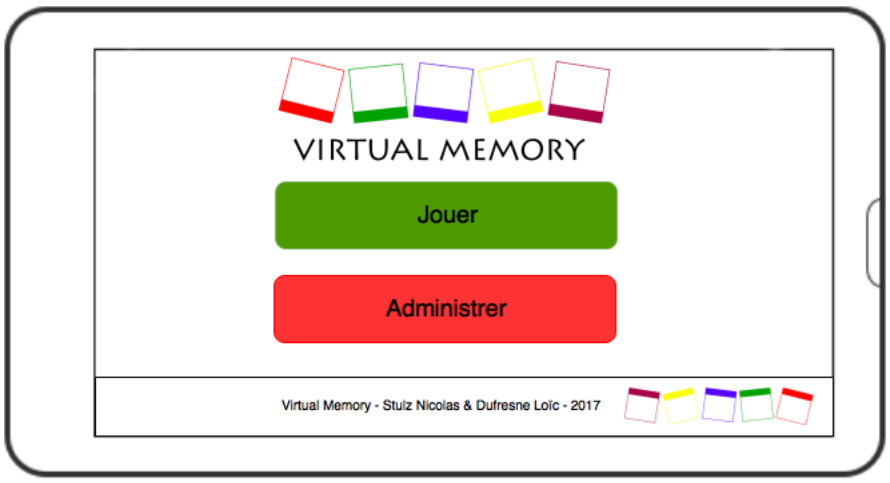


Figure 25 – Maquette de la page d'accueil de notre application sur un Smartphone (mode horizontal)

Partie jeu : choix du thème

Le joueur choisit le quizz qu'il veut jouer, il peut filtrer les quizz par catégories et par types. Le quizz est ensuite généré. Le joueur peut choisir un quizz à la fois.

Figure 26 – Maquette de la partie jeu (choix du thème) de notre application sur un Desktop

Figure 27 – Maquette de la partie jeu (choix du quizz) de notre application sur un Smartphone (mode portrait)

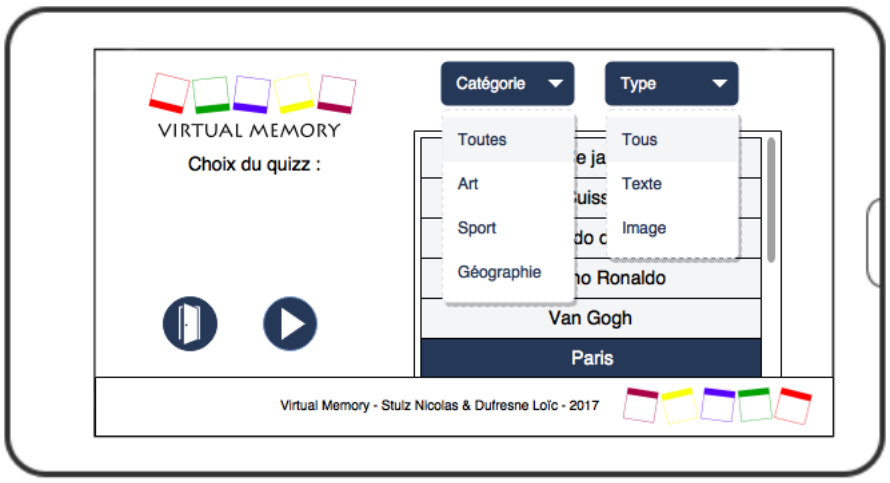


Figure 28 – Maquette de la partie jeu (choix du quizz) de notre application sur un Smartphone (mode horizontal)

Partie jeu : quizz

Le nombre de réponses est généré aléatoirement entre 4 et 8, les réponses sont tirées aussi aléatoirement et il y a au minimum 2 réponses correctes. Le joueur sélectionne ces réponses et valide ces sélections. Une fenêtre affiche les résultats et la fenêtre du choix du prochain quizz s'affiche.

Figure 29 – Maquette de la partie jeu (quizz) de notre application sur un Desktop

Figure 30 – Maquette de la partie jeu (quizz) de notre application sur un Smartphone (mode portrait)

Figure 31 – Maquette de la partie jeu (quizz) de notre application sur un Smartphone (mode horizontal)

Partie administration : ajout/suppression/édition d'une entité

Une première fenêtre d'administration s'ouvre, la liste de toutes les entités s'affiche. L'administrateur peut ajouter une entité, avec un bouton tout le temps actif qui ouvre la deuxième fenêtre d'administration, ou il peut supprimer une entité avec un bouton qui devient actif quand une entité est sélectionnée dans la liste, ou il peut éditer une entité avec un bouton qui devient actif quand une entité est sélectionnée dans la liste et la deuxième fenêtre d'administration. Des filtres permettent de filtrer les entités.

Figure 32 – Maquette de la partie administration, ajout/suppression/édition d'une entité sans sélection d'une entité dans la liste

Figure 33 – Maquette de la partie administration, ajout/suppression/édition d'une entité avec sélection d'une entité dans la liste

Partie administration : modification/gestion des liens d'une entité

Une deuxième fenêtre d'administration s'ouvre. Si l'utilisateur a préalablement choisi d'ajouter une entité, les champs d'une entité sont vides, il peut les remplir et la lier avec d'autres entités avant de valider ces choix. Si l'utilisateur a préalablement choisi d'éditer une entité, il peut gérer ces champs affichés ou ces liens avec d'autres entités. Des filtres permettent de filtrer les entités liées et non liées. Quand tous les champs de l'entité sont remplis, le bouton "Valider" s'active, bien entendu soit le champ texte ou le champ image doit être rempli, pas les deux. Si une entité est sélectionnée dans la liste des entités non liées, le bouton "Lier" s'active. Si une entité est sélectionnée dans la liste des entités liées, le bouton "Délier" s'active.

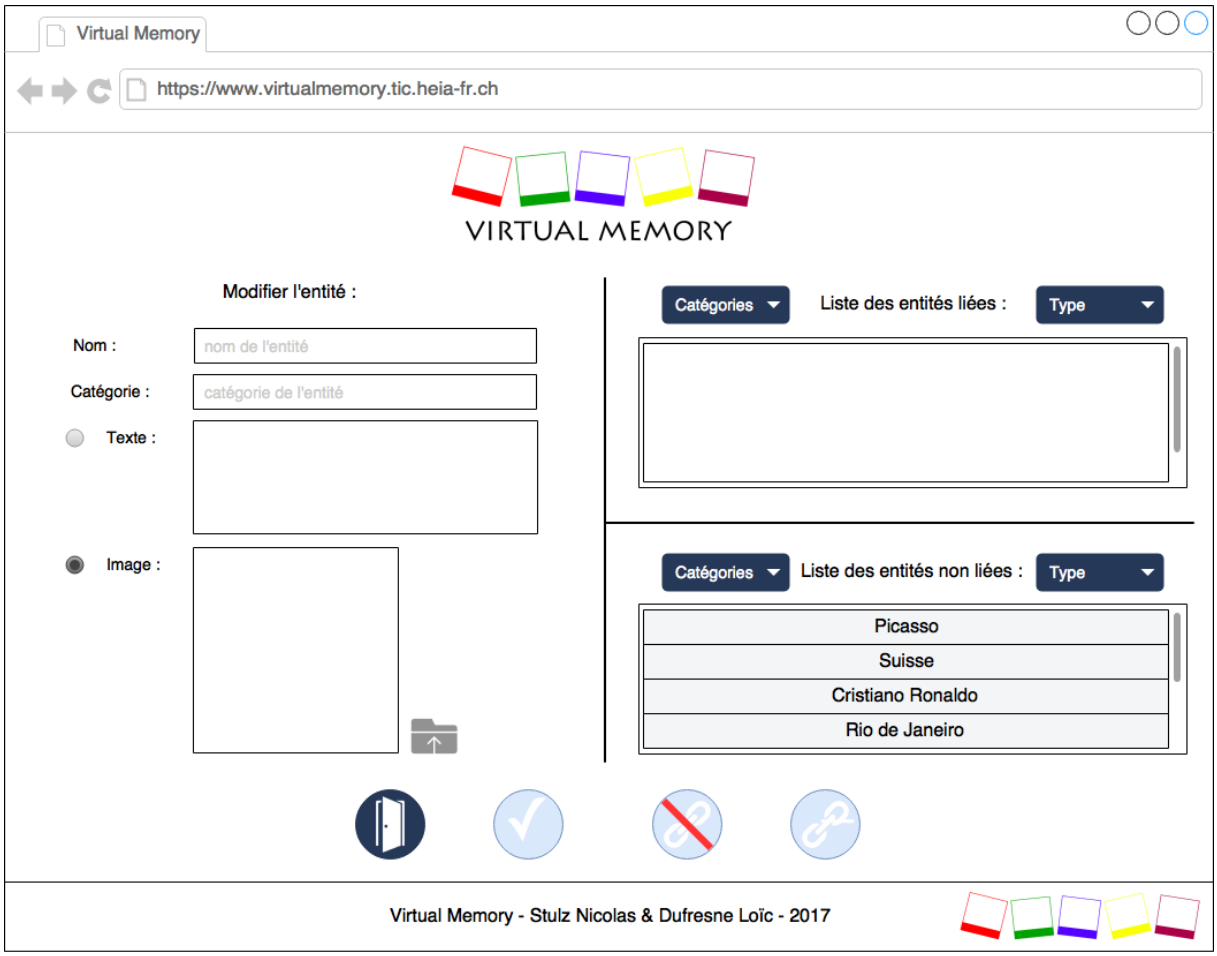


Figure 34 – Maquette de la partie administration, ajout/édition d'une entité (ajout)

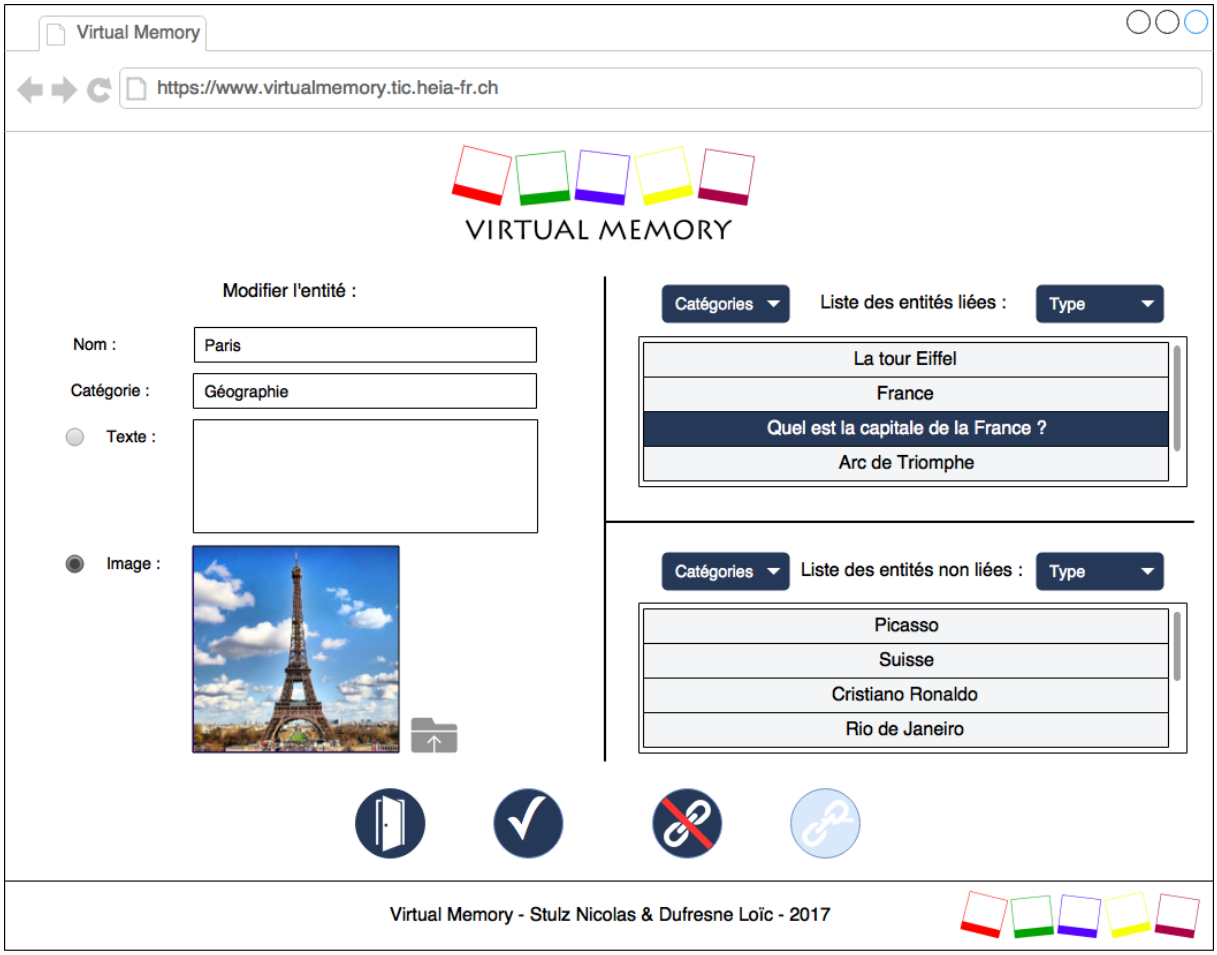


Figure 35 – Maquette de la partie administration, ajout/édition d'une entité (édition)

3.4 UML

Les schémas UML seront réalisés à l'aide du logiciel Visual Paradigm.

3.4.1 Use Case

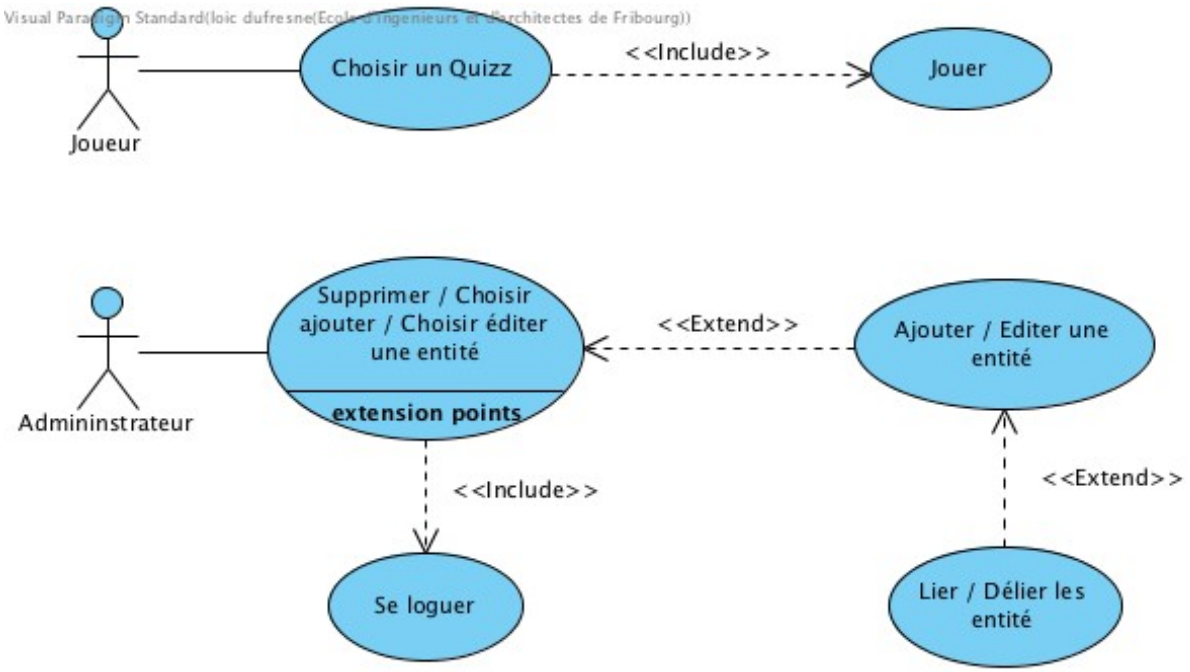


Figure 36 – Use Case de l'application

3.4.2 Fiches descriptives

Cas d'utilisation : Choisir un quizz

- **Sommaire identification**

  Résumé: Le joueur a la possibilité de choisir le quizz auquel il veut jouer.

  Acteur: Joueur

- **Description des enchaînements**

  Précondition: Avoir au moins un quizz listé

  Scénario nominal

  1. Le système affiche "Jouer" ou "Administrer" dans le menu principal
  2. Le joueur clique sur le bouton "Jouer"
  3. Le système liste les quizz
  4. Le joueur choisi un quizz en le sélectionnant, il peut aussi filtrer les quizz par catégorie ou type
  5. Le joueur clique sur le bouton "Play"
  6. «*cas d'utilisation:* Jouer»

- **Enchaînement(s) alternatif(s)**

  A1: Le joueur ne veut plus choisir un quizz

  Démarrage au point 4 ou 5 du scénario nominal

  1. Le joueur clique sur le bouton "Cancel"
  2. Retour au point 1 du scénario nominal

  A2: Le joueur veut administrer

  Démarrage au point 1 du scénario nominal

  1. Le joueur clic sur le bouton "Administrer"
  2. «*cas d'utilisation:* Se loguer»

- **Enchaînement(s) d'exception(s)**

  E1: Il n'y a pas de quizz listé

  1. Le système affiche "L'administrateur doit ajouter un quizz"
  2. Retour au point 1 du scénario nominal

- **Remarque(s) complémentaire(s)**

  Aucune

Cas d'utilisation : Jouer

- **Sommaire d'identification**

  Résumé: Ce scénario permet d'effectuer les actions correspondantes au déroulement du jeu.

  Acteur: Joueur

- **Description des enchaînements**

  Précondition: Avoir cliqué sur le bouton "Jouer"

  Scénario nominal

  1. Le système génère le quizz (nombre aléatoire de questions entre 4 et 8 et questions aléatoires, mais au moins deux de correctes)
  2. Le joueur sélectionne les réponses
  3. Le joueur valide
  4. Le système affiche les scores
  5. Le joueur valide
  6. Le système affiche la page d'accueil

- **Enchaînement(s) alternatif(s)**

  A1: Le joueur ne veut plus choisir un quizz

  Démarrage au point 2 ou 3 du scénario nominal

  1. Le joueur clic sur le bouton "Cancel"
  2. Le système demande une confirmation
  3. Retour au point 3 du scénario nominal du «*cas d'utilisation:* Choisir un quizz»

- **Enchaînement(s) d'exception(s)**

  Aucun

- **Remarque(s) complémentaire(s)**

  Aucune

Cas d'utilisation : Se loguer

- **Sommaire identification**

  Résumé: L'administrateur doit se loguer pour administrer

  Acteur: Administrateur

- **Description des enchaînements**

  Précondition: Avoir choisi d'administrer

  Scénario nominal

  1. Le système affiche une fenêtre demandant le nom d'utilisateur et le mot de passe
  2. L'administrateur entre le bon nom d'utilisateur et le bon mot de passe
  3. Le système contrôle le nom d'utilisateur et le mot de passe
  4. «*cas d'utilisation:* Supprimer/Choisir ajouter/Choisir éditer une entité»

- **Enchaînement(s) alternatif(s)**

  A1: Le joueur entre le mauvais nom d'utilisateur et/ou le mauvais mot de passe

  Démarrage au point 2 du scénario nominal

  1. L'administrateur entre le mauvais mot de passe
  2. Le système affiche mauvais mot de passe
  3. Retour au point 1 du scénario nominal

  A2: Le joueur annule

  Démarrage au point 2 du scénario nominal

  1. L'administrateur clic sur le bouton "Cancel"
  2. Le système demande une confirmation
  3. L'administrateur confirme
  4. Retour au point 1 du scénario nominal du «*cas d'utilisation:* Choisir un quizz»

- **Enchaînement(s) d'exception(s)**

  Aucun

- **Remarque(s) complémentaire(s)**

  Aucune

Cas d'utilisation : Supprimer/Choisir ajouter/Choisir éditer une entité

- **Sommaire identification**

  Résumé: L'administrateur peut ajouter, supprimer et éditer une entité

  Acteur: Administrateur

- **Description des enchaînements**

  Précondition: Avoir cliqué sur le bouton "Administrer", et s'être logué

  Scénario nominal

  1. Le système affiche la page d'administration
  2. Le système liste toutes les entités avec la possibilité de les lister par catégorie ou type
  3. L'administrateur sélectionne une entité dans la liste
  4. Le bouton "Supprimer" devient actif
  5. L'administrateur supprime l'entité sélectionnée avec le bouton "Supprimer"
  6. La base de données supprime l'entité à ces données
  7. Le système rafraîchit la liste des entités

- **Enchaînement(s) alternatif(s)**

  A1: L'administrateur veut ajouter une entité

  Démarrage au point 3 du scénario nominal

  1. L'administrateur clique sur le bouton "Ajouter"
  2. «*cas d'utilisation:* Ajouter/Éditer/Lier/Délier une entité»
  3. Retour au point 1 du scénario nominal

  A2: L'administrateur veut éditer une entité

  Démarrage au point 3 du scénario nominal

  1. L'administrateur sélectionne une entité dans la liste
  2. Le bouton "Éditer" devient actif
  3. L'administrateur clique sur le bouton "Éditer"
  4. «*cas d'utilisation:* Ajouter/Éditer/Lier/Délier une entité»
  5. Retour au point 1 du scénario nominal

  A3: L'administrateur annule

  Démarrage au point 3 ou 4 du scénario nominal

  1. L'administrateur clic sur le bouton "Cancel"
  2. Le système demande une confirmation
  3. L'administrateur confirme
  4. Retour au point 1 du scénario nominal du «*cas d'utilisation:* Choisir un quizz»

- **Enchaînement(s) d'exception(s)**

  Aucun

- **Remarque(s) complémentaire(s)**

  Aucune

Cas d'utilisation : Ajouter/Éditer une entité

- **Sommaire identification**

  Résumé: L'administrateur peut ajouter et éditer une entité

  Acteur: Administrateur

- **Description des enchaînements**

  Pré-condition: Aucune

  Scénario nominal

  1. Le système affiche la page d'administration
  2. Si l'administrateur a cliqué sur le bouton "Ajouter" dans le «*cas d'utilisation:* Supprimer/Choisir ajouter/Choisir éditer une entité», le système affiche un formulaire avec les champs composants une entité vide et les entités non liées
  3. L'administrateur remplit correctement les champs composants l'entité
  4. Le bouton "OK" devient actif
  5. L'administrateur clique sur le bouton "OK"
  6. La base de données ajoute l'entité à ces données
  7. L'entité est ajoutée ou modifiée

- **Enchaînement(s) alternatif(s)**

  A1: L'administrateur veut éditer une entité

  Précondition: Avoir choisi d'éditer une entité

  Démarrage au point 2 du scénario nominal

  1. Si l'administrateur a cliqué sur le bouton "Éditer" dans le «*cas d'utilisation:* Supprimer/Choisir ajouter/Choisir éditer une entité», le système affiche un formulaire avec les champs composants une entité remplie avec les informations de l'entité et les entités non liées et liées
  2. L'administrateur modifie correctement les champs composants l'entité
  3. Le bouton "OK" devient actif
  4. L'administrateur clique sur le bouton "OK"
  5. La base de données met à jour l'entité à ces données
  6. Retour au point 7 du scénario nominal

  A2: L'administrateur veut lier ou délier les entités

  Démarrage au point 1 ou 7 du scénario nominal

  1. «*cas d'utilisation:* Lier/délier une entité»

  A3: L'entité existe déjà

  Démarrage au point 5 du scénario nominal, ou au point 4 de l'enchaînement alternatif A1

  1. Le système affiche "L'entité existe déjà"
  2. L'utilisateur change le nom de l'entité
  3. Retour au point 1 du scénario nominal

A4: L'administrateur annule

Démarrage au point 1 à 7 du scénario nominal, ou au point 1 à 5 de l'enchaînement alternatif A1

1. L'administrateur clic sur le bouton "Cancel"
2. Le système demande une confirmation
3. L'administrateur confirme
4. Retour au point 1 du scénario nominal du «*cas d'utilisation:* Supprimer/Choisir ajouter/Choisir éditer une entité»

- **Enchaînement(s) d'exception(s)**

  Aucun

- **Remarque(s) complémentaire(s)**

  Aucune

Cas d'utilisation : Lier/délier une entité

- **Sommaire identification**

  Résumé: L'administrateur a la possibilité de lier ou délier les entités entre elles

  Acteur: Administrateur

- **Description des enchaînements**

  Précondition: Aucune

  Scénario nominal

  1. L'administrateur sélectionne une entité à lié
  2. Le bouton "Link" devient actif
  3. L'administrateur clique sur le bouton "Link"
  4. Le système rafraîchit les entités liées et non liées
  5. Retour au point 1 du scénario nominal

- **Enchaînement(s) alternatif(s)**

  A1: L'administrateur délier les entités

  Démarrage au point 1 du scénario nominal

  1. L'administrateur sélectionne une entité à lié
  2. Le bouton "Unlink" devient actif
  3. L'administrateur clique sur le bouton "Unlink"
  4. Le système rafraîchit les entités liées et non liées
  5. Retour au point 1 du scénario nominal

- **Enchaînement(s) d'exception(s)**

  Aucun

- **Remarque(s) complémentaire(s)**

  Aucune

3.4.3 Diagramme de séquence

Cas d'utilisation : Choisir un quizz

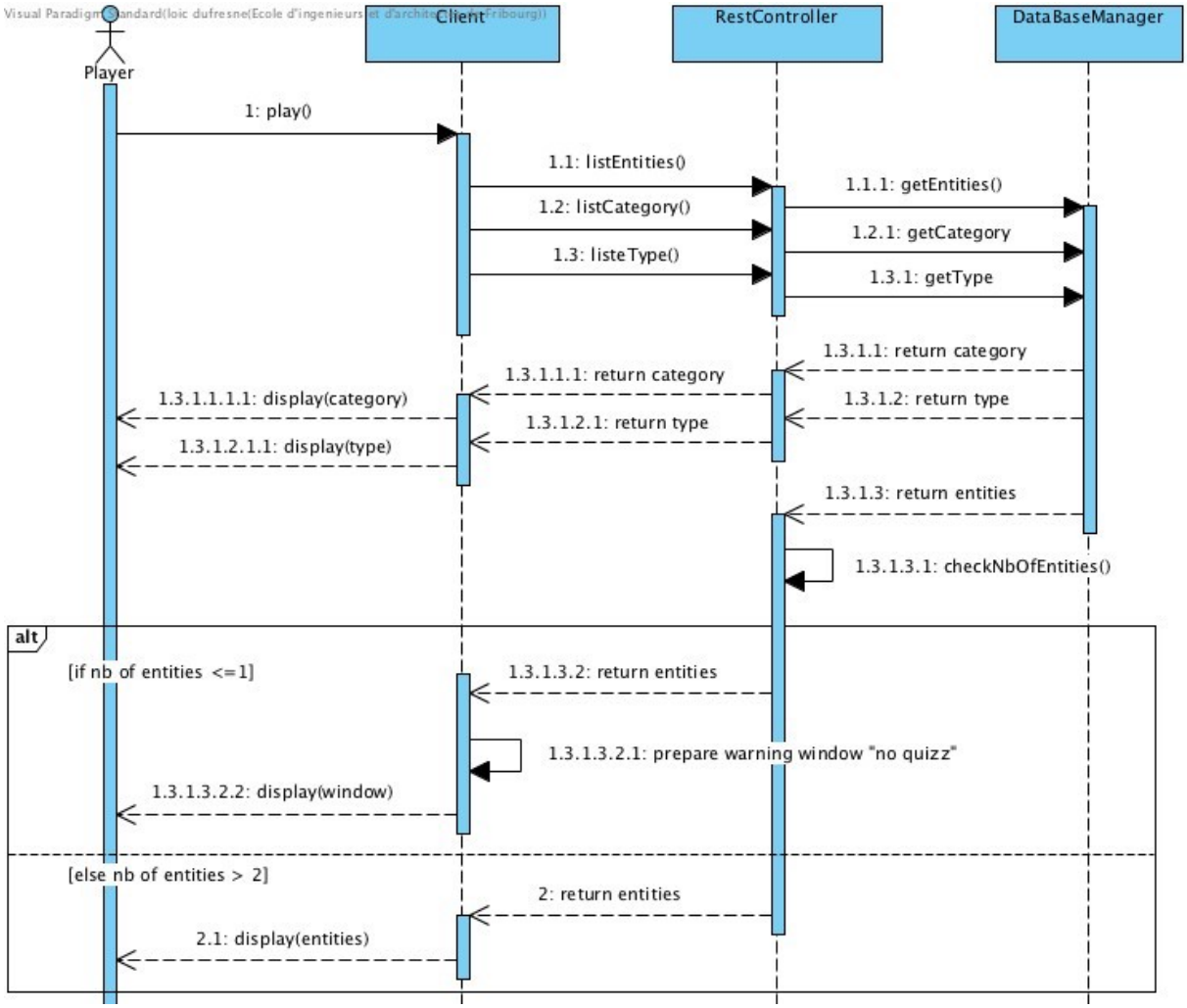


Figure 37 – Diagramme de séquence : Choisir un quizz (partie 1)

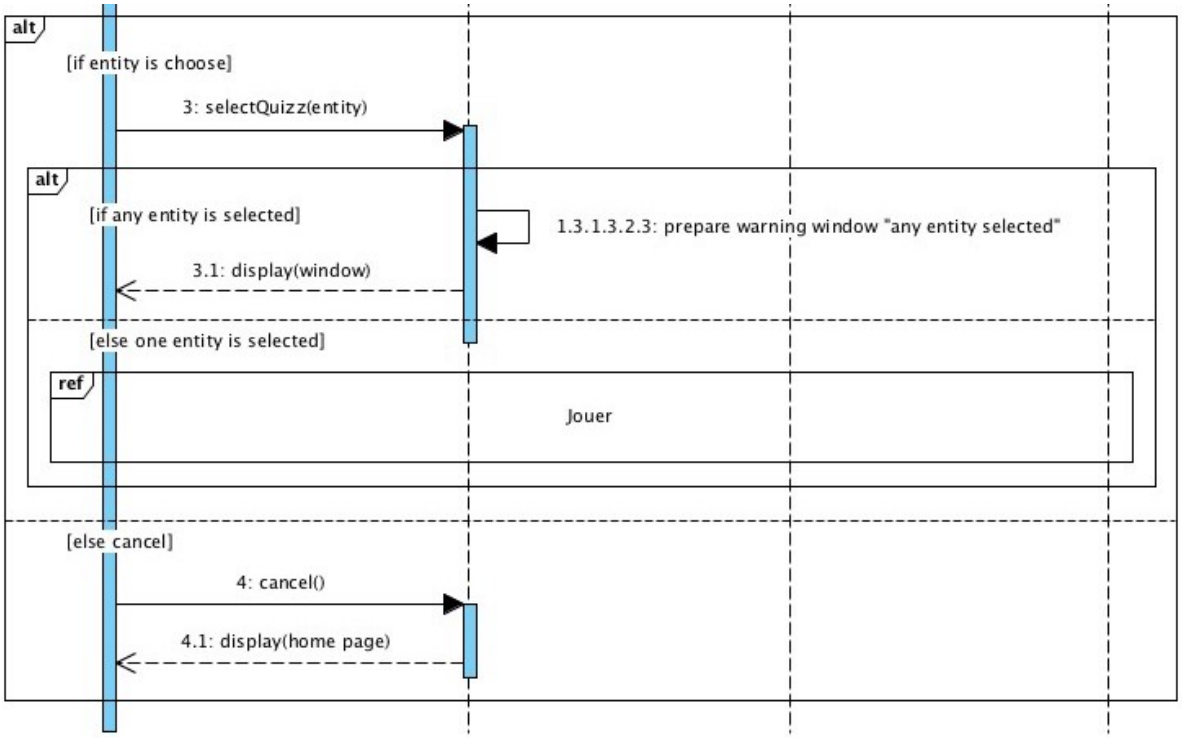


Figure 38 – Diagramme de séquence : Choisir un quizz (partie 2)

Cas d'utilisation : Jouer

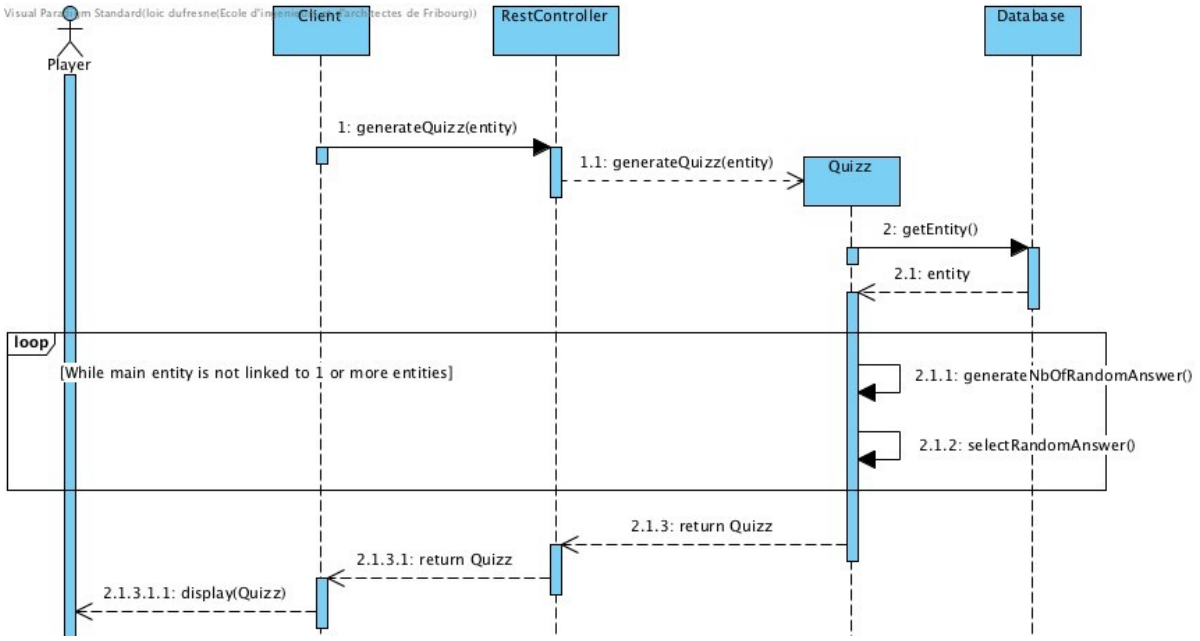


Figure 39 – Diagramme de séquence : Jouer (partie 1)

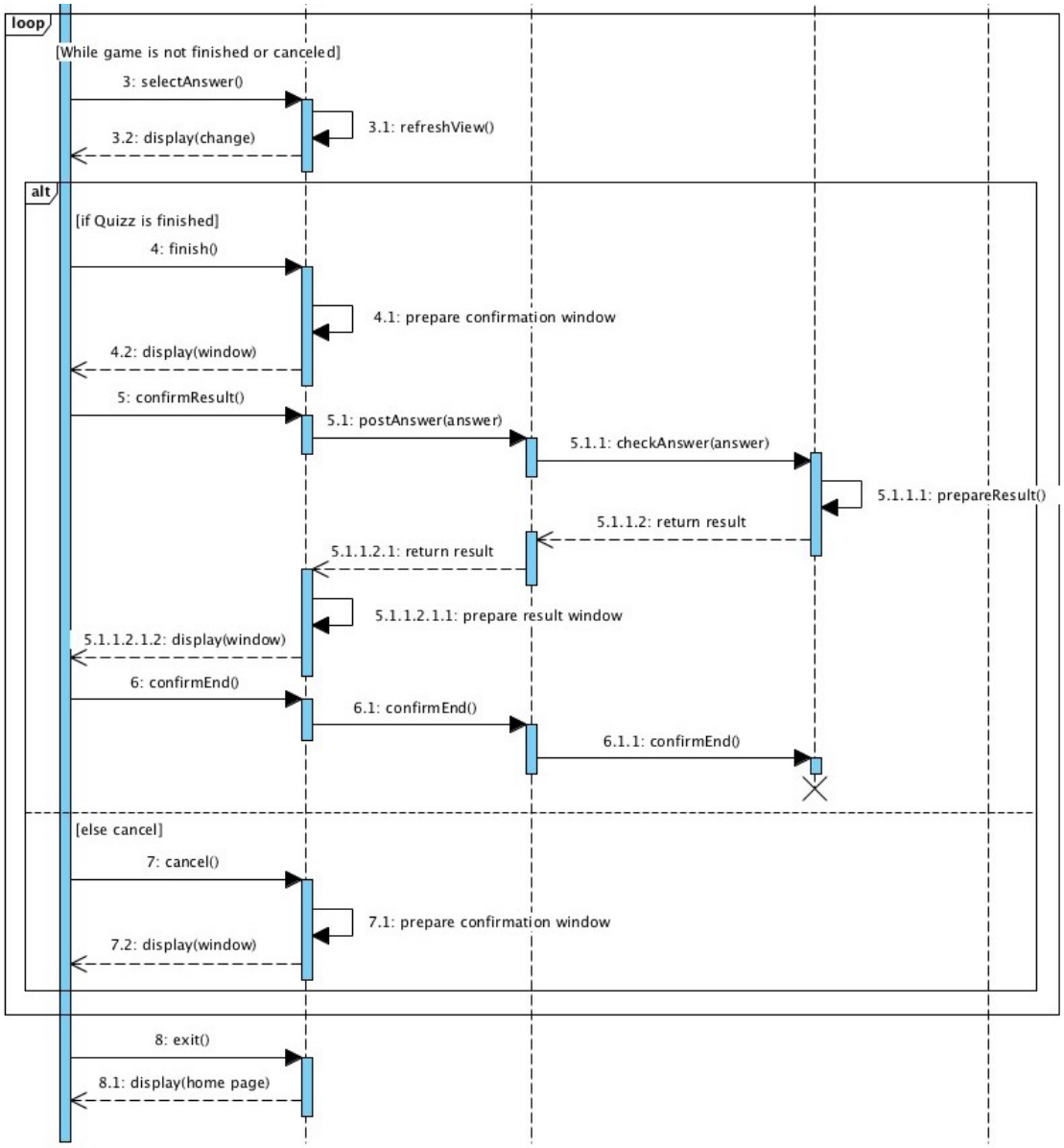


Figure 40 – Diagramme de séquence : Jouer (partie 2)

Cas d'utilisation : Se loguer

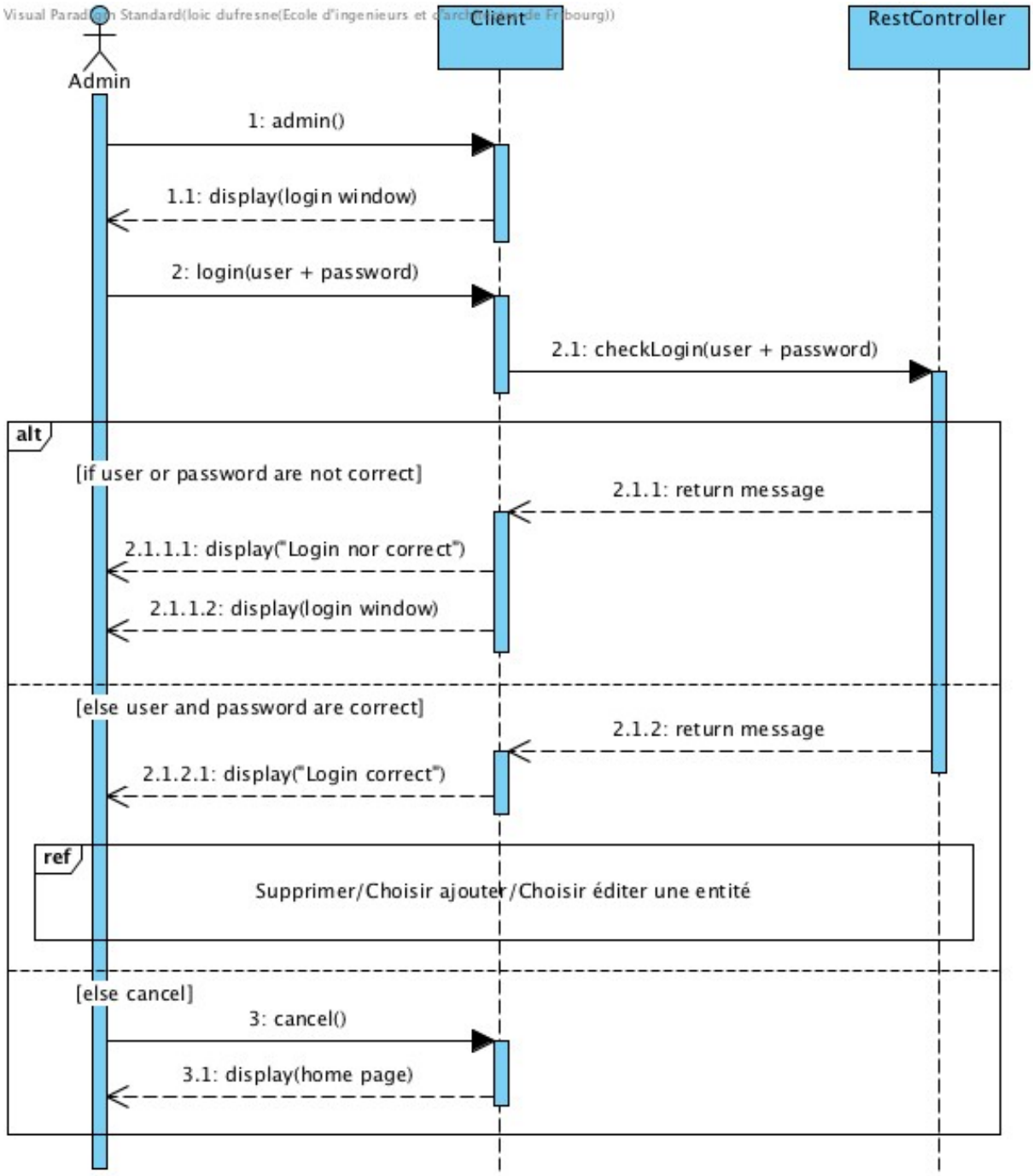


Figure 41 – Diagramme de séquence : Se loguer

Cas d'utilisation : Supprimer/Choisir ajouter/Choisir éditer une entité

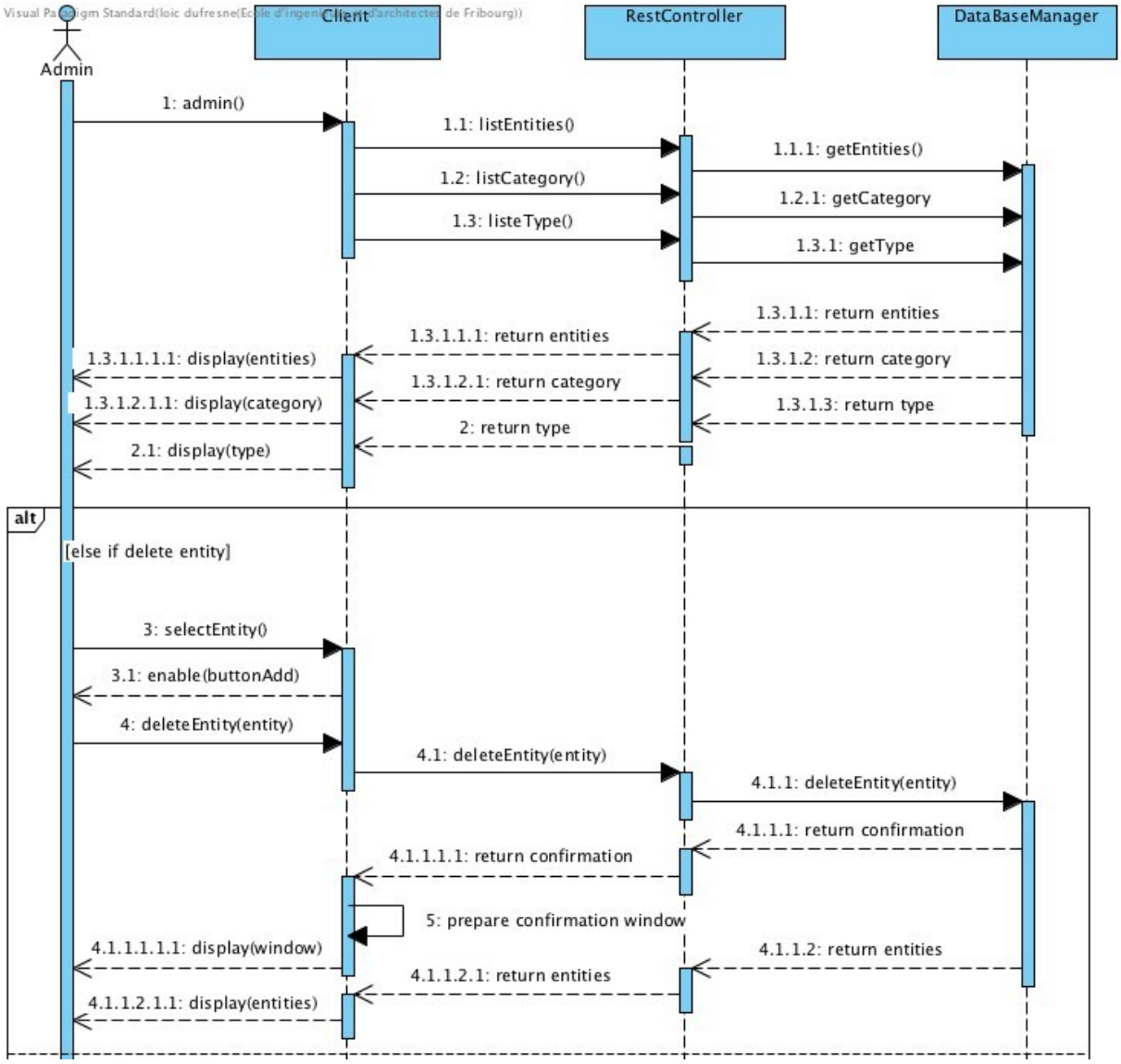


Figure 42 – Diagramme de séquence : Supprimer/Choisir ajouter/Choisir éditer une entité (partie 1)

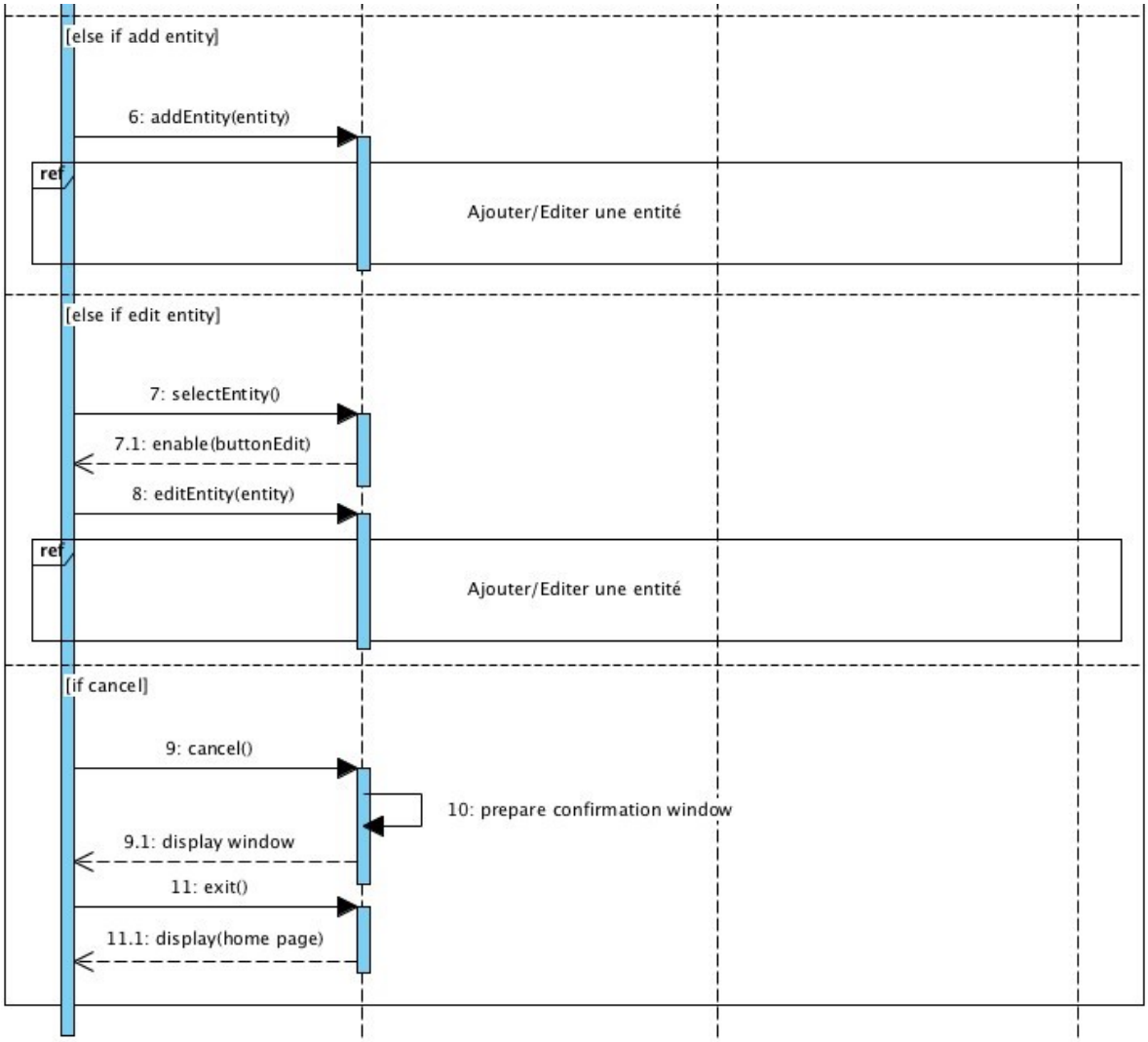


Figure 43 – Diagramme de séquence : Supprimer/Choisir ajouter/Choisir éditer une entité (partie 2)

Cas d'utilisation : Ajouter/Éditer une entité

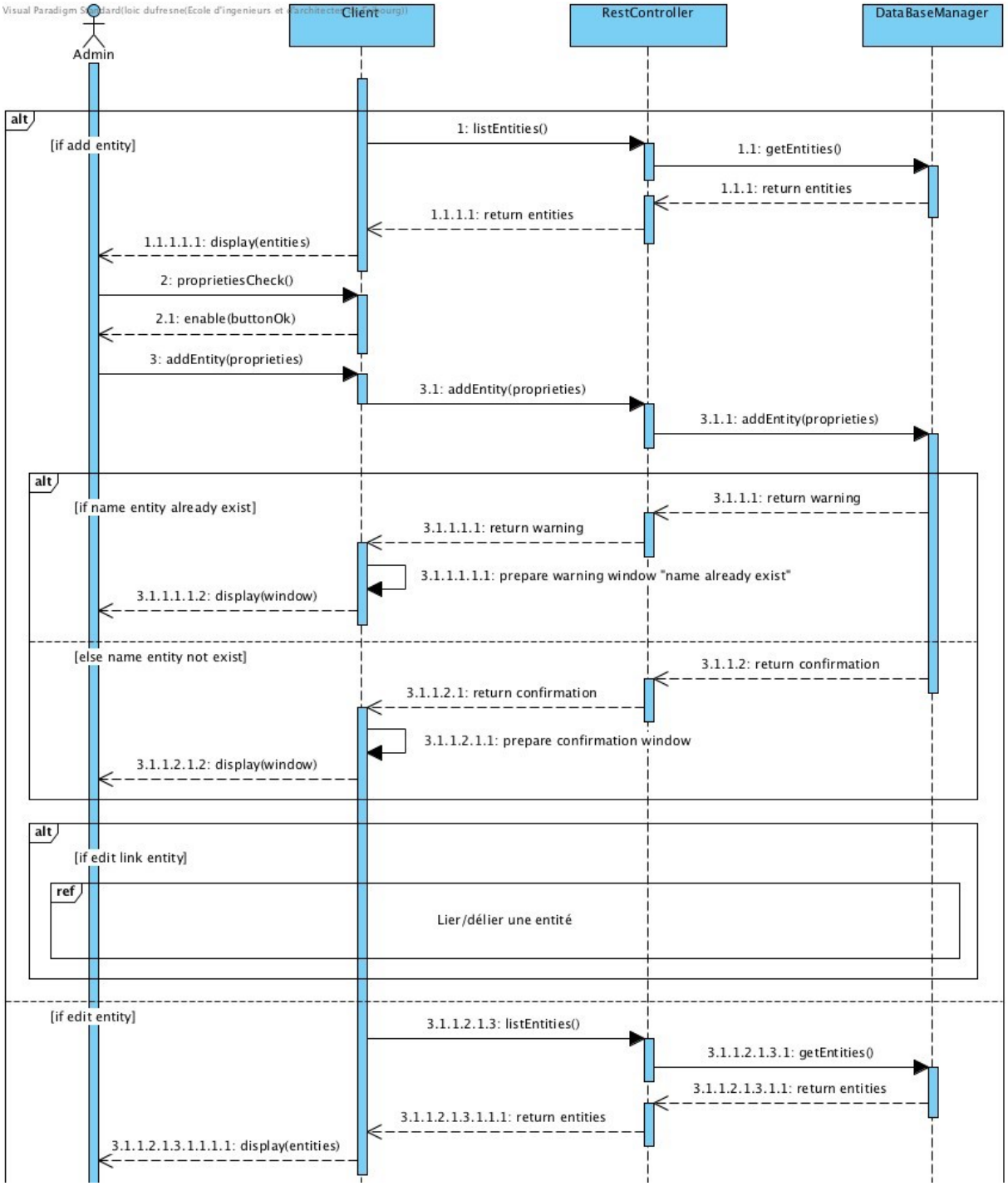


Figure 44 – Diagramme de séquence : Ajouter/Éditer une entité (partie 1)

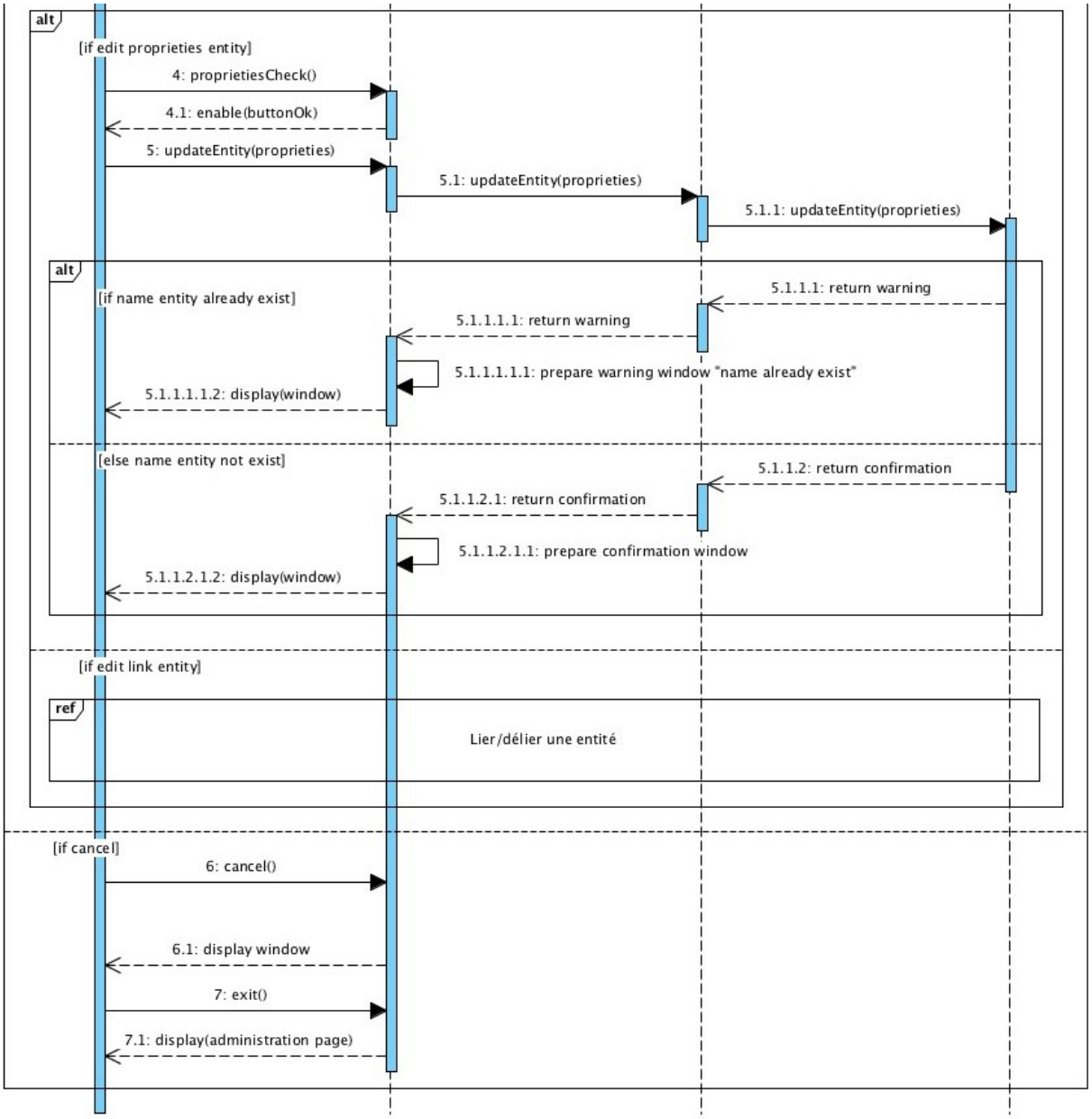


Figure 45 – Diagramme de séquence : Ajouter/Éditer une entité (partie 2)

Cas d'utilisation : Lier/Délier une entité

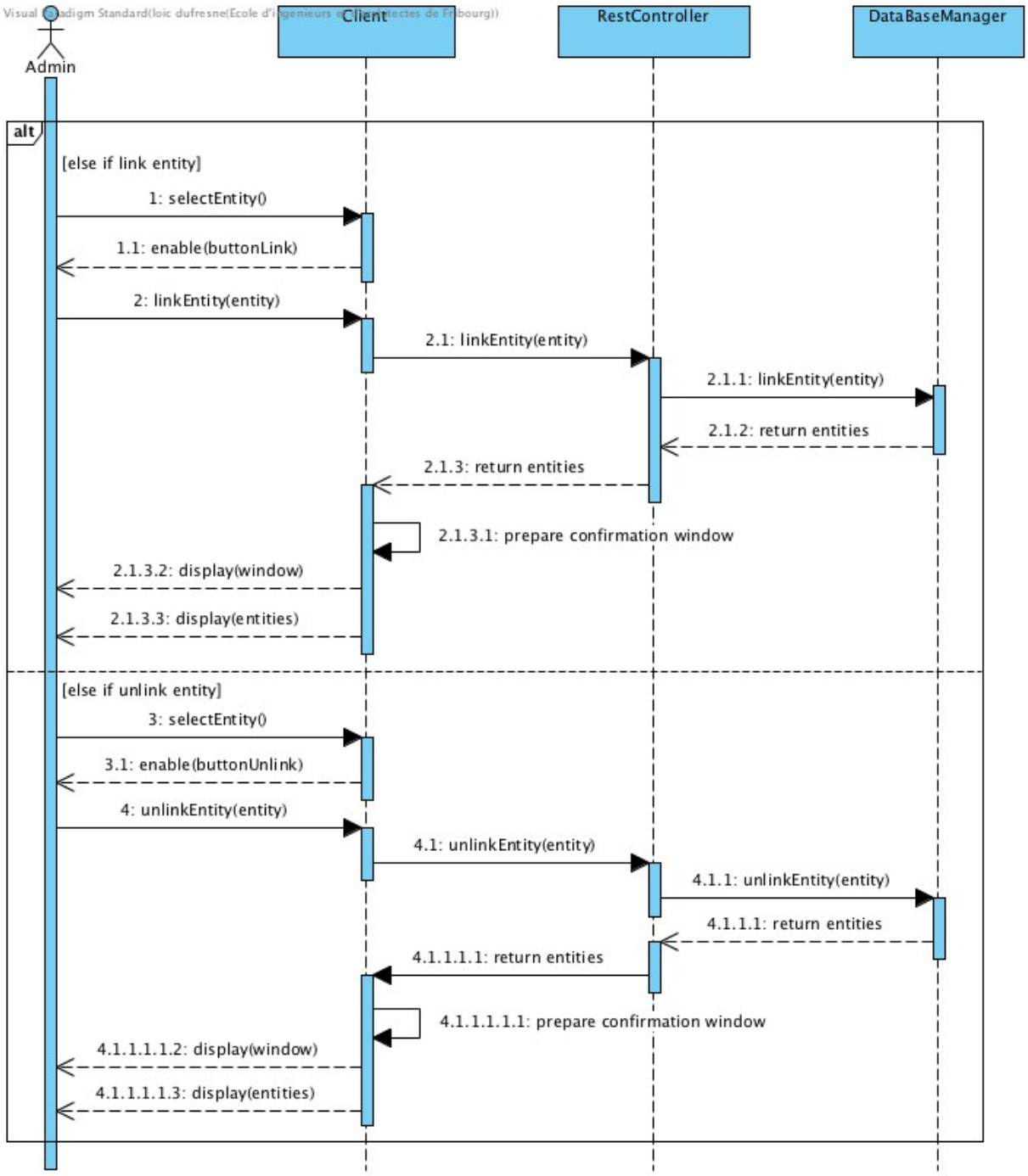


Figure 46 – Diagramme de séquence : Lier/Délier une entité

3.4.4 Diagramme de classe

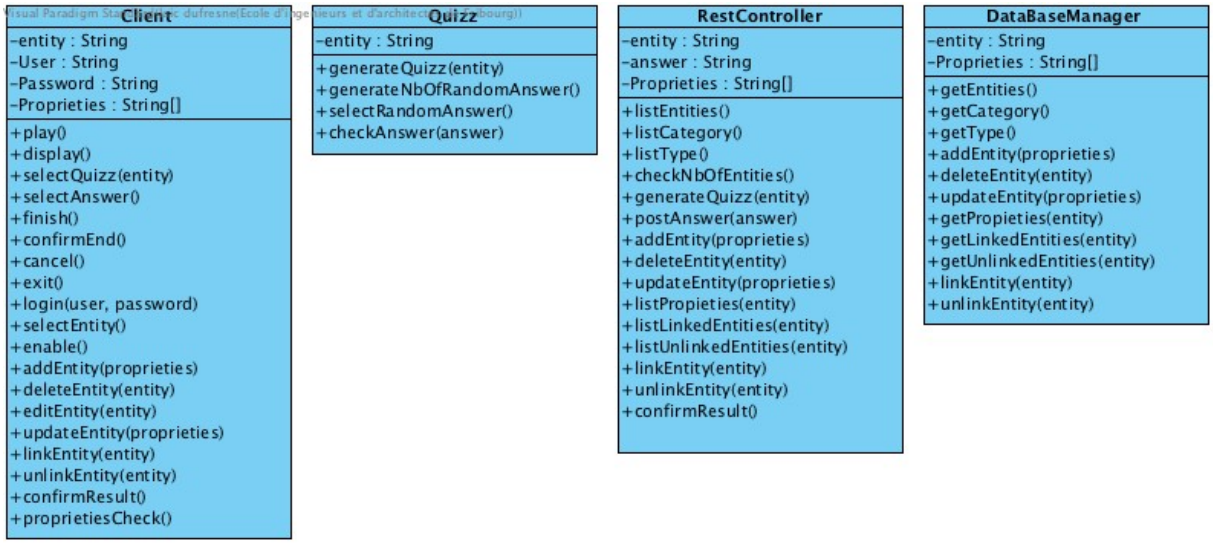


Figure 47 – Diagramme de classes

3.5 Base de données et requêtes

3.5.1 Noeud

Nous allons définir les noeuds (ou entités) et les requêtes dont nous aurons besoin pour l'implémentation de notre application.

Dans un premier temps, nous étions partis sur un noeud à quatre champs:

- **name:** pour le nom du noeud
- **cat:** pour la catégorie du noeud afin de mettre en place des filtres
- **text:** pour le texte de l'entité qui apparaît pendant le jeu
- **img:** pour l'image de l'entité qui apparaît pendant le jeu

Et cette entité peut avoir soit un champ "text", soit un champ "img".

Nous avons ensuite eu une discussion avec Madame Chabbi et elle nous a conseillé de penser la chose différemment. Concernant le champ définissant la catégorie du noeud, nous pouvons la supprimer afin de le remplacer par un label. Un label peut être attribué à un noeud lors de sa création. Ce qui est intéressant, c'est qu'une entité peut se voir attribuer un ou plusieurs label. Nous avons donc décidé d'utiliser les labels pour les catégories et nous allons effectuer des tests afin de voir si cette solution est avantageuse pour nous.

Autre changement, nous n'aurons plus un champ pour le texte et un champ pour l'image dans une entité, nous n'aurons plus qu'un champ pour le contenu pouvant contenir du texte ou une image.

Nous pensions mettre en place un ID pour chacun des noeuds, mais comme nous avons pu le voir, mettre en place un ID numérique qui s'auto-incrémente, comme avec les bases de données relationnelles, n'est pas supporté directement par Neo4j et donc assez compliqué à mettre en place. Nous avons décidé d'utiliser le champ définissant le nom du noeud comme l'ID de ce dernier, nous ne créerons pas directement un noeud avec une contrainte ID et unique pour son nom, mais à chaque création d'un nouveau noeud, nous contrôlerons dans la base de données si le nom n'existe pas déjà, ce qui nous garantira un nom unique pour les noeuds. Cette manière de faire est précisée pour cette version de l'application, car cela nous simplifie la gestion des noeuds lors des diverses requêtes comme une suppression d'un noeud. Bien entendu, le fait de gérer des entités avec plusieurs mêmes noms peut être un point d'amélioration.

Voici notre entité finale avec un label et deux champs :

- **name:** pour le nom du noeud
- **label:** pour la catégorie du noeud afin de mettre en place des filtre
- **content:** pour le texte ou l'image de l'entité qui apparaît pendant le jeu

Concernant le stockage des images dans la base de données, les images seront converties et stockées dans la base de données en format Base64. Nous avons effectué des tests et il est vrai qu'avec des images de grande qualité, les performances de la base de données sont vite diminuées. Les images qui apparaîtront lors du jeu n'ont pas besoin d'être de haute qualité et elles seront affichées dans le jeu avec une taille de 150px par 150px. Il est vrai que les bases de données Graph ne sont pas vraiment faites pour supporter de gros contenus.

Voilà la requête que nous utiliserons pour créer un noeud, dans cette exemple, un label représentant la catégorie "Géographie", le nom "Paris" et un court texte comme contenu :

```
CREATE (:Node{ name: 'Paris', content: 'La ville des lumieres'})
```

Comme exemple, voici aussi la requête permettant de créer un noeud avec plusieurs labels, donc les catégories "Géographie", "Art" et "Ville" :

```
CREATE (:Geographie:Art:Ville{ name: 'Paris', content: 'La ville des lumieres'})
```

3.5.2 Liens

Après la définition des noeuds, nous avons défini leurs liens. L'idée est que lorsque l'entité A connaît l'entité B, obligatoirement l'entité B connaît l'entité A, les liens seront bidirectionnels. Si nous prenons deux exemples, au niveau du thème de la géographie premièrement, Paris connaît la France comme étant son pays et la France connaît Paris comme étant sa capitale. Dans un second exemple, au niveau des ressources humaines, nous définissions une personne comme étant motivée, nous pouvons vouloir connaître la qualité de cette personne qui est d'être motivée et nous pouvons vouloir savoir quelles sont les personnes étant motivées, qui est entre autres cette personne. Dans ces exemples, le lien se fait dans les deux sens.

Par contre, dans une amélioration future, il n'est pas à mettre de côté le fait que l'administrateur pourrait gérer et choisir le type de liens, unidirectionnels ou bidirectionnels entre-deux entités.

Il y a aussi la possibilité de créer plusieurs types de liens différents, mais comme nous l'avons déjà défini, les entités seront liées avec un seul type de lien (KNOWS). A noter aussi que dans une amélioration future, le jeu pourrait entièrement se jouer avec les liens, nous aborderons ce sujet dans les améliorations.

Voici les requête que nous utiliserons pour créer nos liens bidirectionnels, dans cet exemple, nous vous lier le noeud "Paris" et le noeud "France" avec un type de lien "KNOWS" et bidirectionnel :

```
MATCH (a{name:'Paris'})
   MATCH (b{name: 'France'})
   MERGE (a)-[:KNOWS]->(b)
   MERGE (b)-[:KNOWS]->(a)
```

Voilà le résultat sous plusieurs forment dans Neo4j après la création et le liage des deux noeuds :

Figure 48 – Création et liage de noeuds

Dans un premier nous "matchons" les noeuds afin de les récupérer et créer les liens avec un "merge", si nous avions créer le lien directement sans le "match", les noeuds se créent à double.

Nous "matchons" le noeud avec son nom, ici le fait que le nom soit unique dans la base de données est primordial, car si nous avons deux noeuds avec le même nom, lors d'une création de lien avec un autre noeud, les deux entités avec le même nom se verraient liées à ce même noeud.

3.5.3 Requêtes

Pour ce point, nous avons créé une base de données avec les requêtes mentionnées plus haut pour la création et le liage des entité, voici cette base de données qui nous servira de base pour l'implémentation du projet:

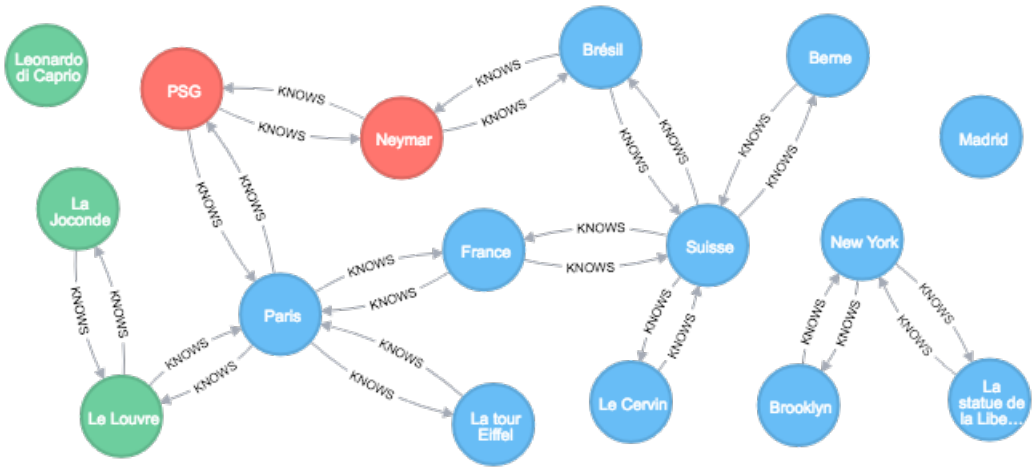


Figure 49 – Base de données de base pour l'implémentation

Nous avons maintenant défini les requêtes dont nous aurons besoin dans l'application:

- Ajouter un noeud "Paris":

```
CREATE (:Geographie{ name: 'Paris', content: 'La ville des lumieres'})
```

- Lier le noeud "Paris" et le noeud "France":

```
MATCH (a{name:'Paris'})
   MATCH (b{name: 'France'})
   MERGE (a)-[:KNOWS]->(b)
   MERGE (b)-[:KNOWS]->(a)
```

- Supprimer le noeud "Paris" et ces relations (avant de supprimer un noeud, il faut supprimer ces liens):

```
MATCH (n { name: 'Paris' })
DETACH DELETE n
```

- Supprimer le noeud "Paris" (avant de supprimer un noeud, il faut supprimer ces liens):

```
MATCH (a{name: 'Paris'})
DELETE a
```

- Supprimer les liens entre le noeud "Paris" et le noeud "France" (supprime la bidirectionnalité):

```
MATCH (n { name: 'Paris' })-[r:KNOWS]-(m { name: 'France' }) DELETE r
```

- Supprimer tous les liens du noeud "Paris" allant vers le(s) noeuds X et les liens des noeuds X allant vers le noeud "Paris" (supprime la bidirectionnalité):

```
MATCH (n { name: 'Paris' })-[r:KNOWS]-()
DELETE r
```

- Éditer le noeud "Paris" en changeant son nom par "Rome":

```
MATCH (n { name: 'Paris' })
SET n.name = 'Rome'
```

- Retourner tous les noeuds par ordre alphabétique:

```
MATCH (n) RETURN n.name, n.content
ORDER BY n.name
```

- Retourner le nombre de noeuds que le noeud "Paris" connaît:

```
MATCH (n { name: 'Paris' })-[r:KNOWS]->()
RETURN COUNT(r)
```

- Retourner les noeuds que le noeud "Paris" connaît (cette requête nous sera utilisée pour vérifier les bonnes réponses lors du quizz):

```
MATCH (n { name: 'Paris' })-[r:KNOWS]->(m)
RETURN m
```

Figure 50 – Les noeuds que le noeud "Paris" connaît

- Retourner tous les noeuds qui ont plus de 1 lien vers d'autres noeuds (cette requête nous sera utilisée pour afficher les quizz):

```
MATCH (n)-->() WITH n,
COUNT(*) AS rel_cnt
WHERE rel_cnt > 1
RETURN n.name
```

n.name
"New York"
"Le Louvre"
"Brésil"
"Suisse"
"PSG"
"Paris"
"France"
"Neymar"

Figure 51 – Les noeuds qui ont plus de 1 lien vers les autres noeuds

Pour des fonctionnalités futures, il y aura peut-être besoin d'utiliser ces requêtes :

- Supprimer le lien entre le noeud "Paris" et le noeud "France" (fait venir la notion de liens unidirectionnels):

```
MATCH (n { name: 'France' })-[r:KNOWS]->(m { name: 'Pris' })
DELETE r
```

- Supprimer tous les liens du noeud "Paris" allant vers le(s) noeuds X (fait venir la notion de liens unidirectionnels):

```
MATCH (n { name: 'Paris' })-[r:KNOWS]->()
DELETE r
```

- Retourner tous les types de liaison (dans notre cas retournera seulement "KNOWS"):

```
MATCH ()-[r]-()
RETURN DISTINCT TYPE(r)
```

- Supprimer tous les noeuds et les liens de la base de données:

```
MATCH (n)
DETACH DELETE n
```

3.5.4 Labels

Nous avons vu précédemment les principales requêtes dont nous avons besoin pour l'implémentation de l'application afin de communiquer avec la base de données. Cependant, il y a un point que nous n'avons pas réussi à éclaircir. Afin d'attribuer une catégorie à un noeud, nous utilisons un label, mais nous avons remarqué que le fait de requêter des noeuds par rapport aux labels et extrêmement difficile et nous n'arrivons pas à faire ce que nous voulons avec ces labels, par exemple, filtrer les noeuds par catégorie. La documentation sur les requêtes étant assez faible sur Neo4j et le temps compté, après plusieurs jours de recherches, nous avons décidé d'abandonner les labels sur les noeuds et nous repartons avec notre idée de base et attribuant un label à un noeud grâce à un champ.

Bien entendu, les requêtes présentées ci-dessus restent correctes avec ce changement, nous allons juste redéfinir la requête permettant de créer un noeud :

- Ajouter un noeud "Paris" :

```
CREATE (:node{ name: 'Paris', category: 'Geographie' ,content: 'La ville des
    lumieres'})
```

Nous sommes donc revenus en arrière sur la définition d'un noeud, un noeud comprend 3 champs :

- **name:** pour le nom du noeud
- **category:** pour la catégorie du noeud afin de mettre en place des filtre
- **content:** pour le texte ou l'image de l'entité qui apparaît pendant le jeu

Maintenant, nous pouvons requêter avec les catégories plus facilement :

- Retourner les noeuds qui ont pour catégorie "Géographie" (cette requête nous sera utilise pour filtrer les entités par catégories) :

```
MATCH (n) WHERE n.category = 'Geographie' RETURN n.name
```

3.6 Conclusion de la conception

Pour conclure cette partie, nous pouvons voir que la suite logique Use Case, fiches descriptives, diagrammes de classes et diagrammes de séquences n'est pas vraiment simple à faire, mais une fois tous ces éléments réalisés, ils nous seront vraiment utiles pour la partie suivante qui est la partie implémentation. C'est pourquoi il est nécessaire de bien penser et de bien créer ces diagrammes.

Nous avons réalisé plusieurs versions des maquettes, ce qui nous a amenés à modifier tous les diagrammes qui en découlaient. À force de faire, de nouvelles idées naissent surtout au niveau des maquettes, les maquettes ont été changées au dur et à mesure des séances, ce qui nous a apporté des solutions plus simples et plus efficaces. Il est aussi clair que lors de la partie implémentation, d'autres idées viendront et nos maquettes ne ressembleront peut-être plus à celles de base.

4 Implémentation

Ce chapitre traite de l'implémentation de notre projet. Nous ne présenterons pas tout le code de notre application, mais nous présenterons et expliquerons les parties intéressantes et qui requièrent une attention particulière.

4.1 Étapes de l'implémentation

Afin de mieux structurer l'implémentation de l'application de notre projet, nous avons définit les étapes à réaliser:

- **Page "Accueil":**
  - Créer la page « Accueil »
  - Afficher le bouton « Administrer » afin d'accéder à la page « Choix du quizz »
  - Afficher le bouton « Jouer » afin d'accéder à la page « Choix du quizz »
  - Gérer le côté Responsive
- **Page "Choix du thème" (partie jeu) :**
  - Lister les catégories (filtre)
  - Lister les contenus (filtre)
  - Lister les quizz (entités)
  - Lister les quizz (entités) en fonction du filtre des catégories
  - Lister les quizz (entités) en fonction du filtre des contenus
  - Sélectionner un quizz (entité) afin d'accéder à la page « Quizz »
  - Afficher le bouton « Retour » afin de revenir à la page « Accueil »
  - Gérer le côté Responsive
- **Page "Quizz" (partie jeu):**
  - Afficher le quizz (entités) avec des bonnes et des mauvaises réponses
  - Sélectionner les réponses (entités)
  - Afficher le bouton « Valider » afin de vérifier les réponses (entités) cochées
  - Afficher le résultat du quizz
  - Afficher le bouton « Retour » afin de revenir à la page « Choix du quizz »
  - Gérer le côté Responsive

- **Page "Admin" (partie administration):**
  - Lister les catégories (filtre)
  - Lister les contenus (filtre)
  - Lister les quizz (entités)
  - Lister les quizz (entités) en fonction du filtre des catégories
  - Lister les quizz (entités) en fonction du filtre des contenus
  - Afficher les boutons « Supprimer » dans la liste des quizz (entités) afin de supprimer le quizz sélectionné
  - Afficher un message d'alerte permettant de confirmer la suppression d'un quizz (entité)
  - Afficher les boutons « Editer » dans la liste des quizz (entités) afin de modifier le quizz sélectionné
  - Afficher le bouton « Ajouter » afin d'accéder à la page « Administration 2a »
  - Afficher le bouton « Retour » afin de revenir à la page « Accueil »
  - Gérer le côté Responsive
- **Page "Create" (partie administration) :**
  - Afficher le champ « Nom » vide
  - Afficher le champ « Catégorie » vide
  - Afficher un moyen de choisir entre un contenu ou une image
  - Afficher le champ « Contenu » vide
  - Afficher le téléchargement d'image
  - Lister les catégories (filtre)
  - Lister les contenus (filtre)
  - Lister les quizz liés (entités liées)
  - Lister les quizz liés (entités liées) en fonction du filtre des catégories
  - Lister les quizz liés (entités liées) en fonction du filtre des contenus
  - Afficher le bouton « Ajouter » afin d'ajouter le quizz (entité)
  - Afficher le bouton « Retour » afin de revenir à la page « Administration 1 »
  - Gérer le côté Responsive

- **Page "Edit" (partie administration) :**
  - Afficher le champ « Nom » avec le nom du quizz (entité)
  - Afficher le champ « Catégorie » avec la catégorie du quizz (entité)
  - Afficher un moyen de choisir entre un contenu ou une image
  - Afficher le champ « Contenu » avec le contenu du quizz (entité)
  - Afficher le téléchargement d'image avec l'image du quizz (entité)
  - Lister les catégories (filtre)
  - Lister les contenus (filtre)
  - Lister les quizz liés (entités liées)
  - Lister les quizz liés (entités liées) en fonction du filtre des catégories
  - Lister les quizz liés (entités liées) en fonction du filtre des contenus
  - Afficher les boutons « Délier » dans la liste des quizz (entités) afin de délier un quizz
  - Lister les catégories (filtre)
  - Lister les contenus (filtre)
  - Lister les quizz non liés (entités non liées)
  - Lister les quizz non liés (entités non liées) en fonction du filtre des catégories
  - Lister les quizz non liés (entités non liées) en fonction du filtre des contenus
  - Afficher les boutons « Lier » dans la liste des quizz (entités) afin de lier un quizz
  - Afficher le bouton « Valider » afin d'ajouter le quizz (entité)
  - Afficher le bouton « Retour » afin de revenir à la page « Administration 1 »
  - Gérer le côté Responsive

4.2 Changement

Lors de l'implémentation de notre projet et lors des séances avec le mandant et les superviseurs, nous avons réfléchi autrement et modifié certains points :

- Nous avons séparé en deux pages différentes la partie création et modification d'entités, car il y a un certain nombre de différences entre ces deux fonctions comme le remplissage automatique des champs lors de la modification de l'entité ou au niveau du Backend les appels sur des requêtes différentes (requêtes create ou edit).
- En ce qui concerne la partie de l'administration permettant de lier et de délier les entités, nous avons centralisé cette fonction sur une seule page, la page d'édition des entités. En effet, lorsqu'un administrateur crée une entité sur la page de création d'entités, il entre le nom, la catégorie et le contenu de l'entité. Une fois le bouton "Créer" cliqué, une redirection est effectuée vers la page d'édition de l'entité et c'est ici que l'administrateur peut lier et/ou délier l'entité en question, ce qui lui permet aussi de directement pouvoir modifier un champ de l'entité.
- Nous avons modifier l'emplacement des boutons sur les pages d'administration des entités afin de faciliter l'expérience utilisateur.

4.3 REST API

Nous avons créé deux API REST, une pour les entités et une pour les quizz.

4.3.1 API pour les entités

Voici la liste des URL et des méthodes HTTP utilisées:

- Obtenir une entité: /api/entity/id, méthode GET
- Créer une entité: /api/entity, méthode POST
- Editer une entité: /api/entity/id, méthode PUT
- Effacer une entité: /api/entity/id, méthode DELETE
- Obtenir toutes les entités, mais sans leurs entités liées: /api/entity, méthode GET

Pour requêter une entité l'URL utilisée est "http://localhost:8080/api/entity/id", dans notre projet, l'ID est le nom de l'entité. Ci-dessous se trouve un exemple d'une entité au format JSON.

```
{
    "name": "La Joconde",
    "category": "Art",
    "content": "data:/image...",
    "link": [
        "Le Louvre",
        "France",
        "PSG",
        "La tour Eiffel"
    ]
}
```

Le champ "name" de cette entité est son nom et aussi son ID. Le champ "category": contient sa catégorie, "content" contient soit du texte, soit une image encodée en Base64 et le champ "link" contient les noms (les IDs) des entités liées avec elle-même.

API côté Backend

La classe "AdminController.java" est le contrôleur pour la gestion des entités de notre application. Springboot nous a permis d'implémenter facilement ce contrôleur.

```
@RestController
@RequestMapping("/api/entity")
public class AdminController {

    @Autowired
    private EntityService entityService;

    @RequestMapping(method = RequestMethod.POST)
    public Entity create(@RequestBody Entity entity) {
        return entityService.create(entity);
    }

    @RequestMapping(value = "/{name}", method = RequestMethod.PUT)
    public Entity edit(@PathVariable("name") String name, @RequestBody Entity entity) {
        return entityService.edit(name, entity);
    }

    @RequestMapping(value = "/{name}", method = RequestMethod.DELETE)
    public String delete(@PathVariable("name") String name) {
        return entityService.delete(name);
    }

    @RequestMapping( method = RequestMethod.GET, produces = "application/json")
    public List<Entity> findAll(){
        return entityService.findAll();
    }

    @RequestMapping(value = "/{name}", method = RequestMethod.GET, produces =
         "application/json")
    public Entity findOne(@PathVariable("name") String name){
        return entityService.findOne(name);
    }
}
```

L'annotation "@RestControlleur" indique que cette classe est un contrôleur HTTP. L'annotation "@RequestMapping" en dehors de la classe indique l'URL de base de cette classe. Nous utilisons aussi cette annotation pour compléter l'URL de la classe avant chaque méthode si besoin. Son paramètre méthode permet de passer une énumération qui correspond aux méthodes du protocole HTTP. L'annotation "@Requestbody" permet de récupérer le contenu du body.

L'annotation @Autowired est très utile et nous permet d'instancier automatiquement un objet de type "EntityService". Nous n'avons donc pas besoin de faire dans chaque méthode "EntityService entityService = new EntityService()...".

Puis ces méthodes retournent nos objets en format JSON par défaut en se basant sur les "getters" de notre classe "Entity".

API côté Frontend

Voici le contenu de la classe "entity-service.ts":

```
@Injectable()
export class EntityService {

  constructor(private httpClient: HttpClient) {}

  public getEntity(id: string): Observable<Entity> {
    return this.httpClient.get<Entity>(`/api/entity/${id}`);
  }

  public getList(): Observable<Entity[]> {
    return this.httpClient.get<Entity[]>(`/api/entity`);
  }

  public create(e: Entity): Observable<Entity> {
    return this.httpClient.post<Entity>(`/api/entity`, e);
  }

  public edit(id: string, e: Entity): Observable<Entity> {
    return this.httpClient.put<Entity>(`/api/entity/${id}`, e);
  }

  public delete(id: string): Observable<Entity> {
    return this.httpClient.delete<Entity>(`/api/entity/${id}`);
  }
}
```

Sur le Frontend on fait l'inverse que l'implémentation sur le Backend. Le principe est le même. On doit indiquer l'URL et la méthode "HTTP" à utiliser.

L'annotation "@Injectable()" permet à Angular de savoir qu'une classe peut être utilisée avec l'injecteur de dépendances. Il suffit d'appeler les dépendances et Angular se charge de les instancier et de les injecter pour nous.

4.3.2 API pour les quizz

Voici la liste des URL et des méthodes HTTP utilisées:

- Obtenir la liste des quiz (nom, catégorie et tous les autres champs a null): /api/quiz, méthode GET
- Générer un quiz (donne l'entité question et les réponses possibles générées aléatoirement): /api/quiz/id, méthode GET
- Calculer le résultat: /api/quiz, méthode POST

Cette fois-ci l'URL de base est "api/quiz".

API côté Backend

```
@RestController
@RequestMapping("/api/quiz")
public class QuizController {

    @Autowired
    QuizService quizService;

    @RequestMapping()
    public List<Quiz> findAll(){
        return quizService.findAll();
    }

    @RequestMapping(value = "/{name}")
    public Quiz generate(@PathVariable("name") String name){
        return quizService.generate(name);
    }

    @RequestMapping(method = RequestMethod.POST)
    public Quiz check(@RequestBody Quiz quiz) {
        return quizService.check(quiz);
    }
}
```

API côté Frontend

```
@Injectable()
export class QuizService {

  constructor(private httpClient: HttpClient) {}

  public findAll(): Observable<Quiz[]> {
    return this.httpClient.get<Quiz[]>('/api/quiz');
  }

  public get(name: string): Observable<Quiz> {
    return this.httpClient.get<Quiz>('/api/quiz/${name}');
  }

  public check(q: Quiz): Observable<Quiz> {
    return this.httpClient.post<Quiz>('/api/quiz', q);
  }
}
```

4.4 Composants Angular

Nous allons décrire le composant "admin-view" qui représente la page d'administration principale de l'application. Chaque composant possède son propre fichier CSS, HTML et TS (pour le Typescript). Dans cette exemple nous avons donc un fichier "admin-view.component.ts", "admin-view.component.css" et "admin-view.component.html".

Voici une partie du fichier "admin-view.component.ts" sans les imports:

```
@Component({
  selector: 'app-admin-view',
  templateUrl: './admin-view.component.html',
  styleUrls: ['./admin-view.component.css'],
  encapsulation: ViewEncapsulation.None
})
export class AdminViewComponent implements OnInit {

  public initialEntityList: Entity[];
  public entityList: Entity[];
  public categories: string[];

  public filterCategory: string;

  constructor(private activatedRoute:ActivatedRoute, private entityService:
      EntityService) {
    this.filterCategory = `All`;
  }

  ngOnInit() {
    this.entityService.getList().subscribe((list: Entity[]) => {
      this.initialEntityList = list;

      let cat = list.map((e:Entity) => e.category);
      cat.push('All');
      cat.sort();

      this.categories = cat.filter(function(item, pos) {
        return cat.indexOf(item) == pos;
      });
      this.applyFilter();
    });
  }

  public applyFilter():void{
    if(this.filterCategory === 'All'){
      this.entityList = this.initialEntityList;
    }
    else {
      this.entityList = this.initialEntityList
        .filter((e: Entity) => e.category === this.filterCategory);
    }
  }
```

Voici une partie du fichier "admin-view.component.html":

```
<select (change)="applyFilter()" [(ngModel)]="filterCategory">
    <option *ngFor="let cat of categories" value="{{cat}}">{{cat}}</option>
</select>
```

Dans l'exemple ci-dessus, nous pouvons observer plusieurs cas intéressants d'Angular. Le (change)="applyFilter()" est un listeneur qui va appeler la méthode "applyFilter()" définit dans le fichier .ts lorsqu'il y a un changement de la sélection sur la balise <select...>.

L'élément [ngModel] permet de binder la valeur du "filterCategory" définit dans le fichier .ts.

Le *ngFor correspond a une boucle foreach qui liste les catégories définies dans la liste des catégories du fichier .ts. Je trouve que la façon de faire Angular est beaucoup plus élégante qu'avec des sites faits en PHP par exemple, car ici dans le fichier HTML, il n'y a que des balises.

Une variable peut être affichée directement dans le HTML en l'appelant de la sorte cat. Dès que la valeur de cette variable change, elle va changer partout sans avoir besoin de rafraîchir la page.

Ce qui est aussi très intéressant c'est qu'on peut instancier un composant en utilisant son "selector" dans notre cas 'app-admin-view' dans le fichier HTML d'un autre composant en utilisant la balise <app-admin-view></app-admin-view>.

Enfin, avec Angular, si l'on veut créer une nouvelle page on doit créer un "router".

4.5 Constantes

Certaines constantes basiques pour notre application ont été définies.

A noter que la définition des constantes est implémentée dans le fichier du Backend:
"main/java/ch.heiafr.virtual_memory/Constants".

```
// the total of possible answers(bad or good),
// should not be bigger than the total of entities in the database and NB_OF_GOOD_ANSWERS
public final static int  NB_OF_ANSWERS           = 8;

// the total of good answers, should not be bigger than NB_OF_ANSwERS
public final static int  NB_OF_GOOD_ANSWERS      = 2;

// the minimum of relationship that an entity has to have to become a quiz,
// should not be smaller than NB_OF_GOOD_ANSWERS
public final static int  MINIMUM_OF_RELATIONSHIPS = NB_OF_GOOD_ANSWERS;
```

La première constante NB_OF_ANSWER permet de définir le nombre de réponses (bonnes et/ou mauvaises) à afficher dans le quizz, cette constante ne doit pas être supérieur aux nombres d'entités dans le quizz.

La deuxième constante NB_OF_GOOD_ANSWER permet de définir le nombre de bonnes réponses à afficher dans le quizz, cette constante ne doit pas être supérieur au nombre de réponses (bonnes et/ou mauvaises) à afficher dans le quizz.

La troisième constante MINIMUM_OF_RELATIONSHIPS permet de définir le nombre de relations qu'une entité doit avoir afin de pouvoir devenir une question, cette constante ne doit pas être supérieur au nombre de réponses bonnes à afficher dans le quizz.

4.6 Requêtes

Toutes les requêtes permettant l'interaction avec la base de données sont définies et utilisées dans des fonctions.

A noter que ces fonctions interagissant avec la basse de données sont implémentées dans le fichier du Backend: "main/java/ch.heiafr.virtual_memory/repository/QuizRepository".

4.6.1 Retourner les entités

Cette fonction "getAllEntities() permet de retourner une liste de toutes les entités de la base de données:

```
public List<Entity> getAllEntities() {
    List<Entity> entityList = new ArrayList<>();
    try (Session session = DRIVER.session()) {
        StatementResult result = session.run("MATCH (n) RETURN n.name AS name,
             n.category AS cat, n.content AS content ORDER BY name",
                parameters());
        while (result.hasNext()) {
            Record record = result.next();
            String name = record.get("name").asString();
            String category = record.get("cat").asString();
            String content = record.get("content").asString();
            entityList.add(new Entity(name, category, content));
        }
    }
    return entityList;
}
```

4.6.2 Retourner les entités liées

Cette fonction "entitiesLinkedWith(String name) permet de retourner une liste de toutes les entités de la base de données étant liées à une entité spécifique passée en paramètre:

```
public List<String> entitiesLinkedWith(String name) {
    List<String> linkedList = new ArrayList<>();
    try (Session session = DRIVER.session()) {
        StatementResult result = session.run("MATCH (n{name:'" + name +
             "'})-[:KNOWS]->(m) RETURN m.name as name", parameters());
        while (result.hasNext()) {
            Record record = result.next();
            linkedList.add(record.get("name").asString());
        }
    }
    return linkedList;
}
```

4.6.3 Retourner les entités non liées

Cette fonction "entitiesUnlinkedWith(String name) permet de retourner une liste de toutes les entités de la base de données n'étant pas liées à une entité spécifique passée en paramètre:

```
public List<String> entitiesUnlinkedWith(String name) {
    List<String> unlinkedList = new ArrayList<>();
    try (Session session = DRIVER.session()) {
        StatementResult result = session.run("MATCH (n{name:'" + name + "'}) MATCH (m)
            WHERE (NOT (n)-[:KNOWS]-(m)) RETURN m.name as name", parameters());
        while (result.hasNext()) {
            Record record = result.next();
            if (!record.get("name").asString().equals(name)) {
                unlinkedList.add(record.get("name").asString());
            }
        }
    }
    return unlinkedList;
}
```

4.6.4 Créer une entité

Cette fonction "create(Entity e) permet de créer une entité si l'entité n'existe pas déjà, le contenu de l'entité est passé en paramètre:

```
public Entity create(Entity e) {
    try (Session session = DRIVER.session()) {
        try (Transaction tx = session.beginTransaction()) {
            tx.run("CREATE (n:node{ name: '" + e.getName() + "', category: '" +
                e.getCategory() + "', content: '" + e.getContent() + "'})",
                parameters());
            tx.success();
        }
    }
    return getAnEntity(e.getName());
}
```

4.6.5 Editer une entité

Cette fonction "edit(String name, Entity e) permet de modifier une entité, le nom de l'entité est passé en paramètre afin de savoir quelle entité il faut modifier et le nouveau contenu de l'entité est passé en paramètre:

```
public Entity edit(String name, Entity e) {
    try (Session session = DRIVER.session()) {
        try (Transaction tx = session.beginTransaction()) {
            tx.run("MATCH (n:node{ name: '" + name +"'})"
                   + "SET n.name = '" + e.getName() + "' "
                   + "SET n.category = '" + e.getCategory() + "' "
                   + "SET n.content = '" + e.getContent() + "'", parameters());
            tx.success();
        }
    }
    return getAnEntity(e.getName());
}
```

4.6.6 Supprimer une entité

Cette fonction "delete(String name) permet de supprimer une entité de la base de données, le nom de l'entité est passé en paramètre afin de savoir quelle entité il faut supprimer:

```
public String delete(String name) {
    try (Session session = DRIVER.session()) {
        try (Transaction tx = session.beginTransaction()) {
            tx.run("MATCH (n{name:'" + name + "'}) DETACH DELETE n", parameters());
            tx.success();
        }
    }
    return null;
}
```

4.6.7 Lier deux entités

Cette fonction "link(String name1, String name2) permet de lier deux entités entre elles, les noms des deux entités sont passés en paramètre afin de savoir quelles entités il faut lier:

```
public void link(String name1, String name2) {
    try (Session session = DRIVER.session()) {
        try (Transaction tx = session.beginTransaction()) {
            tx.run("MATCH (a{name:'" + name1 + "'})" + " MATCH (b{name:'" + name2 + "'})"
                +
                          " MERGE (a)-[:KNOWS]->(b)" + " MERGE (b)-[:KNOWS]->(a)"
                  , parameters());
            tx.success();
        }
    }
}
```

4.6.8 Délier deux entités

Cette fonction "unlink(Entity entity) permet de délier des entités entre elles, les noms des entités sont passés en paramètre afin de savoir quelles entités il faut délier:

```
public void unlink(Entity entity) {
    try (Session session = DRIVER.session()) {
        try (Transaction tx = session.beginTransaction()) {
            tx.run("MATCH (n{name:'" + entity.getName() + "'})-[r:KNOWS]-()" +
                   "DELETE r", parameters());
            tx.success();
        }
    }
}
```

4.6.9 Retourner les entités pour créer le quizz

Cette fonction "findAll() permet de retourner une liste d'entité qui permet de créer le quizz, dans la fonction une constante "MINIMUM_OF_RELATIONSHIPS" permet de définir que l'on veut les entités qui possèdent au minimum une relation vers une autre entité:

```
public List<Quiz> findAll() {
    List<Quiz> quizList = new ArrayList<>();
    try (Session session = DRIVER.session()) {
        StatementResult result = session.run("MATCH (n)-->()" +
                " WITH n, COUNT(*) AS rel_cnt" +
                " where rel_cnt >= " + MINIMUM_OF_RELATIONSHIPS +
                " return n.name as name, n.category as cat ORDER BY name", parameters());
        while (result.hasNext()) {
            Record record = result.next();
            String name = record.get("name").asString();
            String category = record.get("cat").asString();
            Entity e = new Entity(name, category, null);
            Quiz q = new Quiz(e, null, null, null);
            quizList.add(q);
        }
    }
    return quizList;
}
```

4.7 Insertion dans la base de données

4.7.1 Caractères spéciaux

En réalisant divers tests d'insertion de contenus dans la base de données via les interfaces d'ajout et de modification d'entités, nous avons pu voir que 2 caractères en particulier posaient problème et créaient une erreur dans la base de données, ces caractères sont ' et ". En effet, les requêtes qui interagissent avec la base de données se composent comme ceci:

```
CREATE ({ name: 'Paris', content: 'La ville des lumieres'})
```

Si nous ajoutons un de ces deux caractères dans une requête, nous pouvons voir qu'il y a un conflit avec les guillemets qui définissent le contenu d'un champ. Nous avons donc empêché l'écriture de ces caractères dans les divers champs définissant une entité, qui sont les champs "name", "category" et "content" avec une méthode qui est appelée dans les balises "input" et "textarea".

Par exemple dans le champ définissant le nom de l'entité, nous appelons la méthode "omitSpecialChar()" quand un caractère est pressé au clavier avec ce caractère en paramètre:

```
<input type="name" class="form-control" id="name" required minlength="1"
     (keypress)="omitSpecialChar($event)" [(ngModel)]="entity.name"
     (keyup)="checkField()" name="name">
```

Dans le fichier ".ts" en relation avec le fichier ".html" défini précédemment, nous avons réalisé la méthode empêchant ces deux caractères:

```
omitSpecialChar(event) {
    var k;
    k = event.charCode;
    return ((k!=34) && (k!=39));
  }
```

Les codes ASCII 34 et 39 représentent les caractères ' et ".

Lors de la création d'une entité, nous avons aussi utilisé la fonction "trim()" sur le nom, la catégorie et le contenu de l'entité:

```
this.entity.name = this.entity.name.trim();
this.entity.content = this.entity.content.trim();
this.entity.category = this.entity.category.trim();
```

Cette fonction permet de supprimer les caractères en début et en fin d'une chaîne de caractères comme les caractères invisibles, comme les espaces, les tabulations ou les retours à la ligne. Nous avons décidé d'utiliser cette fonction, car nous avons pu constater que lorsque nous insérions un espace en fin d'une chaîne de caractères, la base de données ne le supportait pas très bien.

4.7.2 Taille minimum

Lors de la création et de la modification d'une entité, nous avons empêché le fait que l'administrateur puisse insérer dans la base de données des contenus vides pour le nom, la catégorie et le contenu, car, là aussi, un conflit avec la base de données était détecté. Nous avons fixé une limite de 1 caractère minimum et tant que les 3 champs ne contiennent pas un minimum de 1 caractère, les boutons "Ajouter" ou "Valider" ne s'activent pas.

Dans le fichier ".html" suivant, nous pouvons voir qu'un minimum de 1 caractère est requis:

```
<input type="name" class="form-control" id="name" required minlength="1"
    (keypress)="omitSpecialChar($event)" [(ngModel)]="entity.name"
    (keyup)="checkField()" name="name">
```

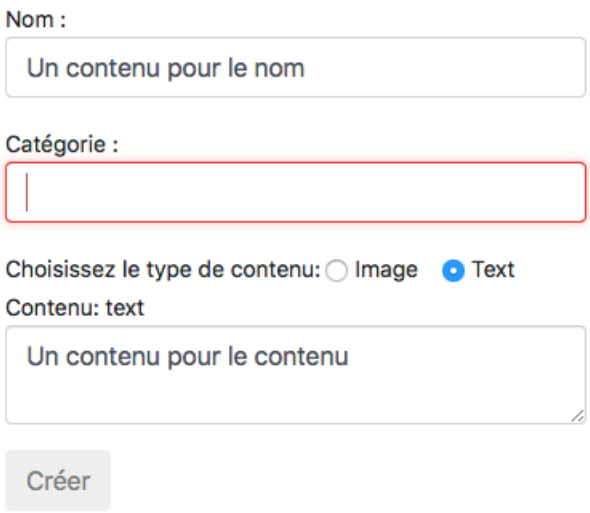


Figure 52 – Aperçu lorsqu'un champ est vide, le bouton n'est pas activé

4.7.3 Images

Nous avons pu voir qu'une entité peut être créée soit avec un contenu textuel, soit avec un contenu sous forme d'image. L'image s'affiche dans le quizz avec une taille 150px sur 150px. Se référer au code CSS du côté du jeu lors de l'affichage des entités et du côté de l'administration pour la création et l'édition des entités:

```
.img-size{
  width: 150px;
  height: 150px;
}

<div *ngIf="isAnImage(question?.content)" class="div-quizz-question-border">
<img class="img-size" src="{{question?.content}}">
</div>
```

Nous avons fait des tests avec plusieurs images de tailles différentes:

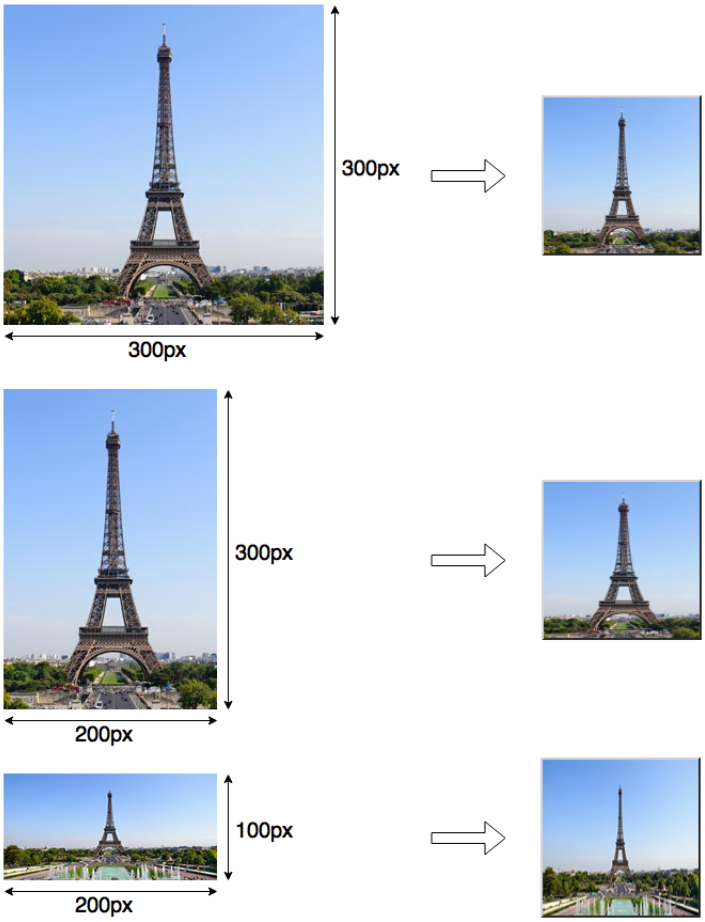


Figure 53 – Aperçu du recadrage automatique des images

Il est donc préférable de choisir des images carrées pour éviter le recadrage déformant l'image et d'opter pour des images pas trop volumineuses afin de ne pas trop surcharger la base de données vu que les images sont converties en Base64.

4.8 Générer le quizz

La génération de quizz est basée sur des constantes définies et sur des valeurs aléatoires. Le joueur sélectionne un quizz dans la liste des entités. L'application génère un quizz en fonction de cette entité. Le nombre de bonnes réponses à afficher est défini par une constante et ces entités sont tirées aléatoirement dans la liste des entités liées à l'entité sélectionnée. Les mauvaises réponses sont tirées aléatoirement dans la liste des entités non liées à l'entité sélectionnée. Le nombre de mauvaises réponses est défini par la différence du nombre de réponses totales bonnes ou mauvaises, définies via une constante. Le nombre de bonnes réponses à afficher est aussi défini par une constante.

A noter que la définition des constantes est implémentée dans le fichier du Backend:
"main/java/ch.heiafr.virtual_memory/Constants".

A noter que la génération du quizz est implémentée dans le fichier du Backend:
"main/java/ch.heiafr.virtual_memory/repository/QuizRepository".

Nous avons défini les constantes suivantes:

- Le nombre de réponses, bonnes ou mauvaises, à afficher dans le quizz: NB_OF_ANSWERS qui est de 8
- Le nombre de bonnes réponses à afficher dans le quizz: NB_OF_GOOD_ANSWERS qui est de 2
- Le nombre minimum de relations que doit avoir une entité pour être jouée en tant que quizz: MINIMUM_OF_RELATIONSHIPS qui est égal au nombre de bonnes réponses à afficher dans le quizz, donc 2

Bien entendu, si ces chiffres ne sont pas respectés, s'ils sont inférieurs, le quizz ne peut pas être joué, nous reviendrons sur ce point.

Les conditions afin qu'un quizz puisse être généré sont lui suivantes:

```
// un quizz ne peut pas etre joue...

// si le nombre de bonnes reponses a afficher est plus que le nombre de reponses bonnes
    et/ou mauvaises totales a afficher
if(NB_OF_GOOD_ANSWERS > NB_OF_ANSWERS){
    return null;
}

// si le nombre de mauvaises reponses tires moins le nombre de bonnes reponses a
    afficher est plus petit que le nombre de reponses bonnes et/ou mauvaises totales a
    afficher
if(badAnswers.size() - NB_OF_GOOD_ANSWERS < NB_OF_ANSWERS){
    return null;
}

// si le nombre de bonnes reponses tires est plus petit que le nombre de bonnes
    reponses a afficher
if(goodAnswers.size() < NB_OF_GOOD_ANSWERS){
    return null;
}
```

Ces conditions signifient que si le joueur commence avec une base de données vides, il doit impérativement la peuplée avec de pouvoir jouer.

Lorsqu'un quizz ne peut pas être joué, car les constantes définies ne sont pas respectées, l'application informe ce dernier:

Figure 54 – Aperçu lorsque la base de données ne contient pas d'entités ou lorsque les constantes ne sont pas respectées

Il est clair que les constantes définies précédemment ne sont modifiables que dans le code. L'idéal serait que le joueur puisse choisir lui même le nombre de réponses à afficher par quizz et que le nombre de bonnes ou de mauvaises réponses soit aussi aléatoire, mais nous n'avons pas implémenté ces fonctions.

L'application informe aussi le joueur, lorsque la base de données est vide que ce dernier doit créer des entités, avec un message du même type que celui que nous avons pu observer précédemment.

4.9 Affichage du contenu

Le contenu d'une image peut soit être du texte ou une image. Lors de l'affichage d'une entité soit dans la partie jeu, soit dans la partie administration, l'application doit savoir si elle doit afficher un texte ou une image. En effet, si tout est interprété comme du texte avec une balise " ... , pour les entités possédant une image, l'application affichera le code de l'image en format Base64. Nous devons donc tester si le contenu doit être interprété comme du texte (balise " ... ") ou comme une image (balise "<img> ... </img>").

Nous avons pu observer que lorsque nous convertissons une image en format Base64, le contenu ressemble à "...", les 10 premiers caractères sont obligatoirement "data:image". Lorsque nous devons afficher une image, nous utilisons un Boolean "isAnImage" qui, s'il est à "false" affiche du contenu texte avec la balise " ... " ou s'il est à "true" affiche du contenu sous forme d'image avec la balise "<img> ... </img>":

```
<div *ngIf="currentQuiz != null" class="row div-quizz-question">
    <div *ngIf="isAnImage(question?.content)" class="div-quizz-question-border">
        <img class="img-size" src="{{question?.content}}">
    </div>
    <div *ngIf="!isAnImage(question?.content)" class="div-quizz-question-border">
        <p>{{question?.content}}</p>
    </div>
</div>
```

Nous avons défini une méthode, dans le fichier ".ts" en relation avec le fichier HTML affichant le contenu, qui est appelée lors de l'affichage de la page:

```
isAnImage(str:string):boolean{
    if(str==undefined){return false;}
    let isAnImage:boolean;
    if(str.substring(0, 10) == 'data:image'){
        isAnImage = true;
    }
    else{
        isAnImage = false;
    }
    return isAnImage;
}
```

Cette méthode test si le contenu est textuel ou sous forme d'image au format Base64, nous pouvons voir que si le contenu d'une entité commence par "data:image" et si c'est le cas, le Boolean "isAnImage" est mis à "true" et le contenu est interprété comme une image avec la balise adéquate, et sinon, le Boolean est mis à "false" et le texte s'affichera dans sa balise.

4.10 Calcul du score

Pour le calcul du score, nous sommes partis dans un premier temps vers un calcul assez complexe, mais qui, à force de tester ce score, ne nous convenait pas. Afin de remédier à cela, nous sommes ensuite partis vers une seconde solution qui nous convenait plus et reflétait plus le calcul de score classique. Mais cette solution était au final incohérente. Ces deux solutions calculaient le score en pourcent, mais nous avons pu voir que de faire un calcul de score de cette manière est assez complexe, car au final, pour un score en pourcent, nous ne savions pas trop sur quoi nous baser.

C'est pourquoi, nous sommes partis sur une troisième solution, retournant tout simplement le nombre d'erreurs.

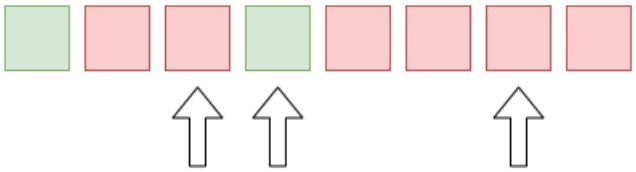
A noter que la réalisation de ce calcul est implémentée dans le fichier du Backend:
"main/java/ch.heiafr.virtual_memory/repository/QuizRepository".

4.10.1 Première solution

Afin de pouvoir calculer le score du quizz, nous avions besoin de plusieurs informations, nous allons vous illustrer ce calcul avec un exemple.

Nous avons un quizz contenant 8 réponses , dont 2 sont bonnes et 5 sont mauvaises:

Le joueur sélectionne 1 bonne réponse et 2 mauvaises réponses:



Il y a donc 3 erreurs de la part du joueur, ce qui lui vaut un score de 62,5%.

Pour calculer ce score, nous avons donc besoin de:

- Le nombre de réponses totales, dans notre exemple 8
- Le nombre de bonnes réponses non sélectionnées, dans notre exemple 1
- Le nombre de mauvaises réponses sélectionnées, dans notre exemple 2

Le calcul est le suivant 100 - (100 / 8 * (1 + 2)) = 62,5%.

Dans le code suivant, les variables sont définies comme ceci:

- Le nombre de réponses totales: NB_OF_ANSWERS
- Le nombre de bonnes réponses non sélectionnées: nb_good_unselected
- Le nombre de mauvaises réponses sélectionnées: nb_bad_selected

Ces variables sont calculées avec les différentes requêtes en particulier celles qui permettent de retourner les entités liées ou non liées des entités sélectionnées ou non sélectionnées dans le quizz.

```
    public Quiz check(Quiz quiz){

        // recuperer la liste des bonnes reponses du quizz (liees a la questions)
        List<String> goodAnswers =
            entityRepository.entitiesLinkedWith(quiz.getQuestion().getName());

        // recuperer la liste des mauvaises reponses du quizz (non liees a la questions)
        List<String> badAnswers  =
            entityRepository.entitiesUnlinkedWith(quiz.getQuestion().getName());

        // recuperer la liste des questions du quizz
        List quizAnswers = quiz.getAnswers();

        float score = 100;
        float percentage = (100 / NB_OF_ANSWERS);
        int nb_good_selected = 0;
        int nb_bad_selected = 0;
        int nb_good_unselected = 0;

        // pour chaque reponse selectionnee
        for(String select:quiz.getSelected()){
            // calculer le nombre de bonnes reponses selectionnees
            if(goodAnswers.contains(select)){
                nb_good_selected++;
            }
            // calculer le nombre de mauvaises reponses selectionnees
            if(badAnswers.contains(select)){
                nb_bad_selected++;
            }
        }
    // calculer le nombre de bonnes reponses non selectionnees
    nb_good_unselected=NB_OF_GOOD_ANSWERS-nb_good_selected;

    // calculer et retourner le score en pourcent selon le calcul definit precedemment
    score -= (percentage*(nb_good_unselected + nb_bad_selected));
    return new Quiz(null, null, null, Float.toString(score));
}
```

Avec cet algorithme, nous pouvons calculer le score, avec le calcul comme nous l'avons décrit dans l'exemple précédent.

Avec les tests sur le quizz et le score qui reflétait ces différents tests, nous n'étions pas assez satisfaits, exemple tout simple, lorsque le joueur clique sur le bouton valider sans sélectionner aucune réponse, il obtenait déjà un score de x% en fonction des réponses fausses non sélectionnées que l'algorithme estime comme correctes, ce qui ne reflète pas vraiment un quizz classique, le score devrait être de 0%.

4.10.2 Deuxième solution

Pour cette seconde solution, nous sommes partis sur un algorithme plus simple qui se base juste sur les réponses correctes sélectionnées et non sélectionnées et non sur les réponses fausses sélectionnées et non sélectionnées.

Voici un exemple concret, sur un quizz de x réponse(s) fausse(s), contenant 4 bonnes réponses:

- si le joueur coche x réponse(s) fausse(s) et 0 réponse correcte, son score sera de 0%
- si le joueur coche x réponse(s) fausse(s) et 1 réponse correcte, son score sera de 25%
- si le joueur coche x réponse(s) fausse(s) et 2 réponses correctes, son score sera de 50%
- si le joueur coche x réponse(s) fausse(s) et 3 réponses correctes, son score sera de 75%
- si le joueur coche x réponse(s) fausse(s) et 4 réponses correctes, son score sera de 100%

Pour calculer ce score avec cette manière de faire, nous avons donc besoin, en nous basant sur l'exemple précédent de:

- Le nombre de réponses totales, dans notre exemple 8
- Le nombre de bonnes réponses totales, dans notre exemple 2
- Le nombre de bonnes réponses sélectionnées, dans notre exemple 1

Le calcul est le suivant 100 / 2 * 1 = 50%.

Avec ce calcul, si le joueur ne sélectionne aucune bonne réponse, son score sera de 0% et non x%, ce qui correspond mieux à un calcul de score classique d'un quizz.

Dans le code suivant, les variables sont définies comme ceci:

- Le nombre de réponses totales: NB_OF_ANSWERS
- Le nombre de bonnes réponses: nb_good_answer
- Le nombre de bonnes réponses sélectionnées: nb_good_answer_selected

Ces variables sont calculées avec les différentes requêtes en particulier celles qui permettent de retourner les entités liées ou non liées des entités sélectionnées ou non sélectionnées dans le quizz.

```
        public Quiz check(Quiz quiz){

            // recuperer la liste des bonnes reponses du quizz (liees a la questions
            List<String> goodAnswers =
            entityRepository.entitiesLinkedWith(quiz.getQuestion().getName());

            // recuperer la liste des mauvaises reponses du quizz (non liees a la questions)
            List<String> badAnswers  =
                entityRepository.entitiesUnlinkedWith(quiz.getQuestion().getName());

            // recuperer la liste des questions du quizz
            List quizAnswers = quiz.getAnswers();

            int score;
            int maxScore = 100;
            int nb_good_selected = 0;

            // pour chaque reponse selectionnee
            for(String select:quiz.getSelected()){
                // calculer le nombre de bonnes reponses selectionnees
                if(goodAnswers.contains(select)){
                    nb_good_selected++;
                }
            }

            // si le nombre de bonnes reponses du quizz est egal au nombre de bonnes reponses
                selectionnes
            if(NB_OF_GOOD_ANSWERS==nb_good_selected){
                // le score est de 100\%
                score = maxScore;
            }else{
                // calculer le score en pourcent selon le calcul definit precedemment
                score = maxScore / NB_OF_GOOD_ANSWERS * nb_good_selected;
            }

            // retourner le score
            return new Quiz(null, null, null, Integer.toString(score));
        }
    }
```

Malheureusement, nous n'y avions pas pensé tout de suite, et à force de faire des tests, nous avons pu remarquer une incohérence. Comme nous l'avons défini précédemment, sur un quizz de x réponse(s) fausse(s), contenant 4 bonnes réponses, si le joueur coche x réponse(s) fausse(s) et 4 réponses correctes, son score sera de 100%, mais cette manière de faire n'est pas correcte. Le joueur ne peut pas avoir un score de 100% s'il a coché x réponse(s) fausse(s).

4.10.3 Troisième solution

Nous avons pu voir que le calcul du score en pourcent n'était pas facile, nous ne savions pas vraiment sur quoi nous baser et il est vrai qu'un score en pourcent n'est peut-être pas assez explicite pour le jouer. Nous avons donc décidé de partir sur un score qui affiche simplement le nombre de mauvaises réponses.

Pour ce calcul, nous additionnons le nombre de réponses correctes non cochées et le nombre de réponses fausses cochées, ce qui nous donne le nombre d'erreurs qui sera notre score. Avec cette manière de faire, nous avons donc besoin, en nous basant sur l'exemple précédent de:

- Le nombre de réponses totales, dans notre exemple 8
- Le nombre de bonnes réponses sélectionnées, dans notre exemple 1
- Le nombre de bonnes mauvaises sélectionnées, dans notre exemple 2
- Le nombre de bonnes réponses non sélectionnées, dans notre exemple 1

Le calcul est le suivant le nombre de bonnes réponses non sélectionnées + le nombre de bonnes mauvaises sélectionnées: 1 + 2 = 3 erreurs. A noter que le nombre de bonnes réponses non sélectionnées est calculé à partir du nombre de réponses totales moins le nombre de bonnes réponses sélectionnées.

Dans le code suivant, les variables sont définies comme ceci:

- Le nombre de réponses totales: NB_OF_GOOD_ANSWERS
- Le nombre de bonnes réponses sélectionnées: nb_good_selected
- Le nombre de bonnes mauvaises sélectionnées: nb_bad_selected
- Le nombre de bonnes réponses non sélectionnées: nb_good_unselected

C'est variables sont calculées avec les différentes requêtes en particulier celles qui permettent de retourner les entités liées ou non liées des entités sélectionnées ou non sélectionnées dans le quizz. A noter que cette fois, le score est transformé en String à partir d'un Interger, car nous ne voulons pas de virgule dans le nombre d'erreurs.

```
    public Quiz check(Quiz quiz){

        // recuperer la liste des bonnes reponses du quizz (liees a la questions)
        List<String> goodAnswers =
            entityRepository.entitiesLinkedWith(quiz.getQuestion().getName());

        // recuperer la liste des mauvaises reponses du quizz (non liees a la questions)
        List<String> badAnswers  =
            entityRepository.entitiesUnlinkedWith(quiz.getQuestion().getName());

        int score;
        int nb_good_selected = 0;
        int nb_bad_selected = 0;
        int nb_good_unselected = 0;

        // pour chaque reponse selectionnee
        for(String select:quiz.getSelected()){
            // calculer le nombre de bonnes reponses selectionnees
            if(goodAnswers.contains(select)){
                nb_good_selected++;
            }
            // calculer le nombre de mauvaises reponses selectionnees
            if(badAnswers.contains(select)){
                nb_bad_selected++;
            }
        }
    // calculer le nombre de bonnes reponses non selectionnees
    nb_good_unselected=NB_OF_GOOD_ANSWERS-nb_good_selected;

    // calculer et retourner le score selon le calcul definit precedemment
    score = nb_good_unselected + nb_bad_selected;
    return new Quiz(null, null, null, Integer.toString(score));
}
```

Cette manière de retourner le score est pour nous la plus optimale, plus compréhensible et plus agréable pour le joueur. Cette méthode à été testée et cette fois-ci, elle nous convient parfaitement.

4.11 État de l'implémentation

Nous avons réalisé un tableau avec chacune des tâches définies au début de l'implémentation en référence avec le cahier des charges afin d'en indiquer l'état d'avancement. Cette grille d'évaluation montre l'état au 31 janvier 2018.

Page	Tâches	Etat
Accueil	Créer la page « Accueil »	
	Afficher le bouton « Administrer » afin d'accéder à la page « Choix du quizz »	
	Afficher le bouton « Jouer » afin d'accéder à la page « Choix du quizz »	
	Gérer le côté Responsive	

Sur la page "Accueil", toutes les tâches définies en relation avec le cahier des charges sont fonctionnelles.

Page	Tâches	Etat
Choix du quizz	Lister les catégories (filtre)	
	Lister les contenus (filtre)	
	Lister les quizz (entités)	
	Lister les quizz (entités) en fonction du filtre des catégories	
	Lister les quizz (entités) en fonction du filtre des contenus	
	Sélectionner un quizz (entité) afin d'accéder à la page « Quizz »	
	Afficher le bouton « Retour » afin de revenir à la page « Accueil »	
	Gérer le côté Responsive	

Sur la page "Choix du quizz", toutes les tâches définies en relation avec le cahier des charges sont fonctionnelles sauf celle du filtre en fonction des contenus.

Page	Tâches	Etat
Quizz	Afficher le quizz (entités) avec des bonnes et des mauvaises réponses	
	Sélectionner les réponses (entités)	
	Afficher le bouton « Valider » afin de vérifier les réponses (entités) cochées	
	Afficher le résultat du quizz	
	Afficher le bouton « Retour » afin de revenir à la page « Choix du quizz »	
	Gérer le côté Responsive	

Sur la page "Quizz", toutes les tâches définies en relation avec le cahier des charges sont fonctionnelles sauf celle du filtre en fonction des contenus.

Page	Tâches	Etat
Admin	Lister les catégories (filtre)	
	Lister les contenus (filtre)	
	Lister les quizz (entités)	
	Lister les quizz (entités) en fonction du filtre des catégories	
	Lister les quizz (entités) en fonction du filtre des contenus	
	Afficher les boutons « Supprimer » dans la liste des quizz (entités) afin de supprimer le quizz sélectionné	
	Afficher un message d'alerte permettant de confirmer la suppression d'une quizz (entité)	
	Afficher les boutons « Editer » dans la liste des quizz (entités) afin de modifier le quizz sélectionné	
	Afficher le bouton « Ajouter » afin d'accéder à la page « Administration 2a »	
	Afficher le bouton « Retour » afin de revenir à la page « Accueil »	
	Gérer le côté Responsive	

Sur la page "Admin", toutes les tâches définies en relation avec le cahier des charges sont fonctionnelles sauf celle du filtre en fonction des contenus.

Page	Tâches	Etat
Create	Afficher le champ « Nom » vide	
	Afficher le champ « Catégorie » vide	
	Afficher le champ « Contenu » vide	
	Afficher le téléchargement d'image	
	Afficher les boutons « Créer » afin d'enregistrer les modifications	
	Afficher le bouton « Retour » afin de revenir à la page « Administration 1 »	
	Gérer le côté Responsive	

Sur la page "Create", toutes les tâches définies en relation avec le cahier des charges sont fonctionnelles.

Page	Tâches	Etat
Edit	Afficher le champ « Nom » avec le nom du quizz (entité)	
	Afficher le champ « Catégorie » avec la catégorie du quizz (entité)	
	Afficher le champ « Contenu » avec le contenu du quizz (entité)	
	Afficher le téléchargement d'image avec l'image du quizz (entité)	
	Lister les catégories (filtre)	
	Lister les contenus (filtre)	
	Lister les quizz (entités) liés (check box activée)	
	Lister les quizz (entités) non liés (check box désactivée)	
	Lister les quizz (entités) liés (check box activée) en fonction du filtre des catégories	
	Lister les quizz (entités) non liés (check box désactivée) en fonction du filtre des catégories	
	Lister les quizz (entités) liés (check box activée) en fonction du filtre des contenus	
	Lister les quizz (entités) non liés (check box désactivée) en fonction du filtre des contenus	
	Activé check box pour lier une entité	
	Désactivé check box pour délier une entité	
	Afficher les boutons « Valider » afin d'enregistrer les modifications	
	Afficher le bouton « Retour » afin de revenir à la page « Administration 1 »	
	Gérer le côté Responsive	

Sur la page "Edit", toutes les tâches définies en relation avec le cahier des charges sont fonctionnelles sauf celle du filtre en fonction des contenus.

4.11.1 Remarques

Un point en particulier est à éclaircir en ce qui concerne la grille précédemment présentée en référence au cahier des charges.

Le filtre en fonction du contenu de l'entité n'a pas été réalisé. Ce filtre devait être appliqué à chacune des listes d'entités dans le but de filtrer les entités en fonction du contenu, soit les entités ayant un contenu textuel, soit les entités ayant un contenu représentant une image (Base64).

Nous n'avons pas implémenté cette partie, car nous avons mis des priorités sur d'autres points comme le téléchargement d'images et la gestion du convertissement (Base64) afin de créer des entités contenant des images de manière simple pour l'administrateur. Pour rappel, la gestion des contenus possédant une image lors de la création d'entités n'était pas gérée et l'administrateur devait convertir l'image sur un site Web externe et copier le code dans le contenu de l'entité, ce qui n'était pas pratique. Nous nous sommes donc concentrés sur ce point pour les dernières heures de travail du projet.

Ce filtre de contenu est le seul point défini dans le cahier des charges n'ayant pas été réalisé. Tous les points importants afin d'administrer et de jouer au quizz sont implémentés et fonctionnels.

4.12 Dernières modifications

Sur les dernières heures de travail sur le projet, nous avons tout mis en oeuvre pour implémenter deux dernières fonctionnalités.

4.12.1 Visualisation des erreurs

Nous avons mis une priorité sur un point qui, après discussion avec nos superviseurs, nous semblait important. En effet, lorsque le joueur validait son quizz, il pouvait voir le nombre d'erreurs qu'il avait fait, mais il ne pouvait pas voir qu'elles étaient les bonnes et mauvaises réponses, où il avait fait ces erreurs.

Le petit plus que nous avons apporté à notre Virtual Memory dans les dernières heures du projet est justement que l'utilisateur puisse avoir une vue des bonnes ou mauvaises réponses. Nous pouvons voir le résultat:

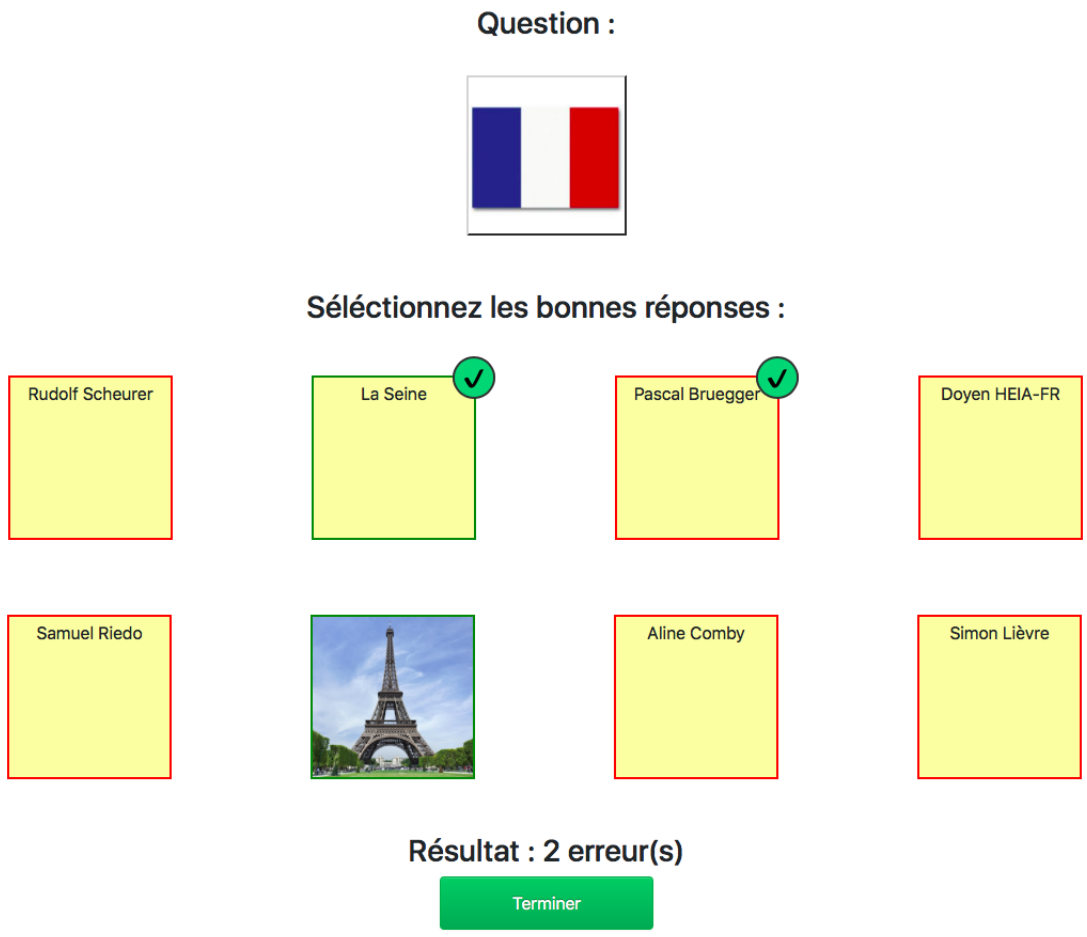


Figure 55 – Aperçu de la visualisation des bonnes et mauvaises réponses lors de la validation des réponses par le joueur

Nous pouvons voir les cadres rouges autour des mauvaises réponses et les cadres verts autour des bonnes réponses.

Tout est basé sur un test, le quizz est testé afin de savoir s'il est terminé ou pas. Un Boolean "isFinish" est initialisé à "False" dans le fichier "quizz-play.components.ts" lors du chargement de la page:

```
ngOnInit() {
    this.quizIsNull = false;
    this.isFinish = false;
    this.activatedRoute.params.subscribe((param:Params)=> {
        let questionName = param.id;
        this.quizService.get(questionName).subscribe((quiz:Quiz) => {
            this.currentQuiz = quiz;
            this.question   = quiz.question;
            this.answers    = quiz.answers;
        });
    })
    this.checkNull();
}
```

Les réponses sont affichées en deux temps, une fois lors de l'affichage du quizz après sa génération et une fois que le score est validé. Lorsque le quizz n'est pas fini (test: *ngIf="!isFinish"), donc pas validé, un certain style est appliqué:

```
<section *ngIf="!isFinish" class="plan plan1 cf">
    <div class="row ">
        <div *ngFor="let selectableAnswer of answers">
            <div class="div-quizz-answer">
                <div *ngIf="isAnImage(selectableAnswer.content)"
                     class="div-quizz-answer-border">
                ...

.div-quizz-answer-border{
    width:154px;
    height:154px;
    border-width:2px;
    border-style:outset;
    border-color:grey;
    ...
}
```

Lorsque le quizz est validé, donc quand il est fini (test: *ngIf="isFinish"), un certain style est affiché pour les bonnes réponses et un certain autre style pour les mauvaises réponses:

```
<section *ngIf="isFinish" class="plan plan1 cf">
    <div class="row ">
        <div *ngFor="let selectableAnswer of answers">
            <div class="div-quizz-answer">
                <div *ngIf="isAnImage(selectableAnswer.content)">
                    <div *ngIf="isGood(selectableAnswer?.name)"
                        class="div-quizz-answer-border-correct">
                    ...

                    <div *ngIf="!isGood(selectableAnswer?.name)"
                        class="div-quizz-answer-border-wrong">
                    ...

.div-quizz-answer-border-correct{
    width:154px;
    height:154px;
    border-width:2px;
    border-style: solid;
    border-color: green;
    ...
}

.div-quizz-answer-border-wrong{
    width:154px;
    height:154px;
    border-width:2px;
    border-style: solid;
    border-color: red;
    ...
}
```

Lorsque le joueur clique sur le bouton "Valider", le méthode "check()" est appelée et le Boolean "isFinish" passe à "true":

```
<div *ngIf="!isFinish" class="col-sm div-quizz-btn">
    <button (click)="check()" type="submit" class="btn btn-default quizz-btn">
        <span class="icon-check icon-2x"></span>
    </button>
</div>
```

Là aussi, ce bouton s'affiche seulement quand le test n'est pas fini (Boolean "isFinish" à "false"), la méthode "check()" met se Boolean à "true":

```
check(){
    this.currentQuiz.selected = this.selected;
    this.quizService.check(this.currentQuiz).subscribe((q: Quiz) => {
        this.resultQuiz = q;
        this.result = this.resultQuiz.result;
    });
    this.entityService.getEntity(this.question.name).subscribe((e: Entity) => {
        this.links = e.link;
    });
    this.isFinish = true;
}
```

Le Boolean "isFinish" à "true", les réponses sont réaffichées et testées afin que l'on sache si elles sont bonnes ou mauvaises, et le style convenable est affiché en fonction du résultat.

4.12.2 Gestion des images dans l'édition des entités

Dans la partie d'édition des entités, le téléchargement d'images n'était pas géré comme dans la partie ajout d'entités. Nous avons donc décidé dans les dernières heures du projet de réaliser cette fonction. Cette amélioration rendrait la partie administration plus performante. Sans cette fonction, l'administrateur a malgré tout accès au contenu, mais pour ajouter une image, il doit créer une nouvelle entité. Nous voulons éviter cela.

Nous avons implémenté les mêmes fonctionnalités que du côté ajout d'entités. C'est à dire, la gestion du téléchargement d'images et le fait que le bouton est inactif tant que tous les champs ne sont pas remplis:

Figure 56 – Gestion des images du côté de l'édition des entités

Nous pouvons voir que l'utilisateur peut télécharger une nouvelle image et en avoir l'aperçu. Le champ contenu n'étant pas rempli, les boutons "Valider" restent inactifs.

4.13 Améliorations

Il est clair que durant l'implémentation du projet, beaucoup d'idées naissent, mais nous ne pouvons malheureusement pas tout réaliser. Voici une liste de points intéressants à, pourquoi pas, implémenter dans le futur:

- **Jouer avec les liens :** nous avons pu voir dans la partie analyse et conception qu'une grande partie de la puissance d'une base de données Graph résidait dans les liens entre les entités. En effet, pour notre projet, afin de rendre le prototype le plus générique possible, nous avons nommé tous les liens de la même façon "KNOWS". Nous avons pensé qu'en nommant les liens de différentes manières, nous pourrions générer les quizz et surtout les questions en fonction des différents liens. Comme nous pouvons le voir dans l'exemple suivant, chacun des liens possède un nom, nous pourrions générer un quizz sur les monuments de Paris, le lien deviendrait la question:

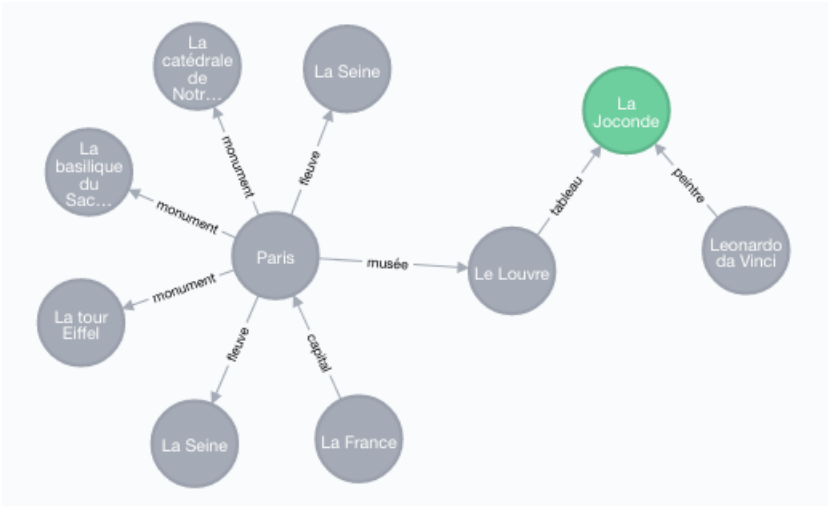


Figure 57 – Création d'une base de données basées sur les liens entre les entités

- **Les labels:** une des autres puissances de la base de données Graph est le notion de labels, que nous avons aussi définis dans les parties analyse et conception. Ici aussi, afin de rendre le prototype le plus générique possible, nous créons chacune des entités avec un label semblable "node". Nous avions pensé au début de la conception utiliser le label d'une entité pour définir sa catégorie, mais les requêtes devenaient vite plus complexes.
- **Génération des quizz:** un des points importants concernant l'amélioration de l'application se trouve au niveau de la génération du quizz. En effet lorsque le joueur génère le quizz, le nombre de réponses et de 8, ce chiffre est fixe et défini par une constante comme nous avons pu l'expliquer précédemment. Mais l'idéal serait que le joueur puisse lui-même choisir le nombre de réponses à afficher. Ce choix pourrait aussi se porter sur le nombre de bonnes et de mauvaises réponses à afficher, le nombre de bonnes réponses étant fixé à 2 dans le prototype.
- **Les catégories:** une idée qui nous est aussi venue en discutant avec les superviseurs et le mandant, est le fait de pouvoir gérer de manière plus concrète les catégories des entités. En effet, l'administrateur crée une entité en lui définissant une catégorie et cette catégorie est créée à ce moment si elle n'existe pas. Mais nous avons réfléchi à plusieurs solutions, les catégories pourraient être définies à l'avance dans un fichier texte ou XML, ou dans une interface d'administration nouvelle, afin que l'administrateur n'entre pas n'importe quoi dans le champ des catégories lors de la création d'une entité. Actuellement ce qui est délicat au niveau des catégories, est le fait que si l'administrateur crée une entité avec la catégorie "Ville" ou "ville", l'application va l'interpréter comme deux catégories différentes, ce qui est correct dans un sens, mais ce n'est pas ce qu'on veut. Le fait de définir à l'avance les catégories pour le domaine d'utilisation du quizz pourrait être avantageux pour la suite.

- **Recherche textuelle:** cette fois-ci c'est une idée venue du mandant qui a retenu notre attention. Actuellement, lors du listage des entités autant du côté jeu pour choisir un quizz ou du côté administration, nous pouvons appliquer un filtre de catégories sur cette liste afin de faciliter la recherche d'une entité. Mais l'idéal serait de pouvoir effectuer une recherche textuelle sur cette liste, cette fonction rendrait plus performante la recherche d'entités en plus du filtre de catégories.
- **Entité plus souple:** actuellement, le contenu d'une entité est défini soit par un texte ou soit par une image, mais pourquoi pas penser plus loin et combiner les deux ? Du texte avec une image ou même sur l'image, ceci pourrait rendre le quizz encore plus ludique.
- **Sécurité:** afin d'amener un peu de sécurité dans notre application, une amélioration importante, serait de mettre en place un système de login pour accéder à la partie administration de l'application. Ce point faisait partie des objectifs secondaires de notre cahier des charges, mais nous n'avons pas eu le temps de l'implémenter.
- **Historique des scores:** un point intéressant que nous pourrions améliorer, est le fait d'ajouter un historique des scores pour le joueur pour les différents quizz. Ce point avait été soulevé lors des séances avec le mandant et les superviseurs, durant les tests utilisateurs, certains utilisateurs nous ont aussi fait part de cette amélioration.
- **Côté sonore:** lors des tests utilisateurs, une idée d'amélioration intéressante nous a été transmise. Amener un côté sonore lors de la partie jeu afin de positiver ou non les bonnes et mauvaises réponses, cela pourrait amener un côté plus ludique et intuitif au jeu.
- **Algorithme de tirage des réponses:** l'algorithme permettant de tirer les mauvaises réponses dans la base de données peut être optimisé afin d'afficher des mauvaises réponses se rapprochant plus aux bonnes réponses, afin de rendre le quizz un peu plus compliqué à résoudre. Par exemple afin d'éviter, si l'on utilise ce quizz pour le sport et l'art, d'avoir une question "La Joconde" et d'avoir une réponse "Cristiano Ronaldo". L'alogoithme pourrait être basé sur les catégories par exemple.

Bien entendu, pour notre prototype du Virtual Memory, les améliorations sont infinies, mais nous avons pu voir que c'est en travaillant dessus que les idées d'améliorations naissent. Malheureusement, nous ne pouvons pas tout réaliser, mais nous espérons que ce projet continue de se développer.

4.14 Problèmes rencontrés

Tout au long du projet, nous sommes tombés sur une multitude de petits problèmes, mais quelques plus gros problèmes ont retenu notre attention.

4.14.1 Affichage

Un point nous a quand même fait nous poser des questions. Malheureusement, nous n'avons toujours pas pu résoudre ce mystère, malgré de bonnes heures passées dessus et beaucoup de recherches. Un problème d'affichage survient sur une page précise de l'application. En effet, nous avons remarqué la semaine avant la fin du projet, donc la partie implémentation pratiquement terminée, que la page permettant d'ajouter une entité ne s'affichait pas correctement environ une fois tous les 15 chargements sur le navigateur Google Chrome de la machine de Nicolas Stulz et sur le navigateur Safari de la machine de Loïc Dufresne. Le plus surprenant est que sur le navigateur Google Chrome de la machine de Loïc Dufresne, ce problème d'affichage n'apparaît jamais. Les pages d'ajout et de modification d'entités sont construites de la même manière et du côté de la modification, aucun souci. Voici le problème:

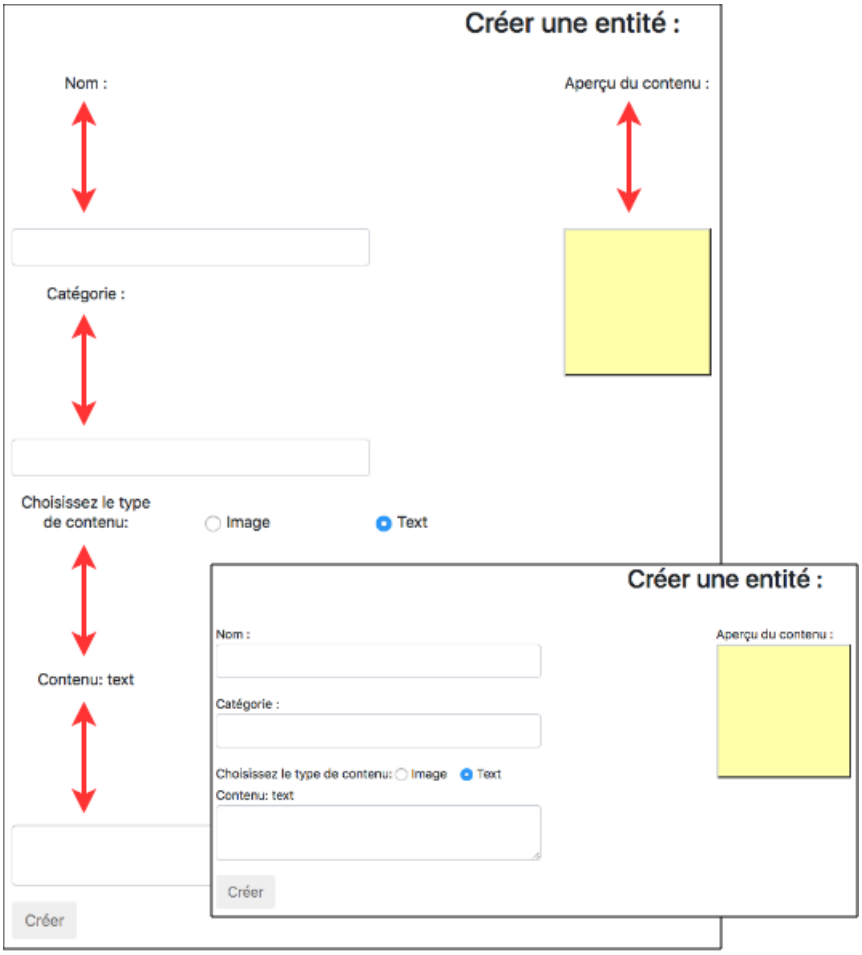


Figure 58 – Problème d'affichage sur la page d'ajout d'entités

Nous avons pensé à un problème de style, nous avons donc analysé tout le code CSS, nous avons aussi pensé à un problème de fermeture de balise ou même une fermeture de balises du côté HTML, mais ça ne coïncidait pas avec le fait que certains navigateurs n'affichaient pas cet écart.

Nous avons analysé la page HTML directement depuis le navigateur Google Chrome où l'erreur s'affichait, et nous avons eu une surprise:

```
.form label {
   font-size: 16px;
   position: relative;
   text-align: center;
   cursor: pointer;
   height: 150px;
   width: 150px;
}
```

Figure 59 – Écart sur les labels générés par Bootstrap

En inspectant la page, nous pouvons voir une hauteur et largeur de 150px est "de temps en temps" appliqué aux labels de cette page seulement par Bootstrap. Nous avons essayé de fixer ce bug en forçant dans notre propre code CSS un hauteur de 10px à chacun des labels via la balise et via une classe:

```
.label-size{
  height: 10px;
}
label{
  height: 10px;
}

<div class="label-size form-group div-admin-form">
    <label for="category">Categorie :</label>
    ...
</div>
```

Mais même avec une attribution d'une hauteur de 10px à tous les labels de la page, il arrive que cet écart de 150px s'affiche quand même, nous n'avons pas réussi à fixer cette erreur. Nous savons juste que ce problème provient de Bootstrap.

4.14.2 Lier et délier les entités

Sur ce point, nous nous sommes rendus compte de l'importance des tests utilisateurs. En effet, nous sommes tombés sur un problème assez important lors d'un test utilisateur. Un problème qui prouve que nous étions passés à côté d'une faille importante lors de nos tests de la partie implémentation.

Sur la page d'édition d'une entité, lorsque l'utilisateur lie et/ou délie les entités, il peut cocher et/ou décocher une ou plusieurs entités et cliquer sur le bouton "Valider", là aucun problème, tout cela est aussi valable avec un filtre. Mais lorsque l'utilisateur commence à cocher et/ou décocher des entités en appliquant un filtre sur ces entités, puis en recommençant se procédé en appliquant un autre filtre sans avoir valider avec le bouton précédemment, les entités cochées et/ou décochées préalablement sur l'ancien filtre, ne sont plus cochées et/ou décochées. L'application ne garde pas en mémoire les entités cochées et/ou décochées si un nouveau filtre est appliqué.

Malheureusement, ce problème a été découvert tardivement et nous avions fixé d'autres priorités ce qui fait que nous n'avons pas pu remédier à ce problème.

Nous invitons donc les utilisateurs à cocher et/ou décocher une ou plusieurs entités en appliquant qu'un filtre à la fois et en validant les choix avec le bouton "Valider".

4.14.3 Délais

Nous avons décelé des problèmes de délais dans la partie jeu de l'application. En effet, lorsque les entités sont chargées dans la liste afin que le joueur puisse faire le choix du quizz à jouer et lorsque le quizz est généré, avant que la question et les réponses s'affichent à l'écran, les messages d'informations, dans le cas où la base de données est vide et dans le cas ou il n'y a pas assez d'entités et/ou de liaisons entre les entités, sont affichés plusieurs micro secondes à l'écran avant que les entités soient chargées.

Ce problème vient du délai entre lesquels l'application requête la base de données et les entités soient affichées à l'écran.

Nous nous sommes donc penchés sur la résolution de ce problème en rajoutant un délai. Du côté HTML, nous testons sur le quizz fonctionnel, en particulier si les constantes sont respectées, le Boolean "quizIsNull" est à "true" si ces dernières ne sont pas respectées et le message d'informations s'affichent.

```
<div class="quizz-alert" *ngIf="quizIsNull">
    <h3>Veuillez creer plus d'entites et/ou de liaisons.</h3>
    <h3>Veuillez-vous referer aux constantes definies.</h3>
</div>
```

Une méthode "checkNull()" est appelée dans la méthode d'initialisation "ngOnInit()", cette méthode permet d'ajouter un délai si le quizz ne peut pas être joué, donc un délais avant d'afficher les messages d'informations:

```
checkNull(){
    setTimeout(() => {
        if(this.currentQuiz==null){
        this.quizIsNull = true;
        }
        else{
            this.quizIsNull = false;
        }
    }, 2000);
}
```

Ce procédé a été réalisé du côté de la page du choix de quizz et du côté de la page permettant de jouer au quizz choisi.

Avec ce délai de 2 secondes avant l'affichage des messages d'informations, nous avons réussi à résoudre ce problème assez dérangent au niveau navigation.

4.15 Entre la conception et le résultat final

Nous trouvons intéressant de faire le point entre ce que nous avons défini du côté implémentation et le résultat final du prototype Virtual Memory. En effet, les changements et modifications ont été nombreux, ce qui nous montre que c'est un faisant que les idées naissent, et ces idées ont pris de la valeur tout au long de l'implémentation et tout au long des séances avec notre mandant et nos superviseurs.

Rien de mieux qu'un simple exemple pour illustrer ces changements et ces nouvelles idées, voici la première maquette que nous avons réalisée lors de la conception pour la partie de l'ajout et l'édition d'une entité:

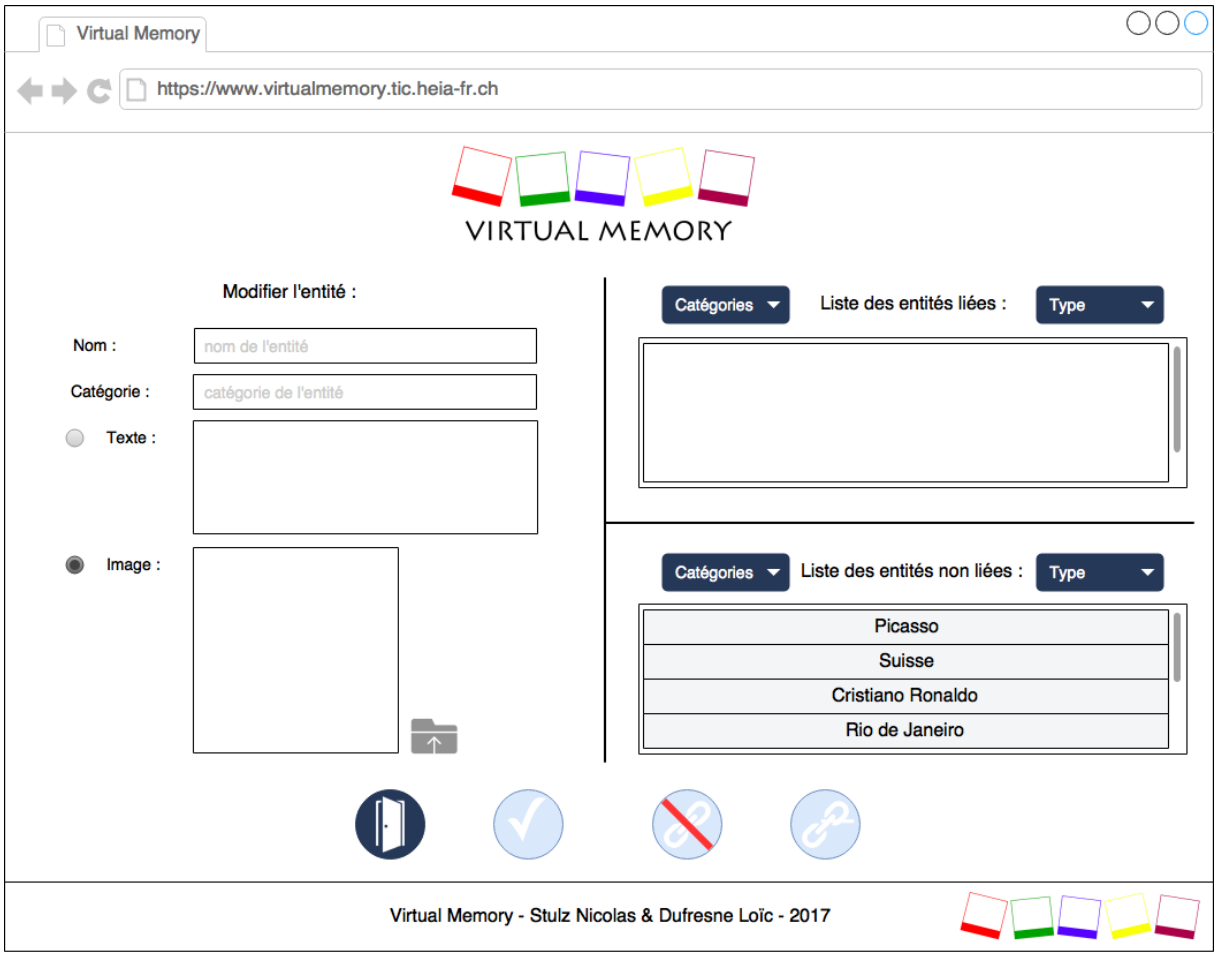


Figure 60 – Maquette de la partie administration, ajout/édition d'une entité (ajout) définie lors de la conception

Nous pouvons voir que lors de la conception, nous avions centralisé le fonction d'ajout et d'édition d'une entité sur la même page.

Et si nous comparons le résultat final sur notre prototype, au niveau de l'ajout d'entités:

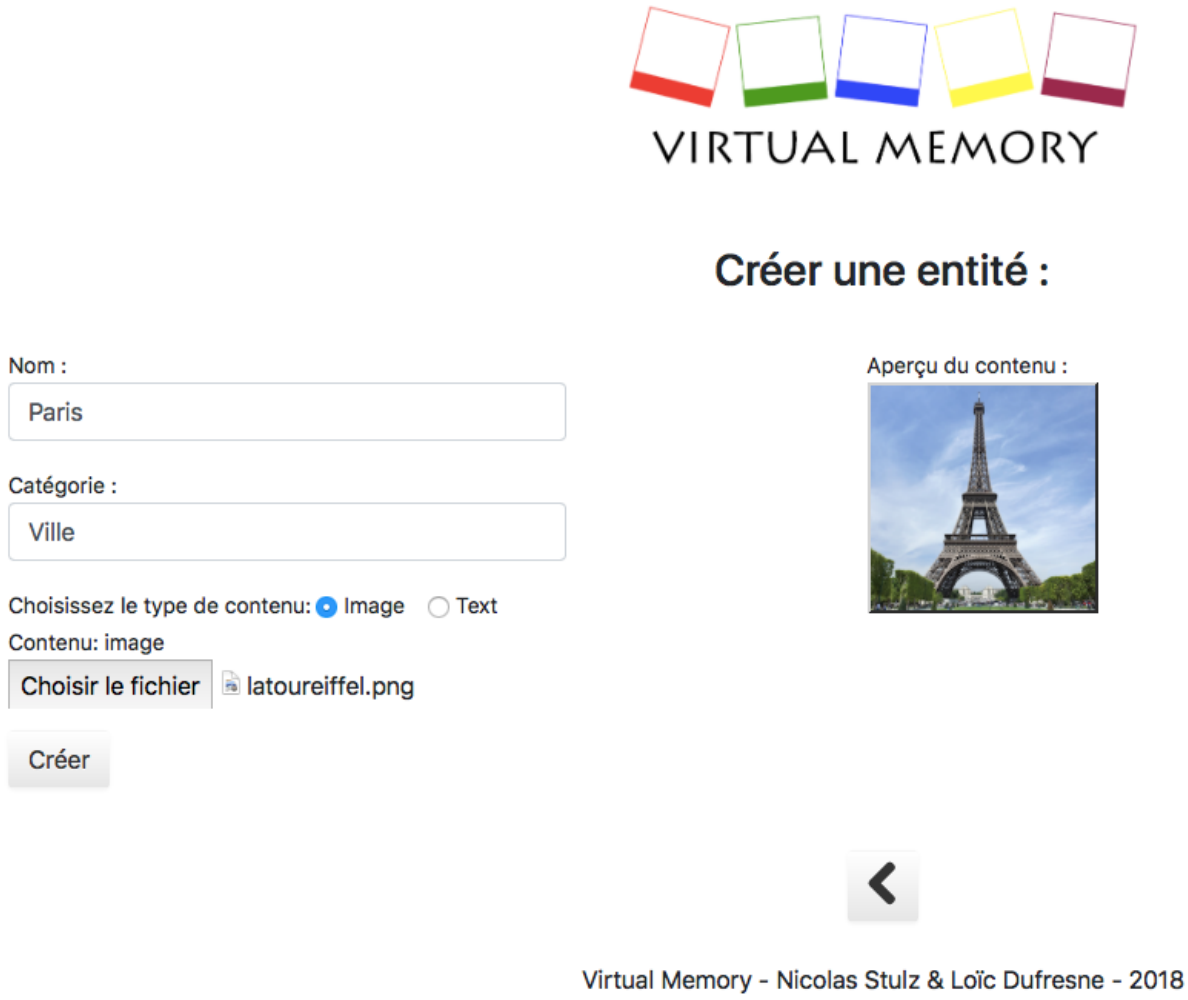


Figure 61 – Application finale de la partie administration, au niveau de l'ajout d'une entité

Nous avons séparé les parties ajout et édition d'entités, la gestion des liens vers les autres entités se passe entièrement du côté édition d'entités maintenant. Lorsque l'administrateur télécharge une image, il a même l'aperçu de cette image afin de savoir comment elle apparaîtra du côté jeu, nous avons ajouté cette fonctionnalité afin que l'administrateur puisse avoir une vision de ce qu'il crée.

Tous ces changements ont été pensés afin de rendre l'application plus logique et simple d'utilisation pour l'utilisateur. Nous avons essayé de simplifier au maximum le Virtual Memory. Tout ça pour dire qu'au niveau des tâches définies dans le cahier des charges, ces dernières sont respectées, mais la manière de faire, la manière de les implémenter, la manière de les afficher pour l'utilisateur a été repensée tout au long de l'implémentation.

4.16 Problème de version Neo4j

Lors de la réalisation du manuel d'installation de l'environnement, nous avons réinstallé tout l'environnement sur un Ubuntu. Nous avons téléchargé la dernière version de Neo4j (10.0.11), cette version étant sorti quelques jours avant la fin du projet, alors que nous avions réalisé notre projet avec la version de Neo4j (10.0.10).

Version avec laquelle nous avons développé le projet:

Figure 62 – Version de Neo4j 10.0.10 avec laquelle nous avons développé le projet

Version avec laquelle nous avons réalisé le manuel d'installation

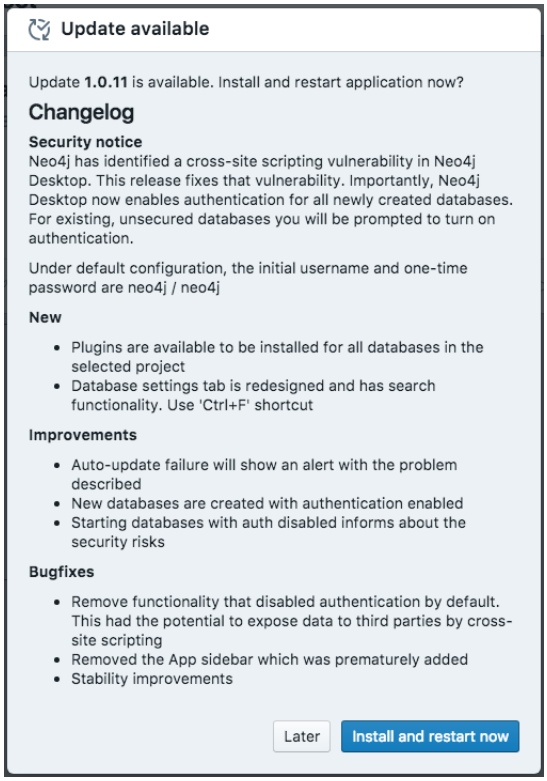


Figure 63 – Version de Neo4j 10.0.11 avec laquelle nous avons réalisé le manuel d'installation

Le problème est que l'ancienne version de Neo4j ne requiert aucune authentification (utilisateur et mot de passe) et la nouvelle version de Neo4j en requiert une. Nous pouvons le voir ci-dessus que les nouvelles bases de données sont crées avec une authentification. Cette ancienne version n'étant peut être plus possible d'obtenir, nous avons du nous adapter et modifier le code au tout dernier moment. Nous voulions un manuel d'installation à jour avec les dernières versions.

Nous avons du faire des recherches et ajouter dans le code un nom d'utilisateur et un mot de passe, après avoir effectué un test, nos modifications se sont avérées payantes, la base de données était opérationnelle et l'application pouvait y accéder après ces modifications.

Voici ces modifications apportées dans le fichier du Backend:
"main/java/ch.heiafr.virtual_memory/Constants".

```
// uri to connect to the running database
public final static String URI  = "bolt://localhost:7687";

public final static String USER = "neo4j";

public final static String PASSWORD = "admin";
```

L'utilisateur devra donc créer une base de données en s'authentifiant avec le nom d'utilisateur "neo4j" et le mot de passe "admin", mais cette démarche est décrite dans le manuel d'installation disponible en annexe.

Nous avons réussi à résoudre ce contretemps au dernier moment et heureusement.

4.17 Conclusion de l'implémentation

Pour cette partie, nous avons essayé de vous présenter les points les plus intéressants et importants, les points clés de notre application. Nous ne pouvons pas tout présenter, c'est clair, mais l'essentiel y est. Nous avons présenté notre manière de penser le codage du Virtual Memory.

Une chose importante dans cette partie aussi était de définir les fonctionnalités principales de l'application en relation avec le cahier des charges. Nous avons mis en place une grille d'évaluation afin que vous puissiez avoir une idée de l'état de l'avancement du projet au niveau implémentation lors de son rendu.

Nous avons aussi ressorti toutes les améliorations intéressantes pour la suite de ce projet, qui, on l'espère, puissent apporter des plus au projet dans le futur.

Des problèmes ont aussi été découverts, pour ce premier prototype, plusieurs ont été résolu, mais d'autres non, nous trouvons dommage de ne pas avoir pu tous les résoudre.

5 Tests

Ce chapitre traite de tests réalisés sur notre application. Nous avons réalisé des tests fonctionnels et des tests utilisateurs. Les tests utilisateurs ont bien été poussés, les tests fonctionnels auraient pu être plus poussés, mais nous sommes restés assez basique en nous concentrons quand même sur toutes les fonctionnalités. Concernant les tests unitaires, malheureusement, nous avons décidé de ne pas en réaliser par faute de temps.

5.1 Tests fonctionnels

Pour les tests fonctionnels, nous avons repris tous les points clés concernant l'implémentation que nous avons définis dans le chapitre précédent en relation avec le cahier des charges. Toutes ces étapes sont testées dans leurs fonctions principales et ensuite, ces fonctions sont testées plus en détail.

Tests	Résultats attendus	Résultat	Etat
Ajouter une entité avec un nom n'existant pas	L'entité doit être ajoutée à la base de données	L'entité a bien été ajoutée à la base de données et apparait dans l'application	
Ajouter une entité avec un nom déjà existant	Un message d'alerte doit être affiché L'entité ne doit pas être ajoutée à la base de données (avoir plusieurs entités avec la même catégorie doit être possible)	Un message d'alerte est affiché L'entité n'est pas ajoutée à la base de données et n'apparait pas dans l'application, un message d'erreur apparaît	
Ajouter une entité avec une catégorie déjà existante	L'entité doit être ajoutée à la base de données (avoir plusieurs entités avec la même catégorie doit être possible)	L'entité a bien été ajoutée à la base de données et apparait dans l'application	
Ajouter une entité avec un contenu déjà existant	L'entité doit être ajoutée à la base de données (avoir plusieurs entités avec le même contenu doit être possible)	L'entité est ajoutée à la base de données et apparaît dans l'application	
Ajouter une entité avec un nom déjà existant en jouant avec les majuscule	L'entité doit être considérée comme une entité avec un nom déjà existant et ne doit pas être ajouté à la base de données	L'entité est ajoutée à la base de données et apparaît dans l'application (point à améliorer, le nom de l'entité est indirectement le même)	
Ajouter une entité avec une catégorie déjà existante en jouant avec les majuscule	L'entité doit être ajoutée à la base de données, mais la catégorie doit être considérée comme la même	L'entité est ajoutée à la base de données et apparaît dans l'application (point à améliorer, la catégorie est indirectement la même)	
Ajouter une entité sans nom	L'entité ne doit pas être ajoutée à la base de données	L'entité ne peut pas être ajoutée car tant que tous les champs ne sont pas remplis, le bouton « Ajouter » n'est pas actif	
Ajouter une entité sans catégorie	L'entité ne doit pas être ajoutée à la base de données	L'entité ne peut pas être ajoutée car tant que tous les champs ne sont pas remplis, le bouton « Ajouter » n'est pas actif	
Ajouter une entité sans contenu	L'entité ne doit pas être ajoutée à la base de données	L'entité ne peut pas être ajoutée car tant que tous les champs ne sont pas remplis, le bouton « Ajouter » n'est pas actif	

Tests	Résultats attendus	Résultats	Etat
Supprimer une entité	Un message d'alerte doit être affiché L'entité et les liens doivent être supprimés de la base de données et elle disparaît de l'application	Un message d'alerte est affiché L'entité et les liens sont supprimés de la base de données et elle disparaît de l'application	
Supprimer une entité sans nom	Un message d'alerte doit être affiché L'entité et les liens doivent être supprimés de la base de données et elle disparaît de l'application	Un message d'alerte est affiché L'entité et les liens sont supprimés de la base de données et elle disparaît de l'application (à noter qu'une entité sans nom ne peut pas être ajouté)	
Supprimer une entité sans catégorie	Un message d'alerte doit être affiché L'entité et les liens doivent être supprimés de la base de données et elle disparaît de l'application	Un message d'alerte est affiché L'entité et les liens sont supprimés de la base de données et elle disparaît de l'application (à noter qu'une entité sans nom ne peut pas être ajouté)	
Supprimer une entité sans contenu	Un message d'alerte doit être affiché L'entité et les liens doivent être supprimés de la base de données et elle disparaît de l'application	Un message d'alerte est affiché L'entité et les liens sont supprimés de la base de données et elle disparaît de l'application (à noter qu'une entité sans nom ne peut pas être ajouté)	
Supprimer une entité en appliquant le filtre	Un message d'alerte doit être affiché L'entité et les liens doivent être supprimés de la base de données et elle disparaît de l'application	Un message d'alerte est affiché L'entité et les liens sont supprimés de la base de données et elle disparaît de l'application	

Tests	Résultats attendus	Résultats	Etat
Editer une entité avec un nom n'existant pas	Un message d'alerte doit être affiché L'entité doit être modifiée dans la base de données	Un message d'alerte est affiché L'entité a bien été modifiée dans la base de données et apparait dans l'application (avoir plusieurs entités avec le même nom ne doit pas être possible)	
Editer une entité avec un nom déjà existant	Un message d'alerte doit être affiché L'entité ne doit pas être modifiée dans la base de données	Un message d'alerte est affiché L'entité n'est pas modifiée dans la base de données et n'apparait pas dans l'application, un message d'erreur apparaît (avoir plusieurs entités avec la même catégorie doit être possible)	
Editer une entité avec une catégorie déjà existante	Un message d'alerte doit être affiché L'entité doit être modifiée dans la base de données	Un message d'alerte est affiché L'entité a bien été modifiée dans la base de données et apparait dans l'application (avoir plusieurs entités avec la même catégorie doit être possible)	
Editer une entité avec un contenu déjà existant	Un message d'alerte doit être affiché L'entité doit être modifiée dans la base de données	Un message d'alerte est affiché L'entité est modifiée dans la base de données et apparaît dans l'application (avoir plusieurs entités avec le même contenu doit être possible)	
Editer une entité avec un nom déjà existant en jouant avec les majuscule	Un message d'alerte doit être affiché L'entité doit être considérée comme une entité avec un nom déjà existant et ne doit pas être modifiée dans la base de données	Un message d'alerte est affiché L'entité est modifiée dans la base de données et apparaît dans l'application (point à améliorer, le nom de l'entité est indirectement le même)	
Editer une entité avec une catégorie déjà existante en jouant avec les majuscule	Un message d'alerte doit être affiché L'entité doit être modifiée dans la base de données, mais la catégorie doit être considérée comme la même	Un message d'alerte est affiché L'entité est modifiée dans la base de données et apparaît dans l'application (point à améliorer, la catégorie est indirectement la même)	
Editer une entité avec un nom vide	Un message d'alerte doit être affiché L'entité ne doit pas être modifiée dans la base de données	Un message d'alerte est affiché L'entité ne peut pas être modifiée car tant que tous les champs ne sont pas remplis, le bouton « Valider » n'est pas actif	
Editer une entité avec une catégorie vide	Un message d'alerte doit être affiché L'entité ne doit pas être modifiée dans la base de données	Un message d'alerte est affiché L'entité ne peut pas être modifiée car tant que tous les champs ne sont pas remplis, le bouton « Valider » n'est pas actif	
Editer une entité avec un contenu vide	Un message d'alerte doit être affiché L'entité ne doit pas être modifiée dans la base de données	Un message d'alerte est affiché L'entité ne peut pas être modifiée car tant que tous les champs ne sont pas remplis, le bouton « Valider » n'est pas actif	

Tests	Résultats attendus	Résultats	Etat
Lier une entité à une ou plusieurs entités sans utiliser les filtres	Un message d'alerte doit être affiché Les entités doivent être liées bidirectionnellement	Un message d'alerte est affiché Les entités ont bien été liées bidirectionnellement	
Lier une entité à une ou plusieurs entités en utilisant un filtre	Un message d'alerte doit être affiché Les entités doivent être liées bidirectionnellement	Un message d'alerte est affiché Les entités ont bien été liées bidirectionnellement	
Lier une entité à une ou plusieurs entités en utilisant plusieurs filtres	Un message d'alerte doit être affiché Les entités doivent être liées bidirectionnellement	Un message d'alerte est affiché Les entités sont liées par rapport aux entités cochées lors du dernier filtre, en passant d'un filtre à l'autre, les entités cochées ne sont pas gardées en mémoire	
Délier une entité à une ou plusieurs entités sans utiliser les filtres	Un message d'alerte doit être affiché Les entités doivent être déliées bidirectionnellement	Un message d'alerte est affiché Les entités ont bien été déliées bidirectionnellement	
Délier une entité à une ou plusieurs entités sans utilisant un filtre	Un message d'alerte doit être affiché Les entités doivent être déliées bidirectionnellement	Un message d'alerte est affiché Les entités ont bien été déliées bidirectionnellement	
Délier une entité à une ou plusieurs entités en utilisant plusieurs filtres	Un message d'alerte doit être affiché Les entités doivent être déliées bidirectionnellement	Un message d'alerte est affiché Les entités sont déliées par rapport aux entités cochées lors du dernier filtre, en passant d'un filtre à l'autre, les entités cochées ne sont pas gardées en mémoire	

Tests	Résultats attendus	Résultats	Etat
Sélectionner une entité afin de jouer au quizz	L'entité doit être sélectionnée et le quizz généré	L'entité est bien sélectionnée et le quizz est généré	
Sélectionner une entité afin de jouer au quizz en utilisant un filtre	L'entité doit être sélectionnée et le quizz généré	L'entité est bien sélectionnée et le quizz est généré	
Sélectionner les entités réponses	Les réponses doivent être sélectionnées	Les entités sont bien sélectionnées	
Valider la sélection	Les réponses doivent être validées, le score et les réponses affichés	Les réponses sont validées, le score et les réponses affichés	

Plusieurs petits détails sont à peaufiner dans notre application, nous avons réussi à en contrer un bon nombre, mais il en reste plusieurs à traiter.

Le plus gros travail maintenant est de traiter le fait que le nom, la catégorie ou le contenu d'une entité, en jouant avec les majuscules, peut être égal à un autre. Il faut fixer des règles, par exemple, pour dire que "Ville" est égal à "ville". Pour les catégories, nous l'avons déjà cité, mais le fait de définir à l'avance les catégories pourrait déjà fixer certains points.

Il s'agit actuellement d'un prototype, des critères sont encore à fixer et c'est aux futurs repreneurs du Virtual Memory de fixer ces points.

5.2 Tests utilisateurs

Les tests utilisateurs vont nous permettre de nous rendre compte si notre application est logique, facile à utiliser et compréhensible pour n'importe qui. Nous avons défini une liste de tâches à réaliser faisant partie d'un scénario au sein de notre application autant au niveau jeu et administration. Nous avons demandé à des personnes lambda de réaliser ces tâches. Notre rôle est d'être attentif à leur réaction et leur comportement afin que nous puissions nous faire une idée de l'utilisabilité de notre application.

Nous avons mis en place un scénario pour que l'utilisateur comprenne mieux ce qu'il doit réaliser et afin qu'il se sente plus à l'aise. Voici les informations que reçoit un utilisateur:

Tests utilisateurs

Virtual Memory est une application Web, créée par Nicolas Stulz & Loïc Dufresne, elle permet de créer et de jouer à n'importe quel type de quizz. Cette application peut être utilisée dans une multitude de domaines comme dans le scolaire, par exemple, pour un cours d'histoire, d'allemand ou de français, dans les ressources humaines afin que ces dernières fassent passer un test à leurs futures recrues, ou tout simplement entre amis.

Le but de ce projet est de créer une base, facilement paramétrable, afin qu'elle puisse être adaptée pour toutes sortes de domaines.

Nom & prénom :
Age :
Profession :
Lieu & date :

Scénario : Vous devez créer votre propre quizz sur la ville de Paris et y jouer, ce scénario est réalisé en 2 étapes, dans un premier temps la création du quizz et, deuxièmement, la partie amusante, le jeu :

1. Créer le quizz
- Créer une entité « Paris » possédant une catégorie « Ville » et un contenu texte « Paris, la ville des lumières »
- Lier l'entité « Paris » aux entités « La France », « Le Brésil », « La Seine » et « Le Louvre »
- Créer une entité « La tour Eiffel » possédant une catégorie « Monument » et un contenu image avec l'image « latoureiffel.png » disponible sur le bureau
- Lier l'entité « La tour Eiffel » aux entités « La France », « Paris », « La Seine » et « Le Louvre »
- Délier l'entité « Paris » de l'entité « Brésil » car nous nous sommes précédemment trompé précédemment
- Supprimer l'entité « Brésil »
- Revenir au menu principal

2. Jouer le quizz
- Démarrer le quizz « Paris »
- Sélectionner les bonnes réponses et valider votre choix
- Démarrer le quizz « La France »
- Sélectionner les bonnes réponses et valider votre choix

Figure 64 – Scénario mis en place pour les tests utilisateurs

L'utilisateur réalise le scénario tout seul, nous l'observons, mais nous le laissons faire afin de voir s'il arrive à réaliser ces tâches tout seul. Une fois les tâches réalisées, l'utilisateur remplit une feuille, disponible en annexe, avec remarques, difficultés, améliorations... Ces tâches ce devraient être réalisés en 5 minutes environ.

5.2.1 Résultats

Utilisateur : Alex Travasso, 22 ans, informaticien
Temps pour la réalisation de ces tâches : 4 minutes
Partie administration (créer le quizz)
Autres remarques : • Petits soucis d'affichage du côté de l'ajout d'entités • Petits soucis au niveau du liement et du déliement d'entité avec application de plusieurs filtres
Partie jeu (jouer le quizz)
Impressions générales : • Je ne sais pas si j'ai gagné
Autres utilités du Virtual Memory
• Examens de fin de semestre à la HEIA-FR
Impression des développeurs
• Beaucoup de facilité • L'application semble bien compréhensible

Le premier utilisateur est aussi étudiant en télécommunication à la HEIA-FR, nous avons pu remarquer qu'il avait beaucoup de facilité et qu'il a mis le doigt sur un problème majeur au niveau de la création de liens entre les entités, un point que nous n'avions pas vu. Ce problème est décrit après les tests utilisateurs.

Utilisateur : Marie-Denise Dufresne, 50 ans, assistante en pharmacie
Temps pour la réalisation de ces tâches : 12 minutes
Partie administration (créer le quizz)
Impressions générales : • Pas compliqué et vite compris comment créer le quizz **Difficultés :** • Je ne suis pas trop « ordinateur », mais compréhensible rapidement
Partie jeu (jouer le quizz)
Difficultés : • Simple à utiliser, on ne passe pas par « 4 chemins » **Améliorations :** • A la fin du quizz, un petit historique des résultats
Autres utilités du Virtual Memory
• Dans les e-learning afin d'améliorer nos connaissances des médicaments
Impression des développeurs
• Pas vraiment facile au début, peut de cliquer sur les boutons • Après avoir compris comment l'application fonctionne, plus d'hésitations et tout allait très vite

Pour ce deuxième utilisateur, la personne ne connaît pas très bien le monde de l'informatique et était très hésitante lorsqu'elle s'est retrouvée devant l'application. Au début, elle n'osait pas cliquer sur les boutons, mais une fois qu'elle a compris le principe, plus confiante, elle trouvait l'application très logique et simple d'utilisation.

Comme amélioration, elle a soulevé un point dont nous avions aussi pensé, qui serait un historique des scores des quizz.

C'est un joli challenge relevé, le fait qu'une personne ne touchant presque jamais aux ordinateurs et ne connaisse pas le monde de l'informatique, arrive à s'en sortir aussi facilement et trouve notre application simple d'utilisation.

Utilisateur : Nathan Dufresne, 24 ans, laborant en chimie
Temps pour la réalisation de ces tâches : 8 minutes
Partie administration (créer le quizz)
Impressions générales : • L'idée est sympa, le fait de réaliser des quizz soit même est chouette • Simple à utiliser après 1 ou 2 créations **Difficultés :** • Les personnes qui n'utilisent pas souvent l'ordinateur ne savent peut-être pas où il faut ajouter, supprimer et éditer une entité **Améliorations :** • Noter où il faut ajouter une entité (+), supprimer (-) et éditer avec une petite annotation quand la souris passe sur la case
Partie jeu (jouer le quizz)
Impressions générales : • L'idée de lier des images, des mots et des phrases à une entité est vraiment top **Difficultés :** • Pas d'endroit pour avoir un retour sur les différents quizz fait (historique des résultats) **Améliorations :** • Historique des scores des quizz **Autres remarques :** • Utiles pour les écoles primaires, CO et apprentissages
Autres utilités du Virtual Memory
• Dans mon apprentissage, cela peut être utile pour certains cours (chimie organique, allemand, anglais...)
Impression des développeurs
• Grande facilité • Beaucoup d'enthousiasme à créer et jouer au quizz • Vraiment intéressé et curieux

Ce troisième utilisateur, nous permet d'avoir un oeil jeune sur notre application tout en n'étant pas vraiment du monde informatique, ce qui peut être intéressant.

Nous avons pu observer que cet utilisateur était très curieux et intéressé par notre application, il avait du plaisir à créer et jouer au Virtual Memory, ce qui nous a rendu une certaine satisfaction. Pour la réalisation des tâches, il a eu une grande facilité à les réaliser et nous a répété à plusieurs reprises que l'application était logique et bien compréhensible, mais qu'il pensait quand même que quelques indications ou annotations pouvaient être utile sur les différents boutons, ce qui peut être intéressant. Là aussi, l'utilisateur nous a dit qu'un historique des scores pouvait être une bonne amélioration.

L'utilisateur nous a aussi fait part que dans ces formations passées comme à l'école ou en apprentissage, le Virtual Memory aurait pu être un bon moyen d'apprentissage

Utilisateur : Quentin Pirlet, 23 ans, ingénieur en télécommunications
Temps pour la réalisation de ces tâches : 6 minutes
Partie administration (créer le quizz)
Impressions générales : • Assez lent pour le chargement de contenu **Améliorations :** • Revenir au menu principal avec un bouton en haut de page
Partie jeu (jouer le quizz)
Impressions générales : • Assez lent pour le chargement de contenu **Améliorations :** • Mentionner plus clairement les fausses réponses (mettre une croix au lieu d'un « vu » en complément du carré rouge)
Autres utilités du Virtual Memory
• Révision des examens à l'aide de contenu visuel et ludique • Gestions des analyses de décisions en macroéconomie avec le modèle Solow
Impression des développeurs
• Grande facilité • Œil plus critique et technique

Avec cet utilisateur, nous retrouvons de nouveau un oeil venu tout droit du monde de l'informatique. Du côté technique, il a pu soulevé que certains chargements étaient des fois un peu lent, sûrement la base de données qui peine un peu. Il trouve aussi que le retour sur le menu principal n'est pas très logique avec le logo et préférerait un bouton de retour en haut de la page.

Pour le côté jeu, le fait d'entourer les bonnes et mauvaises réponses respectivement en vert et en rouge est sympa, mais il préférerait une autre manière de signaler les mauvaises réponses. Le fait de garder les mauvaises réponses sélectionnées avec le "vu" pourrait basculer vers une croix lors de la validation des réponses.

Comme utilité de ce quizz, il le verrait bien dans la révision d'examens ou dans certains cours comme dans la gestion d'analyse afin d'y apporter un côté ludique.

Utilisateur : Daniel Dufresne, 52 ans, maître socio-professionnel
Temps pour la réalisation de ces tâches : 7 minutes
Partie administration (créer le quizz)
Impressions générales : • Bonne compréhension générale • Intuitif **Difficultés :** • Aucune
Partie jeu (jouer le quizz)
Impressions générales : • Sympa • Bonne évolution dans le jeu **Améliorations :** • Apporter un fond sonore en positivant ou pas les réponses
Autres utilités du Virtual Memory
• Dans le cadre de ma classe avec des quizz correspondant à mes élèves
Impression des développeurs
• Bonne facilité • Vraiment intéressé en posant plein de questions sur l'avenir de ce prototype

Nous avons vu que notre Virtual Memory avait un certain potentiel dans le cercle scolaire, il était important pour nous d'avoir un retour sur une personne travaillant dans ce milieu, et nous n'en sommes pas déçus.

Cet utilisateur est maître socioprofessionnel à la fondation de Verdeil et donne des cours de mathématiques, géométrie, informatique et autres à des jeunes en difficultés. Il nous a fait part qu'il y a quelques années il utilisait un type de quizz dans ces cours afin de rendre l'apprentissage de la matière plus ludique et motivante. Après avoir testé notre prototype, il nous a dit qu'il pourrait très bien utilisé ce prototype pour ces cours et qu'il a particulièrement aimé le fait qu'on puisse créé ces propres quizz et le fait de pouvoir mélanger le texte et les images dans le même quizz.

Très intéressé, il a eu une bonne facilité à utiliser la partie administration et jeu du Virtual Memory. Afin de rendre le projet encore plus intuitif, il nous propose d'amener un côté sonore lors de la phase jeu.

Utilisateur : Lauriane Tardy, 22 ans, laborantine en chimie
Temps pour la réalisation de ces tâches : 7 minutes
Partie administration (créer le quizz)
Impressions générales : • Bonne **Améliorations :** • Recherche textuelle des entités
Partie jeu (jouer le quizz)
Impressions générales : • Bonne **Améliorations :** • Recherche textuelle des entités **Améliorations :** • Un peu trop facile au niveau des réponses vu que c'est aléatoire
Autres utilités du Virtual Memory
• Pas forcément dans ma profession, mais plutôt dans l'apprentissage des matières
Impression des développeurs
• Bonne facilité • Intéressée (à continué à jouer après le test)

Pour terminer, ces tests utilisateurs, qui nous confie encore une fois que dans l'apprentissage des matières ce quizz aurait sa place. Une amélioration qu'elle propose est au niveau de la recherche des entités, mettre en place une recherche textuelle, car même avec les filtres s'il y a beaucoup d'entités, ça peut devenir plus compliqué à trouver une entité en particulier.

Une remarque venant de cet utilisateur a attiré notre attention. Il est vrai que lorsque l'on joue au quizz, les mauvaises réponses sont tirées aléatoirement dans la base de données. Mais pourquoi ne pas optimiser cet algorithme afin de ne pas avoir n'importe quelles mauvaises réponses et avoir des mauvaises réponses se rapportant plus à la question afin de rendre le quizz un peu plus compliqué. Car il est vrai que dans certains cas, comme nous l'a fait remarquer cet utilisateur, le quizz pouvait être relativement simple et logique à résoudre.

5.2.2 Synthèse des tests

Dans l'ensemble ces tests se sont bien passés pour les utilisateurs, nous avons fait une liste des points à souligner qui ressortent de ces tests utilisateurs:

- Notre application est très logique et simple d'utilisation autant pour les utilisateurs connaissants bien le monde informatique et surtout des utilisateurs ne connaissant pas vraiment ce monde. Tous les utilisateurs ont bien compris le fonctionnement du prototype et l'ont trouvé très intuitif. Voici le premier retour que nous avons, ce qui est très gratifiant, car c'était le but du Virtual Memory.
- Le Virtual Memory est très ludique et professionnel en même temps. Il pourrait être utilisé autant au niveau de l'apprentissage scolaire qu'au niveau de l'apprentissage professionnel. Ce prototype serait une bonne solution afin de rendre un cours plus ludique et intuitif.
- Le fait de mélanger et lier du texte, des phrases et des images est intéressant et ludique.
- Plusieurs utilisateurs nous ont fait part de leur souhait d'avoir un historique des quizz avec les différents scores, ce qui peut être vraiment sympa.
- Amener un côté sonore lors du jeu, pourrait rendre le quizz plus ludique.
- Améliorer l'algorithme d'affichage des réponses afin que les bonnes et mauvaises réponses appartiennent au même domaine, afin de rendre le quizz plus compliqué à résoudre.
- Comme nous l'avons cité dans les améliorations, la recherche textuelle pour les listes d'entités peut amener un côté permettant la simplification de la recherche des entités.
- Un problème lors de la création de liens a été découvert avec l'application des filtres, malheureusement nous n'avons pas eu le temps de le traiter, mais nous l'avons documenté.
- L'écart des labels lors de l'affichage de la page concernant l'ajout d'entités a été repéré par un utilisateur, ce problème a été documenté.
- Nous avons remarqué qu'aucun des utilisateurs n'a utilisé les filtres de catégories sur les listes d'entités.

5.2.3 Jouabilité du quizz

Dans le premier test utilisateur, nous avons remarqué qu'un point en particulier n'était pas très clair pour le joueur. Lorsqu'il sélectionnait les réponses dans le quizz et qu'ils validaient leur choix, il s'attendait à pouvoir revalider leur choix en cochant et/ou décochant d'autres réponses afin d'arriver au résultat de 0 erreur. Nous étions partis dans l'optique qu'une fois le choix validé, le score s'affichait et le quizz était terminé, donc cette fonctionnalité n'était pas valable, le joueur devait recommencer un autre quizz. Pour finir, nous avons décidé de laisser le bouton "Valider" et le joueur peut revalider ces réponses à plusieurs reprises, voici l'aperçu des modifications:

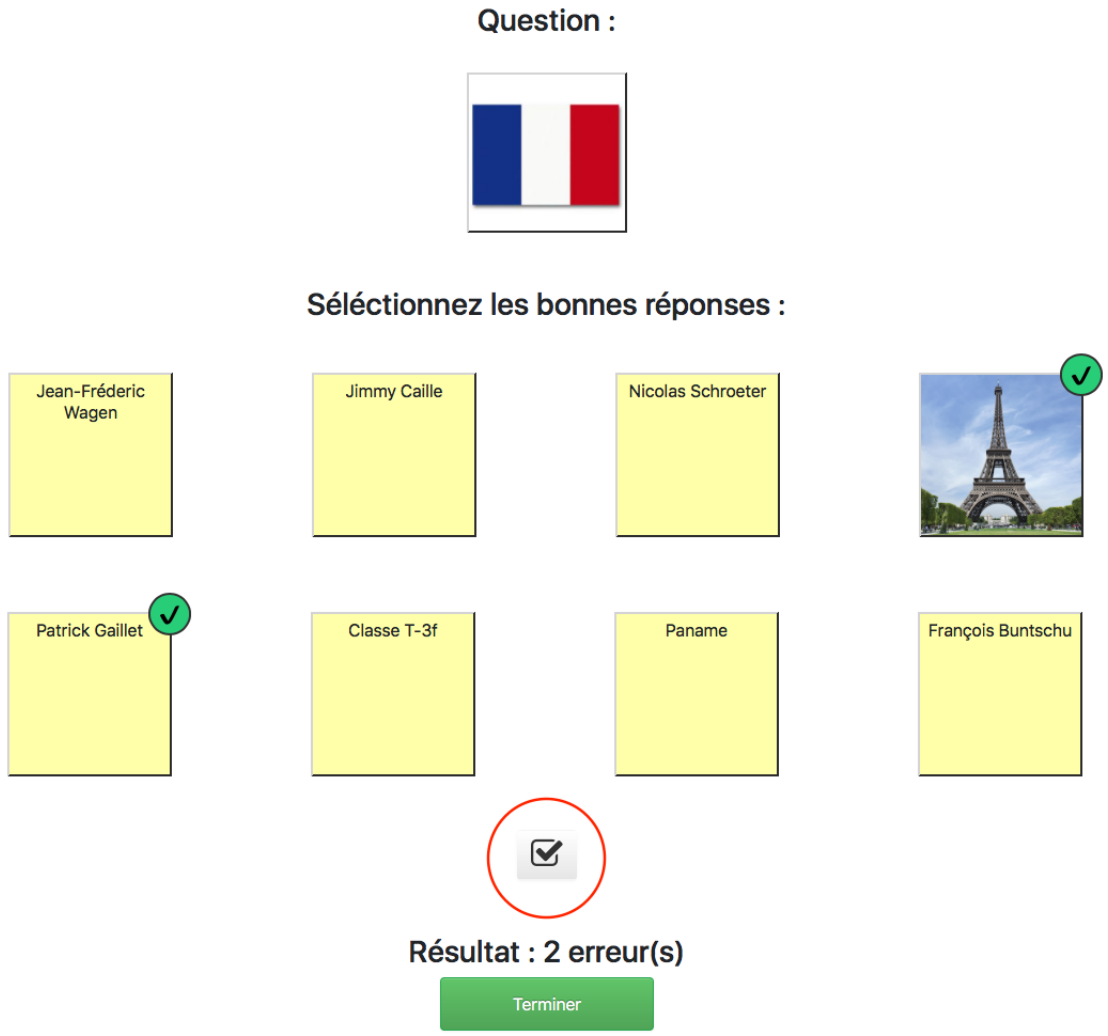


Figure 65 – Le bouton "Valider" est encore disponible après une première vérification

Nous pouvons voir dans l'encadré rouge que le bouton "Valider" est toujours actif et que le joueur peut vérifier son score jusqu'à obtenir 0 erreur.

Entre temps, nous avons implémenté l'affichage des bonnes et des mauvaises réponses avec les carrées respectivement vert et rouge. Nous avons redemandé à l'utilisateur de tester le jeu, et il nous a dit que le fait d'afficher ces carrés afin de montrer aux joueurs les bonnes et mauvaises réponses donnait un plus au jeu et que le bouton "Valider" n'était plus vraiment utile dans ce cas là. Pour finir, nous avons remodifié le code afin que de ne plus afficher le bouton "Valider" après la première vérification.

5.2.4 Conclusion des tests

Nous pouvons voir que les tests utilisateurs nous permettent de voir plus loin, certaines choses que nous, développeurs, ne voyons pas. Nous avons choisi des utilisateurs de plusieurs domaines et plusieurs âges afin d'avoir des retours très variés, au niveau technique, fonctionnalité et même jouabilité.

Nous sommes assez satisfaits du retour sur ces tests. Dans l'ensemble, les utilisateurs étaient motivés et ont apprécié manipuler le Virtual Memory, autant au niveau administration et au niveau jeu. Pour tous, ça paraissait comme quelque chose nouveau, quelque chose d'inédit.

Tous ces utilisateurs ont trouvés une utilité concrète pour ce quizz dans leur domaine de travail. Nous pouvons en ressortir que ce quizz à un potentiel au niveau de l'apprentissage des connaissances, autant au niveau scolaire qu'au niveau professionnel.

Pour nous, ces tests utilisateurs sont très concluants.

6 Conclusion

6.1 Atteinte des objectifs

Dans un premier temps, nous nous sommes penchés sur l'analyse des technologies pouvant permettre la réalisation du projet. Cette partie fut la plus importante, car elle nous a permis de faire un choix sur les meilleures technologies afin de réaliser l'application Virtual Memory.

Le choix du type de base de données fut la plus grosse décision que nous avions à prendre, mais nous avions soif de connaissance et pour sortir un peu des sentiers battus, nous avons décidé d'utiliser un type de bases de données dont nous n'avions pas de connaissances. En effet, durant les cours au sein de notre formation, nous avons vu un bon nombre de types de bases de données, mais les bases de données Graph nous était inconnu et d'après notre analyse, ce type de bases de données remplissait au mieux les critères de nos besoins. Nous nous sommes donc penchés vers cette direction et nous avons appris à travailler avec.

Un bon nombre de technologies était imposé pour ce projet et pour la plupart, des technologies nouvelles pour nous. En parallèle à la phase d'analyse et de conception, nous avons commencé à réaliser des tutoriels afin d'être prêt à attaquer l'implémentation du projet de la meilleure des manières.

Ce qui fut aussi très intéressant, au niveau de la conception, est le fait que nous avions un cahier des charges clair et précis bien entendu, mais nous pouvions en même temps laisser parler notre imagination surtout au niveau des maquettes. La réalisation des maquettes fut aussi un gros travail, pour vous dire, les dernières maquettes ne ressemblaient en rien aux premières maquettes, et c'était le cas aussi les différents diagrammes, toutes les semaines, nous revenions aux séances avec de nouvelles idées, des changements, et aussi des problèmes rencontrés. Mais toutes ces nouvelles idées ou problèmes qui ont apportés des changements, nous ont fait réaliser une chose très importante, nous partions toujours en premier dans des manières de faire complexes, pour au final arriver avec des idées et des manières de faire beaucoup plus simples et intéressantes, pour arriver à une solution de plus en plus générique, ce qui était aussi un de nos buts.

Pour résumer, au niveau de l'atteinte des objectifs, les principaux objectifs qui nous ont été fixés, en référence au cahier des charges, ont été atteints. Ces objectifs n'ont pas été modifiés durant le projet, ils sont restés les mêmes. Ce qui a beaucoup varié était plutôt la manière de les réaliser, ce qui nous a permis d'apprendre encore plus en revenant en arrière, en pensant autrement en faisant des tests aussi beaucoup. Nous avons aussi apporté d'autres idées au projet, des petites touches personnelles en plus des objectifs du cahier des charges, ce qui est important.

Mais pour nous, notre objectif principal, était de découvrir et d'apprendre à travailler avec de nouvelles technologies, et pour ce point, ce projet nous a comblé.

Un des objectifs que nous nous étions aussi fixés était de respecter au mieux le planning défini préalablement. Au début du projet, nous avons défini un planning qui a subi à la première séance de nombreux changements. En effet nous sous-estimions l'importance et le temps à mettre à disposition pour certaines tâches surtout au niveau de l'analyse et de la conception, nous étions peut-être trop enthousiastes à l'idée de commencer l'implémentation. Nos superviseurs nous ont donc invités à revoir notre planning. Après ces modifications, tout au long du projet, le planning à vraiment bien été respecté, et ceci jusqu'à la fin, la partie implémentation ayant peut-être un peu débordé sur la partie des tests afin de peaufiner notre application, ce qui est compréhensible. Mais dans l'ensemble, pour notre tout premier projet dans cette école, nous sommes contents du résultat.

6.2 Perspectives futures

Virtual Memory offre une multitude de perspectives futures. Comme nous l'avons cité précédemment, au niveau des améliorations, il y a de bonnes idées et beaucoup d'autres idées peuvent voir le jour autour d'un projet de ce type.

Ce qui est aussi intéressant, c'est que lorsque nous avons fait nos tests utilisateurs, nous avons demandé à ces personnes de donner une idée de l'utilité de cette application dans leur profession et nous avons pu voir que les perspectives futures de ce projet étaient énormes.

Actuellement, cette application est un prototype, mais avec quelques améliorations, nous avons espoir que le Virtual Memory pourra peut-être être commercialisé afin d'être utilisé dans plusieurs domaines. Comme nous l'avons défini en début de projet, cette application pourrait servir dans le domaine de la scolarité dans certains cours, elle pourrait aussi être utilisée dans les ressources humaines afin que ces dernières fassent passer des tests à leurs nouvelles recrues, il faut laisser parler son imagination !

Maintenant, nous espérons que des autres personnes continuent à travailler sur ce projet afin que son avenir soit fructueux.

6.3 Conclusion de Loïc Dufresne

La réalisation du Virtuel Memory fut une belle expérience. Ce premier gros projet que nous réalisons dans notre formation est très concluant pour moi. En effet, j'ai pu découvrir et apprendre à travailler avec de nouvelles technologies, ce qui est pour moi le point le plus important et intéressant dans un projet, sortir un peu de la routine.

Un point qui m'a surpris tout au long de la réalisation de ce projet est le changement. Par exemple, si nous comparons la première version des maquettes définies au début de la conception avec celles présentées dans ce rapport ou encore avec l'application finale, il y a un monde ! La différence est énorme. J'ai pu voir que c'est un réalisant le projet, en travaillant dessus, que des idées naissent sans cesse, ce qui apporte des modifications et des changements.

Au finale, toutes les fonctionnalités définies dans le cahier des charges sont réalisées, mais une multitude d'autres idées nous a permis d'améliorer ces fonctionnalités et même d'apporter notre petit plus pour l'application, ce qui est important. Malheureusement, comme nous l'avons cité, nous ne pouvons pas tout réaliser, et ce n'est pas l'envie qui manque. Les améliorations et nouvelles idées sont infinies pour ce genre de projet, et c'est cela qui rend ces projets vraiment intéressants.

Maintenant, j'espère que le Virtual Memory aura de l'avenir et que des personnes continuent à travailler dessus afin de, pourquoi pas, en faire un produit et le commercialiser ! Ce serait quand même le but et un bel objectif.

En tout cas, j'en ressors que des points positifs, ce fut une belle expérience et je ne regrette pas d'avoir choisi ce projet.

Merci à toutes les personnes qui nous ont soutenues et aidées, et merci à mon collègue Nicolas pour avoir partagé avec moi tout ce travail.

6.4 Conclusion de Nicolas Stulz

La réalisation de ce premier grand projet IT a été très éducative pour moi, autant sur le plan gestion de projet que sur l'apprentissage de nouvelles technologies.

Je remercie le mandant pour les technologies imposées. Spring Boot permet d'implémenter une API REST très facilement. L'implémentation du Frontend avec Angular, m'a permis de découvrir ce Framework puissant et d'apprendre le Type Script.

La recherche de base de données m'a permis de connaître d'autre type de SGBD que le SQL. La découverte de Neo4j était passionnante, même si nous avons exploité cette technologie orientée Graph que très simplement, mais juste au besoin de l'application. Pour l'interaction entre le Backend et la base de données, nous ne comprenions pas l'API Spring Data Neo4j et avons utiliser l'API Java driver Neo4j pour interagir avec la base de données. Utiliser SpringData Neo4j nous aurait un peu facilité la tâche pour les requêtes simples. Mais nous avons fait toutes les requêtes à la main avec l'autre API.

Je remercie nos deux experts pour leurs accompagnements tout au long du projet ainsi que pour leurs précieux conseils. De plus, leurs recommandations d'utiliser Latex pour la rédaction du rapport m'ont permis enfin de m'y mettre.

Dans un premier temps, je voyais notre quizz beaucoup plus complexe, puis nous avons simplifié la chose au maximum afin de réduire les risques et de pouvoir fournir un premier prototype fonctionnel, ce qui est le cas.

Enfin, je suis très content du projet en lui-même et du résultat final. Je pense quand même que ce prototype devrait à l'avenir être amélioré, par exemple en sécurisant l'accès à l'administration, en créant une autre interface d'administration pour les paramètres du quizz comme le nombre de bonnes réponses et de réponses totales et aussi de pouvoir choisir si les fausses réponses générées aléatoirement proviennent de la même catégorie afin que la liste des réponses soit plus facilement cohérente et enfin de choisir si l'on lie les entités dans un sens ou les deux. En les liant que dans un sens, les entités ne seraient pas automatiquement membre d'un quizz alors que dans cette implémentation une entité liée avec une autre force que l'une soit une des bonnes réponses potentielles d'un quizz et inversement.

6.5 Déclaration d'honneur

Nous, soussignés, Loïc Dufresne et Nicolas Stulz, déclarons sur l'honneur que le travail rendu est le fruit d'un travail personnel. Nous certifions ne pas avoir eu recours au plagiat ou à toutes autres formes de fraudes. Toutes les sources d'information utilisées et les citations d'auteur ont été clairement mentionnées.

Fribourg, le 1 février 2018

7 Figures et Tables

7.1 Liste des figures

8 Lexique

API	Application Programming Interface
CSS	Cascading Style Sheets
db4o	DataBase For Objects
HEIA-FR	Haute école d'ingénierie et d'architecture de Fribourg
HES-SO	Haute École spécialisée de Suisse occidentale
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
NASA	National Aeronautics and Space Administration
noSql	SGBD s'écartant du paradigme des bases relationnelles
ReQL	RethinkDB query language
REST	Representational State Transfer
SGBD	Système de gestion de base de données
SGBDOO	Système de gestion de base de données orienté objet
STS	Spring Tool Suite

9 Bibliographie

[1] Angular Crew. Browser support. https://angular.io/guide/browser-support, 2017. [En ligne; Page disponible le 19-novembre-2017].

[2] Larry. 12 benefits of a responsive web design. https://www.webdesign.org/12-benefits-of-a-responsive-web-design.22565.html, 2015. [En ligne; Page disponible le 12-novembre-2017].

[3] Red Wolf. Web design and marketing. http://redwolfmarketing.uk, inconnu. [En ligne; Page disponible le 22-novembre-2017].

[4] Team Bootstrap. Browsers and devices. https://getbootstrap.com/docs/4.0/getting-started/browsers-devices/, 2017. [En ligne; Page disponible le 22-novembre-2017].

[5] Nicolas Hachet. L'architecture rest expliquée en 5 règles. https://blog.nicolashachet.com/niveaux/confirme/larchitecture-rest-expliquee-en-5-regles/, 2012. [En ligne; Page disponible le 30-octobre-2017].

[6] Vinay Sahni. Best practices for designing a pragmatic restful api. http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api, 2015. [En ligne; Page disponible le 08-décembre-2017].

[7] www.wikipedia.org. Base de données orientée texte. https://fr.wikipedia.org/wiki/Base_de_données_orientée_texte, 2016. [En ligne; Page disponible le 4-novembre-2017].

[8] Michaël Figuière. Nosql europe : Bases de donnÉes orientÉes documents et mongodb. https://blog.xebia.fr/2010/04/30/nosql-europe-bases-de-donnees-orientees-documents-et-mongodb/, 2009. [En ligne; Page disponible le 4-novembre-2017].

[9] RethinkDB. Installing rethinkdb. https://www.rethinkdb.com, 2017. [En ligne; Page disponible le 26-octobre-2017].

[10] RethinkDB team. The realtime javascript backend. http://horizon.io/, inconnu. [En ligne; Page disponible le 26-octobre-2017].

[11] RethinkDB. The open-source database for the realtime web. https://www.rethinkdb.com/docs/install/, 2017. [En ligne; Page disponible le 25-octobre-2017].

[12] Benjamin Sanchez. Découvrez rethinkdb et entrez dans le monde des bases de données en temps réel. http://http://www.blogduwebdesign.com/presentation-ressource/Rethink-db-base-de-donnee-asynchrone-realtime-json/2199, 2016. [En ligne; Page disponible le 25-octobre-2017].

[13] kureikain. Why learn rethinkdb? https://leanpub.com/simplyrethinkdb/read, inconnu. [En ligne; Page disponible le 27-octobre-2017].

[14] RethinkDB team. Ten-minute guide with rethinkdb and java. https://www.rethinkdb.com/docs/guide/java/, inconnu. [En ligne; Page disponible le 26-octobre-2017].

[15] wikipedia.org. Base de données hiérarchique. https://fr.wikipedia.org/wiki/Base_de_données_hiérarchique, 2016. [En ligne; Page disponible le 4-novembre-2017].

[16] Margaret Rouse. Base de données relationnelle. http://www.lemagit.fr/definition/Base-de-donnees-relationnelle, 2014. [En ligne; Page disponible le 4-novembre-2017].

[17] wikipedia. Db4o. https://fr.wikipedia.org/wiki/DataBase_For_Objects, 2017. [En ligne; Page disponible le 4-novembre-2017].

[18] antoyo. Base de données orientée objet. https://openclassrooms.com/forum/sujet/base-de-donnees-orientee-objet-39149, 2011. [En ligne; Page disponible le 6-novembre-2017].

[19] Guillaume Serries. 5 points pour mieux comprendre les bases de données graph. http://www.zdnet.fr/actualites/5-points-pour-mieux-comprendre-les-bases-de-donnees-graph-39839720.htm, 2016. [En ligne; Page disponible le 7-novembre-2017].

[20] GEEKO. Dossier : Quand choisir mysql ou nosql ? http://www.skilly.com/mag/mysql-nosql-que-choisir/, 2013. [En ligne; Page disponible le 4-novembre-2017].

[21] DB-Engines. Db-engines ranking. http://img-0.journaldunet.com/A2lgJJy6aW_lTc9LgBP6rkfN0cU=/1080x/smart/cd89934622884e129263137382b14ef4/ccmcms-jdn/10588111.jpg, 2017. [En ligne; Page disponible le 13-novembre-2017].

[22] Spring Tool Suite. Why learn rethinkdb? https://spring.io/tools, 2017. [En ligne; Page disponible le 27-octobre-2017].

[23] JetBrains. Enjoy productive java. https://www.jetbrains.com/idea/, 2017. [En ligne; Page disponible le 26-octobre-2017].

[24] Dmitry Jemerov. Intellij and webstorm combined together is it possible? https://intellij-support.jetbrains.com/hc/en-us/community/posts/206241279-IntelliJ-and-webstorm-combined-together-is-it-possible-, 2017. [En ligne; Page disponible le 27-octobre-2017].

[25] Guillaume. Pourquoi adopter bootstrap pour dÉvelopper son site ? http://www.escaledigitale.com/blog/pourquoi-adopter-bootstrap-pour-developper-son-site, 2015. [En ligne; Page disponible le 22-novembre-2017].

[26] Team Bootstrap. Grid system. http://getbootstrap.com/docs/4.0/layout/grid/, 2017. [En ligne; Page disponible le 22-novembre-2017].

[27] AFG. Spring boot, késako ? http://www.groupeafg.com/spring-boot-kesako/, 2015. [En ligne; Page disponible le 26-octobre-2017].

[28] Fernando Ferreira. Applications spring boot : ses avantages. https://blog.syloe.com/avantages-des-applications-spring-boot/, 2017. [En ligne; Page disponible le 26-octobre-2017].

[29] Daniel Woods. A la découverte des micro-frameworks : Spring boot. https://www.infoq.com/fr/articles/microframeworks1-spring-boot, 2014. [En ligne; Page disponible le 26-octobre-2017].

[30] Grégory Le Bonniec. 10 raisons de se mettre à spring boot (1ère partie). http://blog.ellixo.com/2015/06/08/10-raisons-de-se-mettre-a-Spring-Boot-1ere-partie.html, 2015. [En ligne; Page disponible le 26-octobre-2017].

[31] William Koza. Apprendre à programmer avec le framework angular. http://wkoza.developpez.com/tutoriels/angular/angular-cli/, 2016. [En ligne; Page disponible le 26-octobre-2017].

[32] Infinite. Angular – débuter avec angular-cli. https://www.infinitestudio.fr/2017/01/03/angular-2-debuter-avec-angular-cli/, 2017. [En ligne; Page disponible le 26-octobre-2017].

[33] www.ordinateur.cc. Avantages et inconvénients d'une base de données modèle relationnel. http://www.ordinateur.cc/Logiciel/Logiciel-de-base-de-données/115744.html, inconnu. [En ligne; Page disponible le 4-novembre-2017].

[34] wikipedia.org. Avantages et inconvénients d'une base de données modèle relationnel. https://fr.wikipedia.org/wiki/Base_de_données_orientée_documents, 2017. [En ligne; Page disponible le 4-novembre-2017].

[35] Next Decision. Editeur de base de donnÉes nosql not only sql. http://www.next-decision.fr/les-editeurs/stockage/mongo-db, 2015. [En ligne; Page disponible le 4-novembre-2017].

[36] jacklee8902. Mongodb tutorial. http://www.w3ii.com/fr/mongodb/mongodb_advantages.html, 2017. [En ligne; Page disponible le 4-novembre-2017].

[37] developpez.com. Cours complet pour apprendre les systèmes de gestion de bases de données. https://sgbd.developpez.com/tutoriels/cours-complet-bases-de-donnees/?page=

bases-de-donnees-reseaux-et-hierarchiques#LIX-4, inconnu. [En ligne; Page disponible le 4-novembre-2017].

[38] Timothe Fulcrand. Etude comparative bdd relationnelle versus nosql. https://blog.talanlabs.com/etude-comparative-bdd-relationnelle-versus-nosql/, 2015. [En ligne; Page disponible le 4-novembre-2017].

10 Annexes

Les annexes citées ci-dessous sont imprimées à part du rapport et fournies avec la version papier du rapport. Pour la version PDF du rapport, les annexes citées ci-dessous sont disponibles sur Multidoc et sur le CD/DVD.

1. Le manuel d'installation de l'environnement
2. Les tests fonctionnels réalisés par les utilisateurs
3. Le cahier des charges

Nous n'avons pas mis le code source de notre application en annexe, car il est trop volumineux, mais il est disponible sur le CD/DVD ou sur la GIT:
`https://gitlab.forge.hefr.ch/nicolas.stulz/VirtualMemory`

Tous les documents, PV, plannings, manuels, diagrammes, maquettes et autres informations relatives au projet sont disponibles sur la FORGE:
`https://forge.hefr.ch/projects/virtual_memory`

10.1 Versions

Ces versions sont celles avec lesquelles nous avons réalisé notre projet et ce sont les dernières disponibles au 31 janvier 2018:

- **Neo4j Desktop:** 1.0.10 et 1.0.11
- **Neo4j:** 3.3.1 et 3.3.2
- **IntelliJ:** ULTIMATE 2017.3
- **NodeJS:** 6.12.3
- **NPM:** 5.6.0
- **Apache Maven:** 3.3.9

Pour plus d'informations, veuillez vous référer au manuel d'installation disponible en annexe.

10.2 Contenu du CD/DVD

```
VIRTUAL MEMORY
   .
   |-- base_de_donnees
   |   `-- neo4j_database.txt
   |-- cahier_des_charges
   |   `-- cahier_des_charges_111017.pdf
   |-- code_source
   |   `-- VirtualMemory
   |       `--*
   |-- diagrammes
   |   `-- *
   |-- divers
   |   `-- schema_fonctionnement.png
   |-- etat_application
   |   `-- etat_application_300118.pdf
   |-- logo
   |   `-- virtualmemory.png
   |-- manuel_installation
   |   `-- manuel_installation_010218.pdf
   |-- maquettes
   |   `-- *
   |-- planning
   |   |-- planning_final_111017.pdf
   |   `-- planning_reel_010218.pdf
   |-- presentation
   |   `-- prensentation_031117.pptx
   |-- pv
   |   `-- *
   |-- rapport
   |   `-- rapport_virtualmemory_010218.pdf
   |-- README.txt
   `-- tests
       |-- fonctionnels
       |   `--tests_fonctionnels_300118.pdf
       `-- utilisateurs
           |--tests_utilisateurs_300118.pdf
           |--tests_utilisateurs_synthese_300118.pdf
           `--tests_utilisateurs_vierge_290118.pdf
```

10.3 Planning