



TER MASTER 1 MIAGE 2021



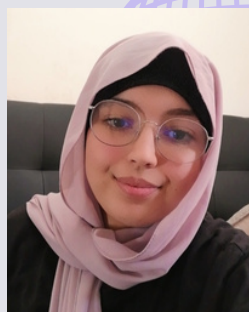
université
PARIS-SACLAY

PROJET QCM EN LIGNE

Réalisé par l'équipe :



MENGUELTI
HACENE



BNIK
BOUCHRA



PISANO
VINCENT

Sous la responsabilité de
DECLERCQ PHILIPPE

2021/2022

TABLE DES MATIÈRES

1. Conception du projet	3
A) Reformulation du sujet	3
B) Diagramme de cas d'utilisation	4
C) Description des cas d'utilisation	5
D) Diagramme de classes	8
E) Diagrammes de séquence et d'états	9
F) IHM et diagrammes de navigation	11
G) Outils et technologies	19
H) Diagramme de Gantt du Développement	20
2. Implémentation du projet	21
A) Les technologies choisies	21
B) Le framework sélectionné	23
C) Description des composants de l'application	26
D) Lien avec la partie conception	28
E) Exemple de fonctionnement de l'application	33
F) Code source des parties significatives	40
G) Le modèle de la base de données	46
H) Limites et évolutions possibles	49
I) Code source	49
J) Conclusion	50

1. Conception du projet

A) Reformulation du sujet

Notre équipe de conception a été mandatée par la société Sondix, spécialisée dans la conception, la mise à disposition et l'analyse de Questionnaires à Choix Multiples (QCM) auprès des internautes.

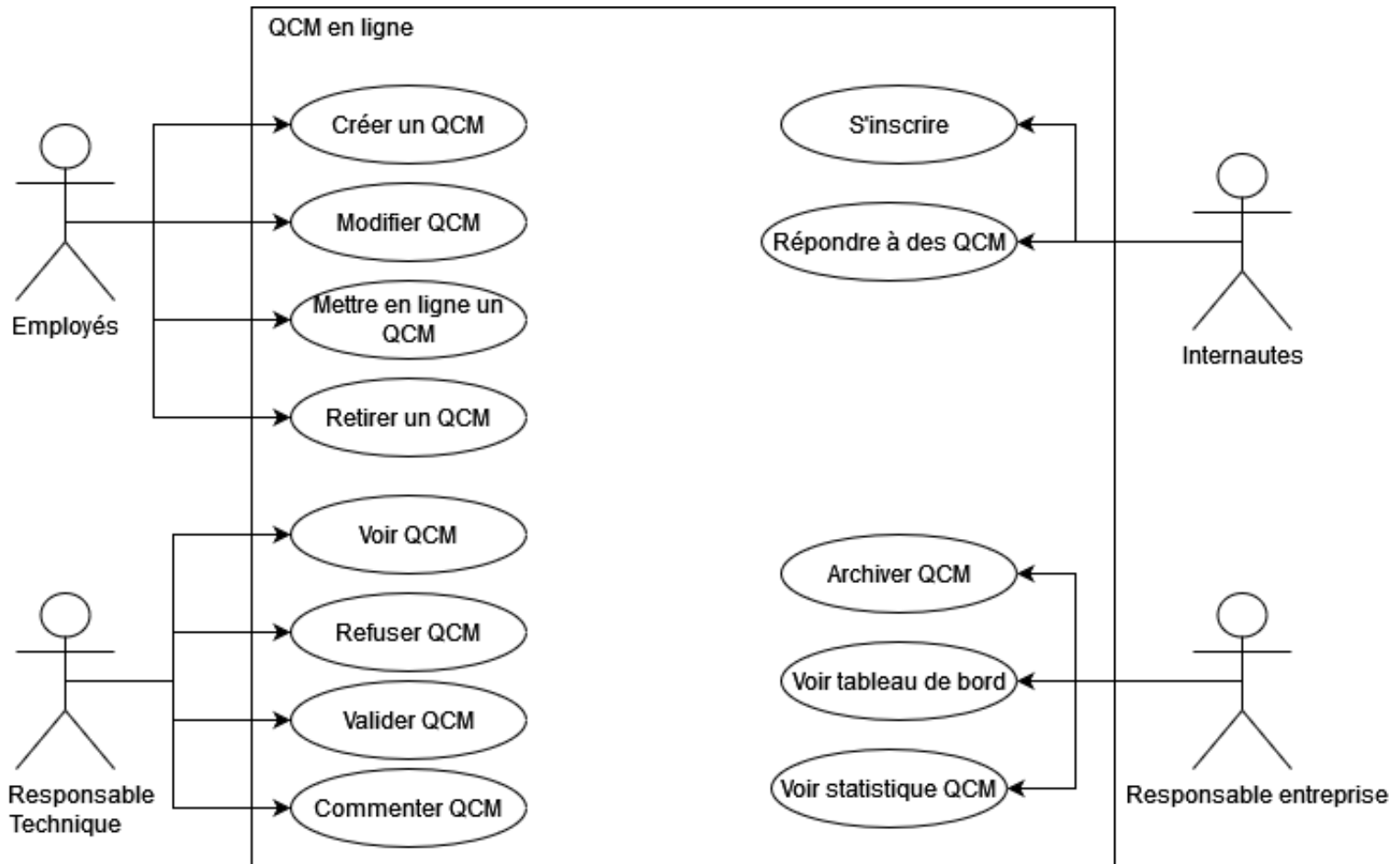
Cette société nous a contacté dans l'objectif de développer un nouveau système informatique qui permettra de créer et de mettre en ligne de manière générique des QCM en ligne. Ces QCM seront accessibles auprès d'utilisateurs qui pourront y répondre et à Sondix pour les analyser.

Il a été convenu que cette application soit accessible depuis le web. Les utilisateurs de l'application web auront accès à une diversité de QCM, chacun composé de plusieurs questions et d'au moins 2 réponses pour chacun.

Au sein du système que nous souhaitons mettre en place, en concordance avec les conditions émises par Sondix, nous identifions 4 utilisateurs distincts à ce point de notre conception.

- Les internautes peuvent s'inscrire puis se connecter à l'application pour enfin pouvoir répondre aux différents QCM disponibles en ligne pour gagner des points de fidélité.
- Les employés de la société ont pour responsabilité la création, la modification, la suppression ainsi que de la mise en ligne d'un QCM.
- Le responsable technique est chargé de la validation ou du refus de mise en ligne d'un questionnaire.
- Le responsable de l'entreprise peut consulter le tableau de bord c'est-à-dire l'analyse détaillée des activités des internautes sur les QCM. De plus, il peut retirer un QCM mis en ligne.

B) Diagramme de cas d'utilisation



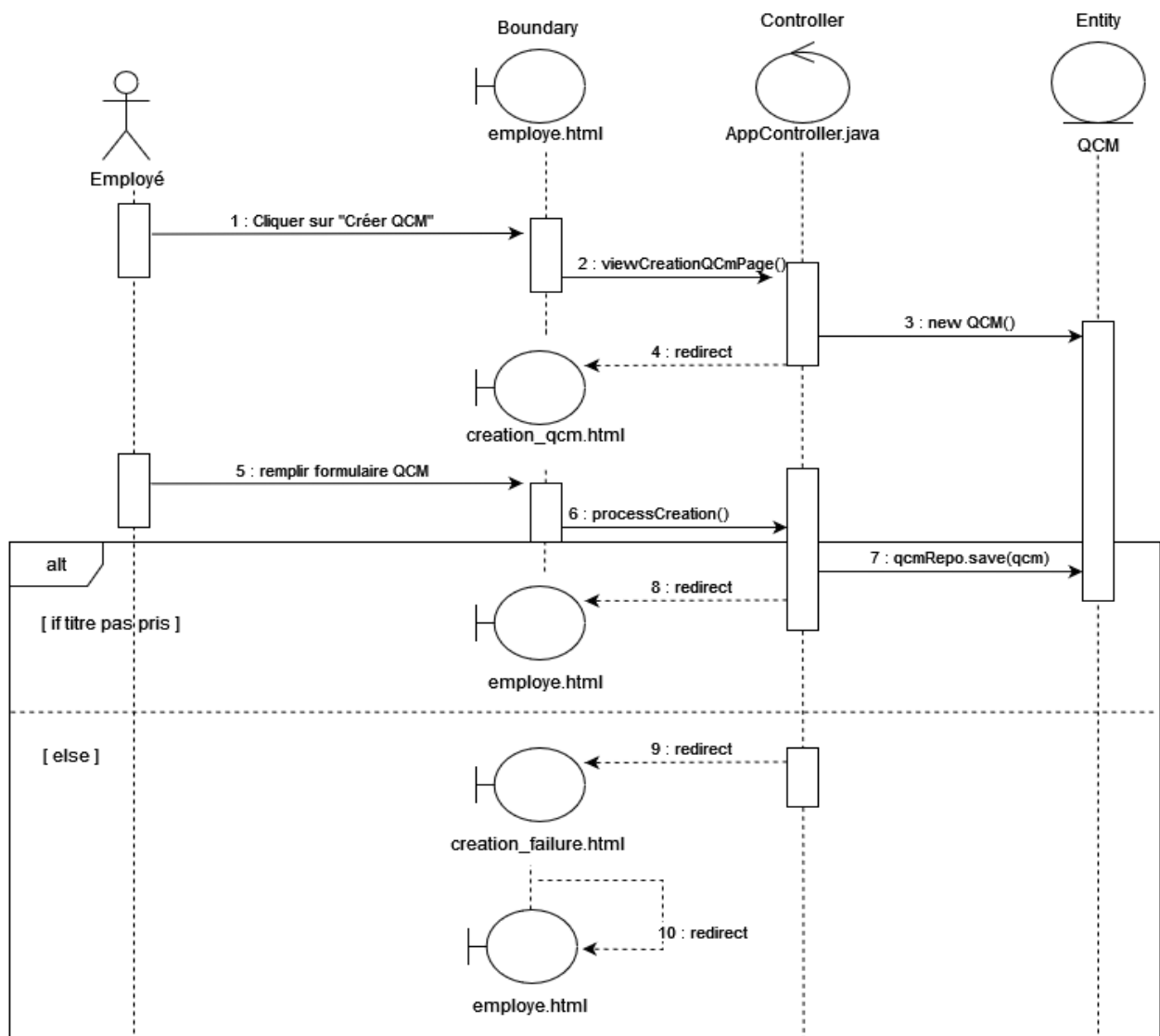
Précondition : Pour accéder à l'application de QCM en ligne il faut impérativement s'authentifier.

C) Description des cas d'utilisation

Nous avons identifié 4 utilisateurs dans notre système : “Employés”, “Internautes”, “Responsable entreprise” et “Responsable Technique”.

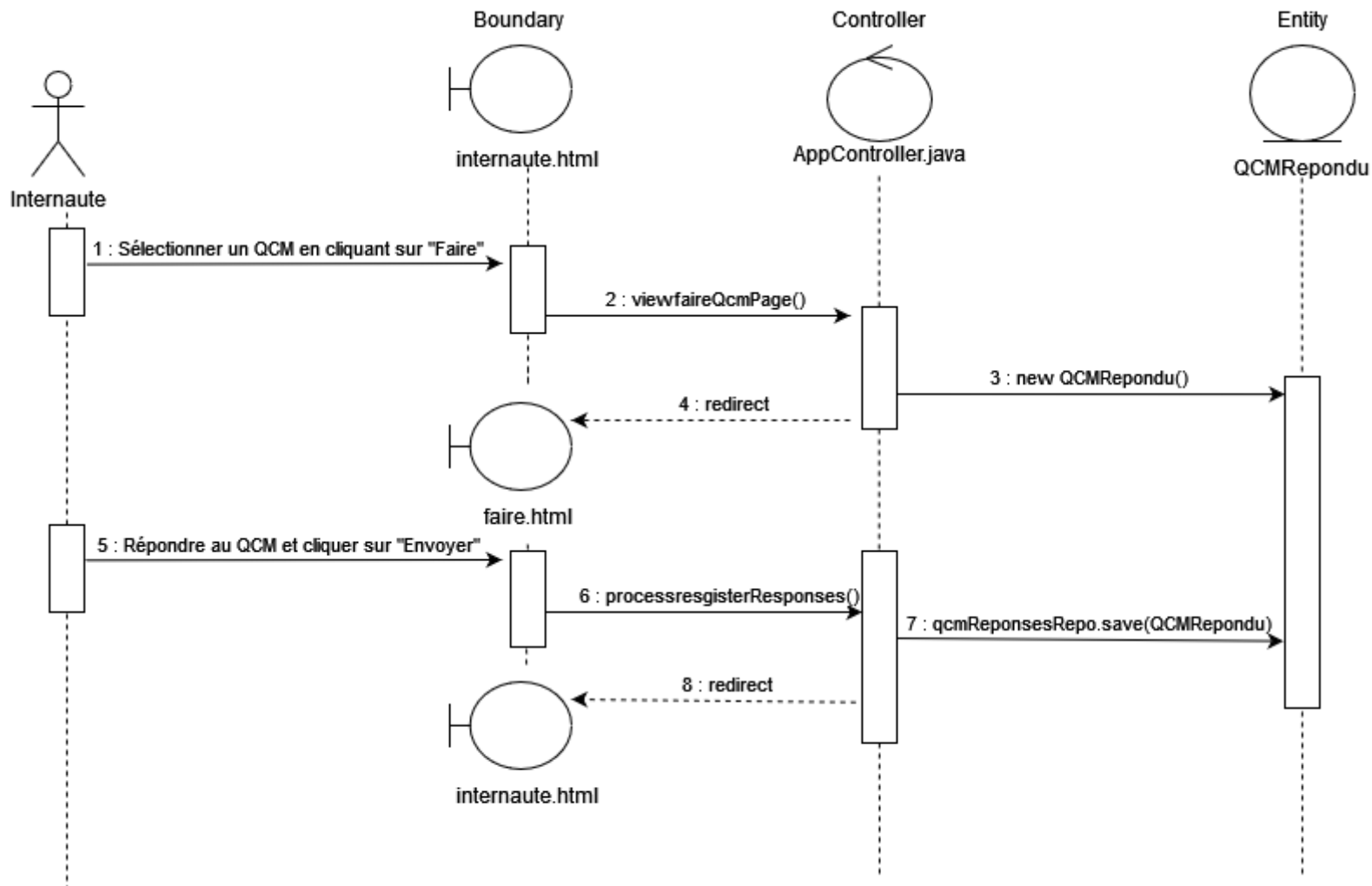
Cas d'utilisation que l'on a distingué :

- Créer un QCM : Un “Employé” est chargé de créer un QCM en choisissant un titre et un thème à celui-ci. De plus, il doit remplir un formulaire composé de 10 champs pour les questions et dont chaque question peut avoir entre 2 à 4 réponses.



- Modifier un QCM : Un “Employé” peut modifier un QCM, soit en changeant l’intitulé des questions, des réponses, le titre ou même le thème.
- Mettre en ligne un QCM : Lorsqu’un QCM a entièrement été réalisé, un “Employé” peut mettre en ligne un QCM, cette action sous-entend qu’il émet le QCM à la validation par un “Responsable Technique”
Une fois validé, le QCM apparaîtra en ligne et sera disponible pour tous les “Internauts”.
- Retirer un QCM : Un “Employé” peut retirer un QCM du site.
- Valider un QCM : Le “Responsable Technique” peut valider ou invalider la mise en ligne d’un QCM qui a été demandé par un “Employé”. Le “Responsable Technique” doit en plus de cela préciser un commentaire pour expliquer sa décision.
Une fois validé, le QCM apparaîtra en ligne et sera disponible pour tous les “Internauts”.
- S’authentifier : Afin d’accéder à l’application, quel que soit les droits que l’utilisateur possède, il doit s’authentifier, c’est-à-dire fournir son identifiant et son mot de passe. Avant de réaliser cette étape pour la toute première fois, les utilisateurs devront passer par l’étape S’inscrire.
- S’inscrire : Un “Internaute” peut s’inscrire sur le site en y ajoutant les informations présentes dans le formulaire d’inscription afin de posséder un compte et de pouvoir répondre aux QCM.

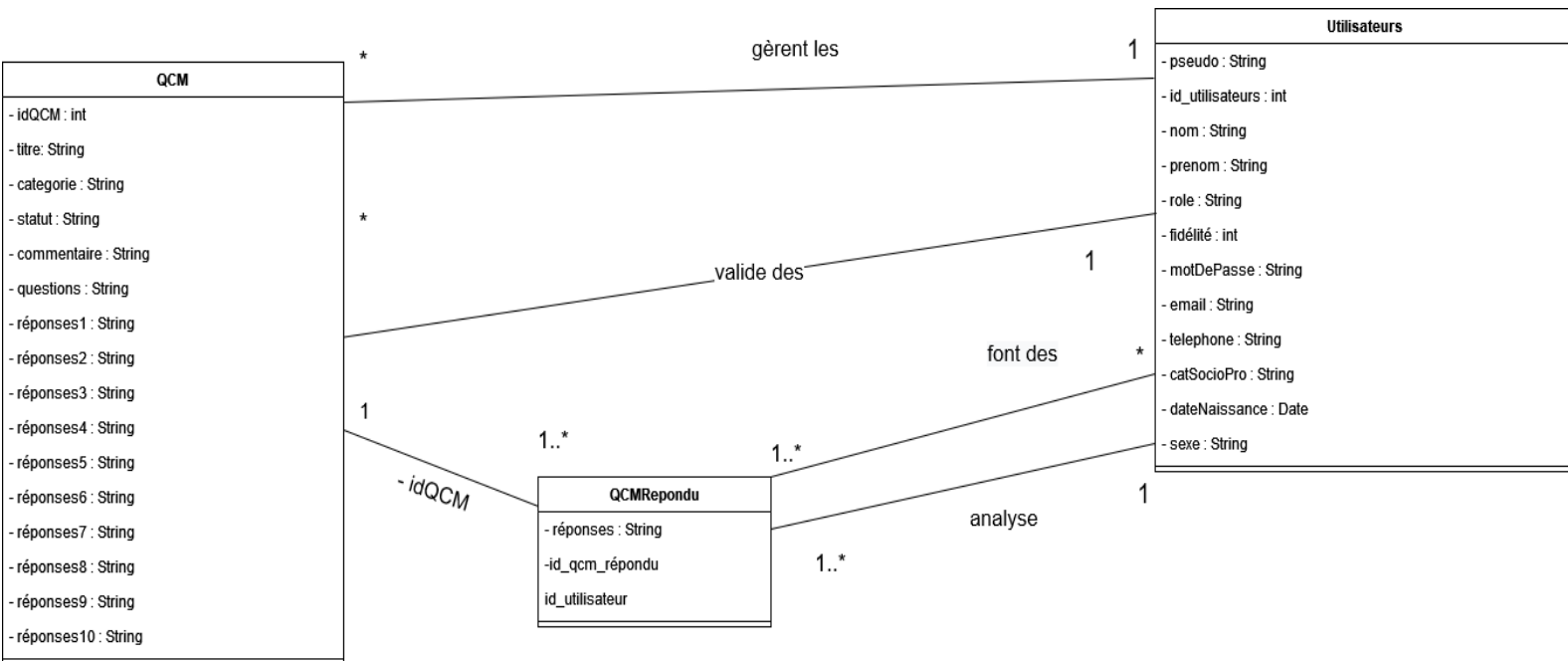
- Répondre à un QCM : Un "Internaute" une fois authentifié peut accéder aux QCM disponibles en ligne et ainsi répondre aux questions de ce dernier.



- Voir tableau de bord : Le "Responsable de l'entreprise" une fois authentifié pourra avoir accès à une vision globale de l'application mais aussi à une analyse de chacun des QCM s'il le souhaite.

D) Diagramme de classes

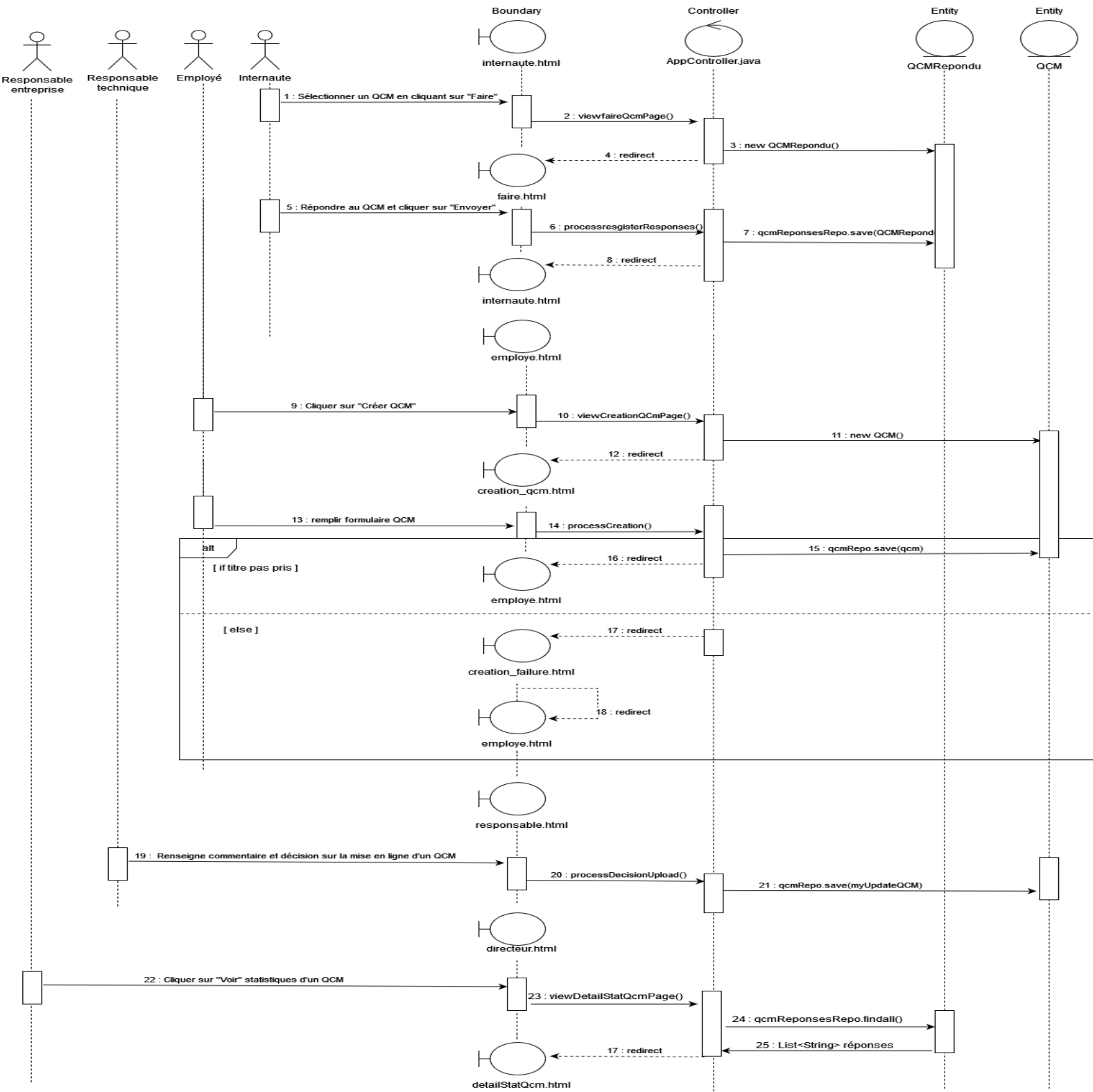
Ce diagramme de classe représente les entités présentes au sein de notre projet. Les 3 principales entités sont : Utilisateur, QCM et QCM Répondu. Donc, au sein de notre base de données, il y aura 3 tables principales.



Lien vers l'image de meilleur qualité : <https://ibb.co/Wtp48J2>

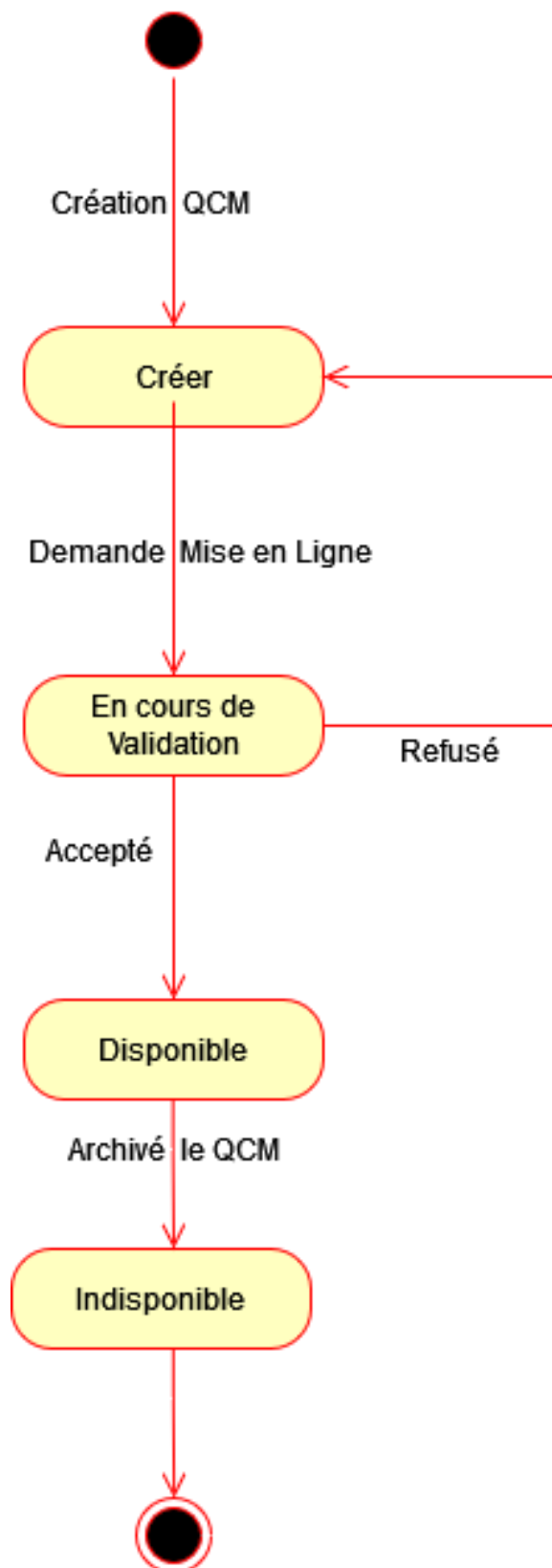
E) Diagrammes de séquence et d'états

1. Diagramme de séquence générale du système Sondix



Lien vers l'image de meilleur qualité : <https://ibb.co/3rjCvZk>

2. Diagramme d'états d'un QCM



F) IHM et diagrammes de navigation

Maquette de notre Interface Homme-Machine (IHM) :

IHM 1 : Page de connexion



Connexion

email

mot de passe

s'inscrire se connecter

Si on clique sur le bouton "s'inscrire" on arrivera sur l'IHM 2.

Si on clique sur le bouton "se connecter" et que notre compte est un compte d'internaute, alors on arrive sur l'IHM 3.

Si on clique sur le bouton "se connecter" et que notre compte est un compte d'employés, alors on arrive sur l'IHM 5.

Si on clique sur le bouton "se connecter" et que notre compte est le compte du responsable technique, alors on arrive sur l'IHM 8.

Si on clique sur le bouton "se connecter" et que notre compte est le compte du responsable de l'entreprise, alors on arrive sur l'IHM 9.

IHM 2 : Page d'inscription

[retour](#)

Inscription

▼

▼

Si on clique sur retour ou sur “Valider” et que tous les champs sont remplis correctement alors nous sommes redirigés sur l’IHM 1.

IHM 3 : Choix de QCM

Déconnexion

Choisir QCM

QCM	Thème	Etat
Dragon ball Z	Anime	Déjà fait
Villes	Culture générale	A faire
James Bond	Cinéma	Déjà fait

Si on clique sur le bouton “déconnexion” on est redirigé sur l’IHM 1.

Si on clique sur le bouton “A faire” on est redirigé sur l’IHM 4.

IHM 4 : Répondre au QCM

Titre du QCM

Thème

Question 1	<input checked="" type="checkbox"/> Réponse 1	<input type="checkbox"/> Réponse 2	<input checked="" type="checkbox"/> Réponse 3	<input type="checkbox"/> Réponse 4
Question 2	<input type="checkbox"/> Réponse 1	<input checked="" type="checkbox"/> Réponse 2	<input type="checkbox"/> Réponse 3	<input type="checkbox"/> Réponse 4
Question 3	<input type="checkbox"/> Réponse 1	<input type="checkbox"/> Réponse 2	<input checked="" type="checkbox"/> Réponse 3	<input type="checkbox"/> Réponse 4
Question 4	<input type="checkbox"/> Réponse 1	<input type="checkbox"/> Réponse 2	<input checked="" type="checkbox"/> Réponse 3	<input type="checkbox"/> Réponse 4
Question 5	<input type="checkbox"/> Réponse 1	<input checked="" type="checkbox"/> Réponse 2	<input type="checkbox"/> Réponse 3	<input checked="" type="checkbox"/> Réponse 4
Question 6	<input type="checkbox"/> Réponse 1	<input type="checkbox"/> Réponse 2	<input type="checkbox"/> Réponse 3	<input checked="" type="checkbox"/> Réponse 4
Question 7	<input type="checkbox"/> Réponse 1	<input checked="" type="checkbox"/> Réponse 2	<input checked="" type="checkbox"/> Réponse 3	<input type="checkbox"/> Réponse 4
Question 8	<input checked="" type="checkbox"/> Réponse 1	<input checked="" type="checkbox"/> Réponse 2	<input type="checkbox"/> Réponse 3	<input type="checkbox"/> Réponse 4
Question 9	<input type="checkbox"/> Réponse 1	<input checked="" type="checkbox"/> Réponse 2	<input type="checkbox"/> Réponse 3	<input type="checkbox"/> Réponse 4
Question 10	<input checked="" type="checkbox"/> Réponse 1	<input type="checkbox"/> Réponse 2	<input type="checkbox"/> Réponse 3	<input type="checkbox"/> Réponse 4

Annuler

Envoyer

Si on clique sur le bouton “annuler” on est redirigé sur l’IHM 3.

Si on clique sur le bouton “envoyer” et que l’on a bien choisi au moins 1 réponse par question alors on est aussi redirigé sur l’IHM 3.

IHM 5 : Gestion des QCM

Déconnexion

Gestion des QCM

Créer QCM

QCM	Thème	Modification	Statut	Suppression
Dragon ball Z	Anime	modifier	disponible	supprimer
Ellon Musk	Espace	modifier	pas disponible	supprimer
James Bond	Cinéma	modifier	disponible	supprimer
Donald Trump	Politique	modifier	pas disponible	supprimer
Villes	Culture générale	modifier	disponible	supprimer

Si on clique sur le bouton “déconnexion” on est redirigé sur l’IHM 1.

Si on clique sur le bouton "modifier" on est redirigé sur l’IHM 6.

Si on clique sur le bouton “Créer QCM” on est redirigé sur l’IHM 7.

IHM 6 : Modification QCM

Modification QCM

Titre
Thème

Question 1	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 2	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 3	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 4	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 5	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 6	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 7	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 8	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 9	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 10	Réponse 1	Réponse 2	Réponse 3	Réponse 4

Annuler
Modifier

Si on clique sur le bouton “annuler” ou “modifier” et que les champs sont correctement remplis alors on est redirigé sur l’IHM 5.

IHM 7 : Création QCM

Création QCM

Question 1	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 2	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 3	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 4	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 5	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 6	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 7	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 8	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 9	Réponse 1	Réponse 2	Réponse 3	Réponse 4
Question 10	Réponse 1	Réponse 2	Réponse 3	Réponse 4

Annuler
Créer

Si on clique sur le bouton “annuler” ou “créer” et que les champs sont correctement remplis alors on est redirigé sur l’IHM 5.

IHM 8 : Validation QCM

Déconnexion

Validation

QCM	Thème	Commentaire	Avis
Dragon ball Z	Anime	Intéressant	Favorable
Ellon Musk	Espace	Développer plus	Défavorable
James Bond	Cinéma	Drôle	Favorable
Donald Trump	Politique	Trop sérieux	Défavorable
Villes	Culture générale	Bien	Favorable

Si on clique sur le bouton “déconnexion” on est redirigé sur l’IHM 1.

IHM 9 : Tableau de bord

Déconnexion

Tableau de bord

Nombre de QCM en ligne	3
QCM le plus populaire	Dragon ball Z
QCM le moins populaire	Villes
Total internautes inscrit	951
Total QCM répondu	753
Maximum point fidélité	300

QCM	Informations
Dragon ball Z	Détail
Villes	Détail
James Bond	Détail

Si on clique sur le bouton “déconnexion” on est redirigé sur l’IHM 1.

Si on clique sur un des boutons de “Détail” nous sommes redirigés vers l’IHM 10.

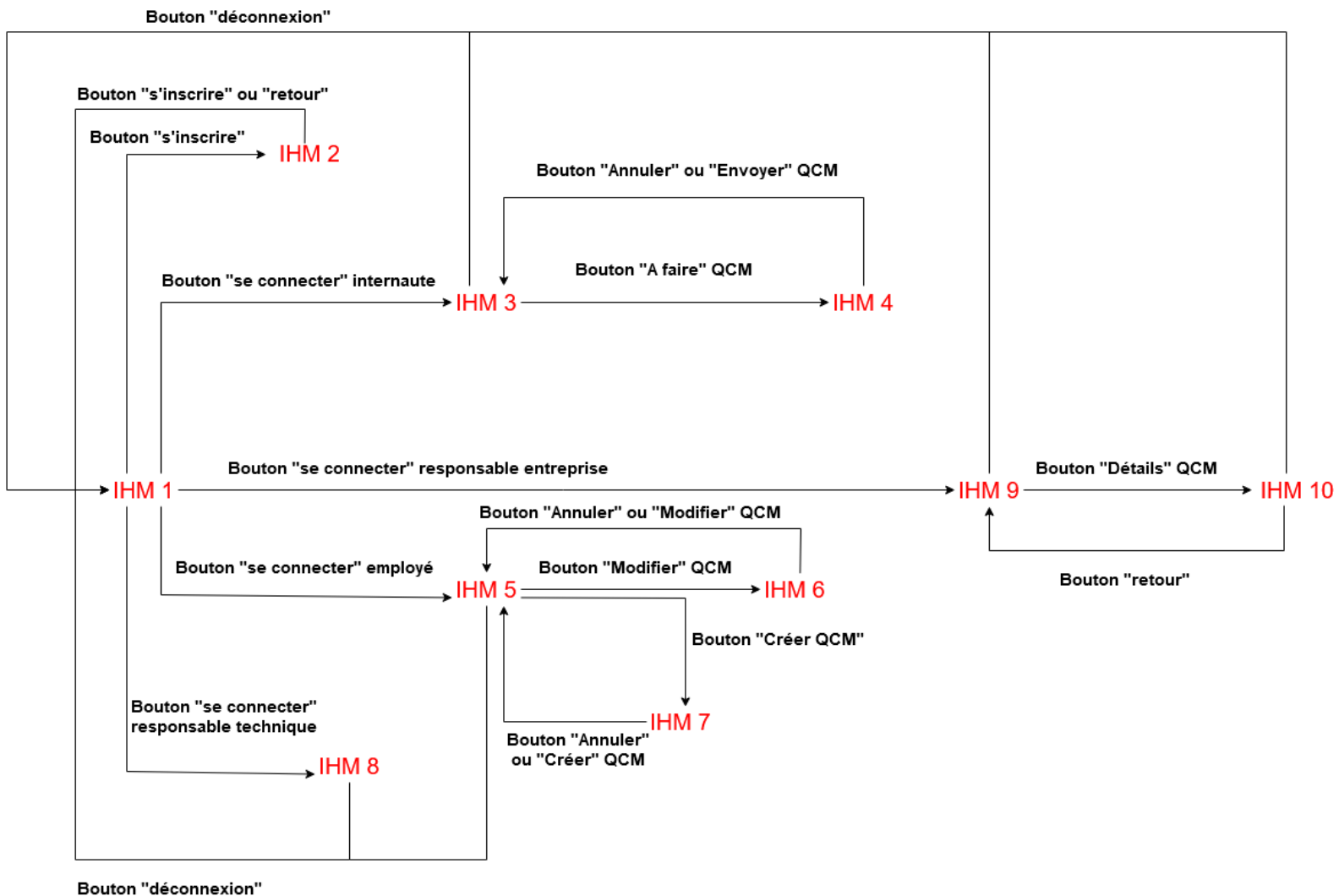
IHM 10 : Analyse QCM

QCM Dragon Ball Z Analyse des réponses								
Retour								
		Déconnexion						
Question 1	80%	Réponse 1	10%	Réponse 2	7%	Réponse 3	3%	Réponse 4
Question 2	20%	Réponse 1	20%	Réponse 2	50%	Réponse 3	10%	Réponse 4
Question 3	5%	Réponse 1	10%	Réponse 2	80%	Réponse 3	5%	Réponse 4
Question 4	99%	Réponse 1	0%	Réponse 2	1%	Réponse 3	0%	Réponse 4
Question 5	0%	Réponse 1	7%	Réponse 2	73%	Réponse 3	20%	Réponse 4
Question 6	10%	Réponse 1	80%	Réponse 2	5%	Réponse 3	5%	Réponse 4
Question 7	10%	Réponse 1	30%	Réponse 2	10%	Réponse 3	50%	Réponse 4
Question 8	0%	Réponse 1	19%	Réponse 2	1%	Réponse 3	80%	Réponse 4
Question 9	15%	Réponse 1	15%	Réponse 2	10%	Réponse 3	60%	Réponse 4
Question 10	50%	Réponse 1	15%	Réponse 2	30%	Réponse 3	5%	Réponse 4

Si on clique sur le bouton “déconnexion” on est redirigé sur l’IHM 1.

Si on clique sur le bouton “Retour” on est redirigé sur l’IHM 9.

Diagramme de Navigation :



Lien vers l'image de meilleur qualité : <https://ibb.co/8YsPqkz>

G) Outils et technologies

Dans le but de réaliser les éléments qui doivent être présents dans le rapport de conception, l'équipe a utilisé l'outil du nom de Draw.io qui est un logiciel gratuit de diagrammes multi-plateformes en ligne et de bureau.

Il permet de créer de nombreux organigrammes, diagrammes, schémas... Mais dans notre cas, nous nous sommes concentrés sur la partie UML notamment afin de réaliser nos diagrammes de : cas d'utilisation, de classes, séquences objet et systèmes, d'état, mais aussi pour l'IHM.

Concernant la réalisation planning sous la forme d'un diagramme de Gantt, nous avons utilisé Gantt Project qui est un logiciel libre de gestion de projet qui permet d'éditer un diagramme de Gantt.

Notre équipe a choisi de réaliser ce projet de QCM en Java en utilisant le Framework open source Spring afin de définir et de construire l'infrastructure de notre projet.

Au début, l'équipe a pensé à réaliser ce projet en Java afin d'améliorer nos compétences dans ce langage que nous avons appris l'an dernier. Cependant afin de rendre possible la mise en ligne de l'application, mais aussi dans le but d'apprendre de nouvelles compétences nous avons fait le choix d'utiliser Spring.

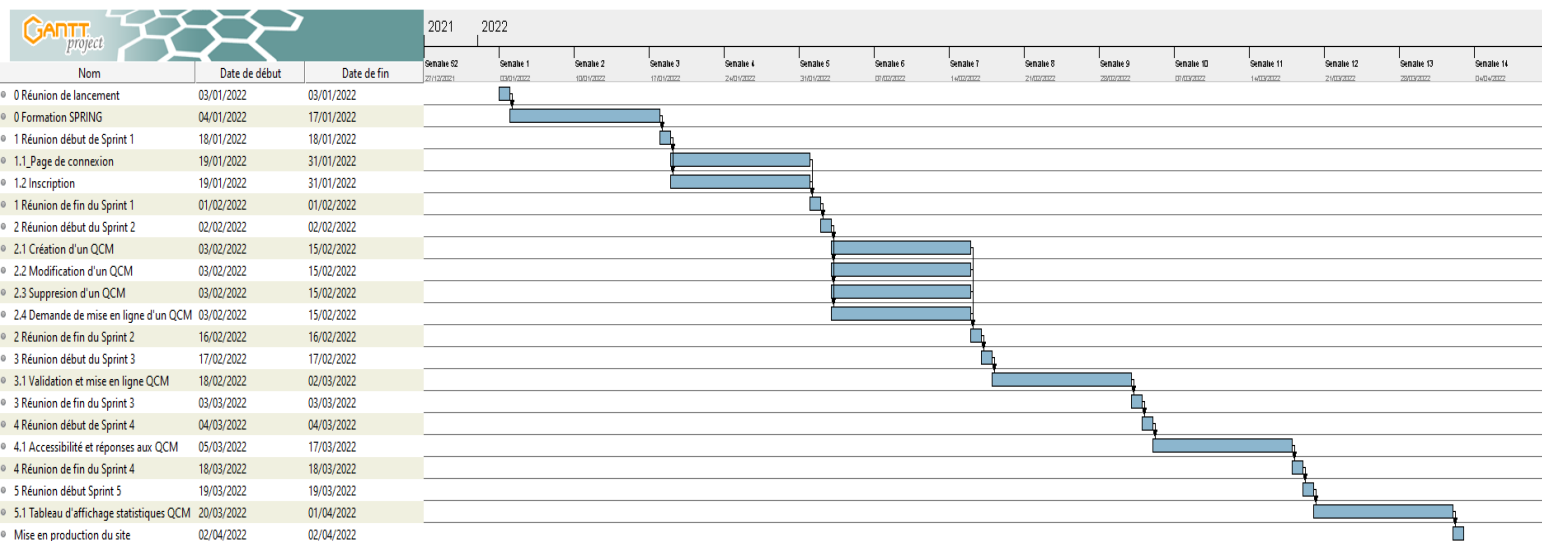
De plus, Spring est une des propositions faites pour les frameworks pouvant implémenter le MVC en plus de type struts, symfony et zend... Spring possède une grande flexibilité dans les fonctionnalités et les projets utilisés dans une application aussi bien au niveau de l'utilisation d'un serveur (Tomcat par exemple) ou d'une base de données (MySQL par exemple).

Mais ce n'est pas tout, ce framework simplifie le développement des applications Java grâce à son code source épuré et possède un temps d'adaptation minime.

L'équipe a choisi comme framework de développement Spring Tool Suite 4.

Enfin, concernant la base de données nous utiliserons une base de données MySQL avec le logiciel de gestion et d'administration de bases de données MySQL Workbench.

H) Diagramme de Gantt du Développement



Lien vers l'image de meilleur qualité : <https://ibb.co/mb3tsP3>

2. Implémentation du projet

A) Les technologies choisies

Dans le but de réaliser ce projet nous avons utilisé 2 technologies qui sont Spring et MySQL.



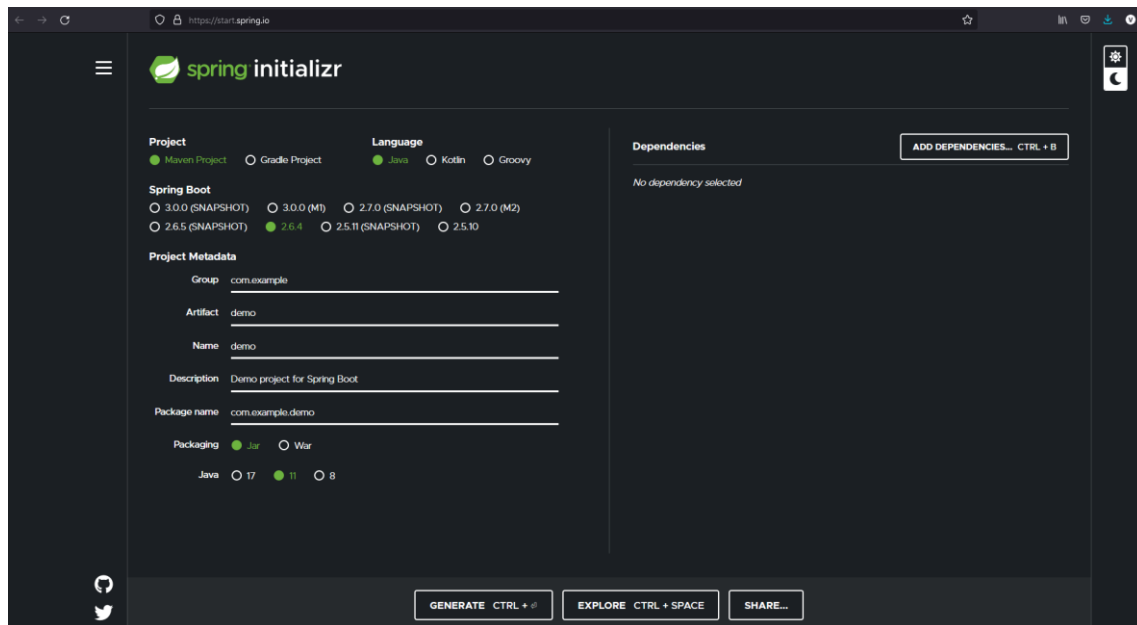
Spring est un framework Open Source très riche, parmi les plus réputés au monde. Il a pour but de construire et de définir l'infrastructure d'une application Java et d'en faciliter le développement et les tests.

Spring est considéré comme un projet Open Source. Il a été créé et développé par Rod Johnson et sa première version date du 25 juin 2003.

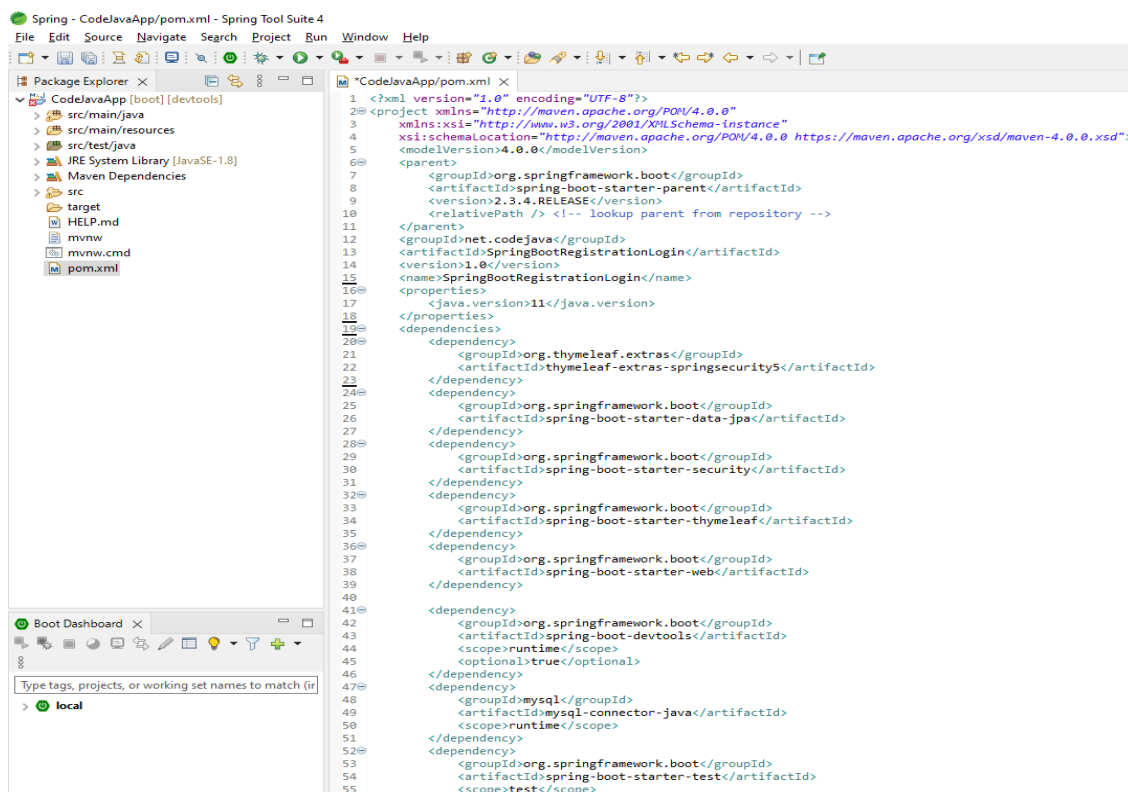
Le Spring offre de nombreuses fonctionnalités, comme la gestion transactionnelle, celle des exceptions Java DataBase Connectivity mais aussi et surtout un framework MVC (Modèle-Vue-Contrôleur). Celui-ci est très puissant, car il permet d'ajouter de multiples fonctionnalités Java.

Le Spring est un conteneur léger qui reste aussi malheureusement connu pour sa configuration plutôt complexe, longue et fastidieuse. Comme solution, les concepteurs ont décidé de faciliter le développement d'applications à travers le Spring Boot, qui allège le temps consacré au démarrage d'un projet.

Pour la réalisation de notre projet Sondix nous avons donc utilisé Spring Boot afin de réduire le temps de démarrage du projet. Pour cela nous sommes allés sur le site : <https://start.spring.io/> dans le but de créer le projet en utilisant la Version 11 de Java.



Le cœur de Spring repose sur un conteneur gérant l'injection des dépendances. Ces dernières peuvent être définies à travers des fichiers de configuration rédigés soit en XML ou en Java. Au sein de notre projet l'on peut voir ses dépendances dans le fichier pom.xml:



Après avoir comparé le Framework Spring avec d'autres Framework tels que zend, symfony et struts, l'équipe a décidé d'employer Spring afin d'améliorer nos compétences en Java mais aussi pour ajouter cette nouvelle compétence à notre CV.



Nous avons choisi comme système de gestion de bases de données relationnelles MySQL pour plusieurs raisons différentes. Dans un premier temps, MySQL est très rapide ce qui a été un critère de choix dans la sélection d'une base de données. De plus MySQL est très portable dans le sens où cela fonctionne aussi bien sur Unix, Windows ou Linux. Enfin le dernier critère qui nous a fait choisir MySQL par rapport à Oracle Database par exemple est le fait qu'il est gratuit et open source.

Pour notre projet nous avons donc installé MySQL Community Server et MySQL Workbench.

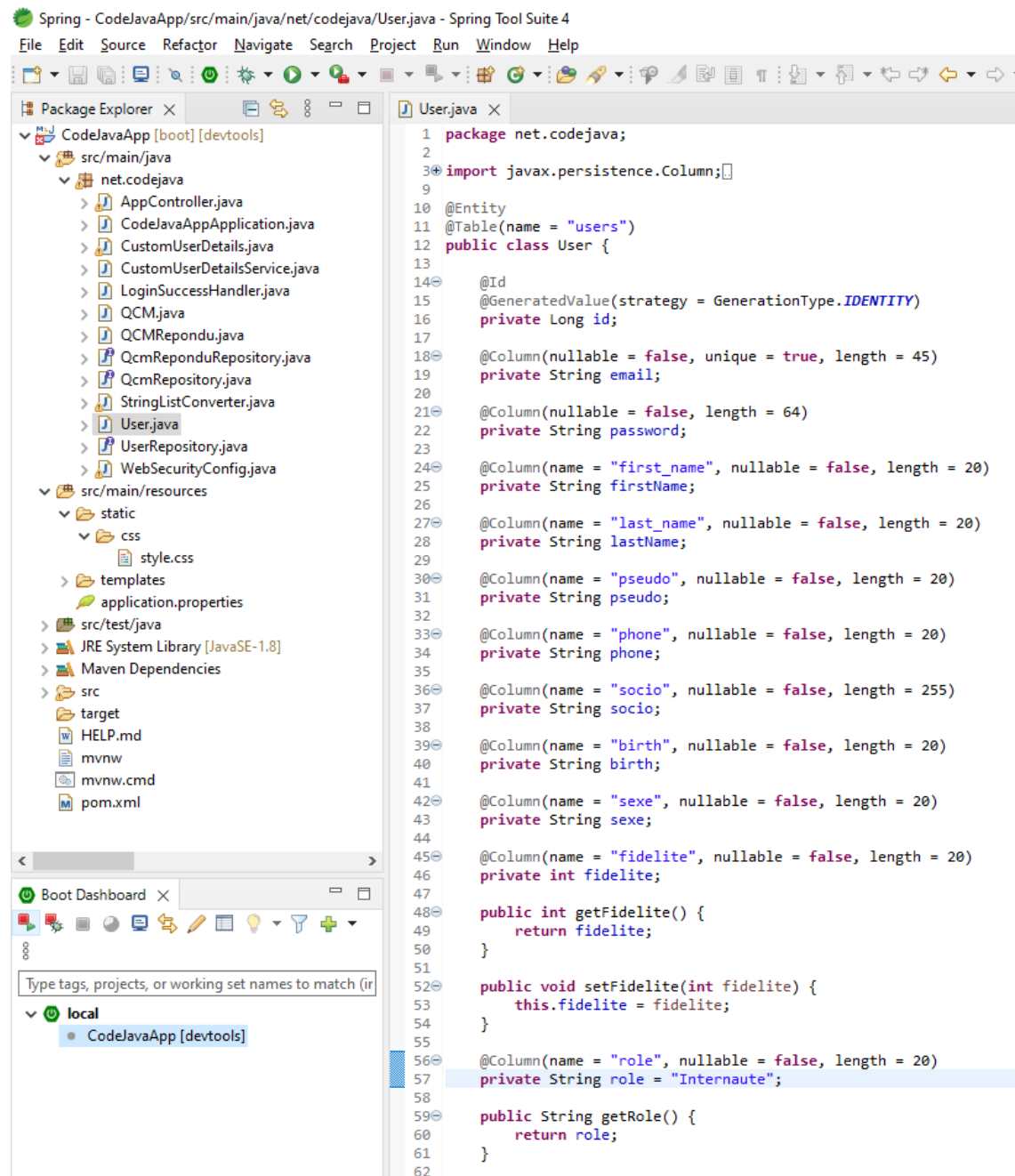
MySQL Workbench est un logiciel de gestion et d'administration de bases de données MySQL. Il permet via une interface graphique de créer, supprimer ou modifier des tables, des comptes utilisateurs, et d'effectuer toutes les opérations inhérentes à la gestion d'une base de données. Pour cela, Workbench doit être connecté à un serveur MySQL.

B) Le framework sélectionné

Après des discussions entre les membres du groupe et plusieurs tests de Framework, nous nous sommes mis d'accord pour utiliser Spring Tool Suite 4. La mise en place de ce Framework sur nos machines fut simple, pour cela nous avons suivi un tutoriel sur youtube d'un anglais. Après cela, nous sommes allés sur le site <https://spring.io/tools> pour télécharger le Jar correspondant à nos Système d'exploitation pour ensuite installer le Framework. Le seul pré-requis à l'installation était que tous les membres du groupe utilisent la même version de Java c'est-à-dire la version : java version 11.

Une fois installé, nous avons juste eu à importer le projet qui a été créé en utilisant Spring Boot sur le site <https://start.spring.io/>. Une fois le projet importé, le Framework va mettre quelques minutes pour l'ouvrir, le Build et vérifier qu'il n'y a aucune erreur. Une fois fini nous avons pu commencer à travailler.

Voici à quoi ressemble l'outil que nous utilisons :



En haut à gauche de l'image nous pouvons voir l'explorateur de package qui nous permet de naviguer dans le projet. Il est composé de 3 grandes parties :

- src/main/java : contient le package net.codejava ou sont présentes les différentes classes java du projet (les entités et contrôleurs).
- src/main/resources : contient les fichiers ressources. Dans ce dossier il y a 2 sous packages. Le package static contenant le css, javascript et aussi les images. Le package templates contenant les fichiers HTML (les vues).
- src/test/java : contient les classes Java de test.

L'exécution du projet est simple, cela se passe dans l'onglet présent en bas à gauche de l'image ci-dessus. Nous pouvons soit cliquer sur le carré rouge avec la flèche verte juste en dessous du Boot Dashboard ou bien faire cliquer droit sur CodeJavaApp [devtools] en dessous de local et (Re)start le projet.

Les intérêts d'apprendre et d'utiliser le Framework Spring sont multiples :

- Tout d'abord, le spring est présent partout dans le monde et est utilisé par des géants comme Microsoft, Google, Amazon, Alibaba... pour la réalisation d'innovations comme par exemple la réalisation de sites d'achat en ligne, le streaming ou bien d'autres encore.
- Ensuite, ce Framework est pensé pour faire de la performance. Le démarrage, l'arrêt d'une application sur spring sont très rapides et l'exécution est optimisée par défaut. Spring Boot aide tous les développeurs à créer des applications avec facilité. De plus, les serveurs web intégrés et la configuration automatique aident à démarrer rapidement le lancement d'application. Enfin, on peut même créer un projet en moins de 1 minute avec le Spring Initializr sur le site : <https://start.spring.io/>.
- Spring est sécurisé et a fait ses preuves dans la gestion rapide et responsable des problèmes de sécurité. D'ailleurs, les committers de spring travaillent en collaboration avec des professionnels de la sécurité dans le but de tester et de corriger toutes les vulnérabilités signalées. Enfin, Spring Security permet d'intégrer plus facilement les systèmes de sécurité standard présents dans l'industrie mais aussi de pouvoir fournir des solutions sécurisées et fiables par défaut.
- Spring dispose d'un énorme support à travers le monde entier grâce à sa communauté. Cela nous a permis de trouver des solutions à des problèmes que nous avons rencontrés durant la réalisation de ce projet. Que cela soit grâce à des vidéos sur youtube ou à des discussions entre développeurs sur stackoverflow.

C) Description des composants de l'application

Au sein du projet Sondix, l'on peut distinguer 2 Composants qui sont la Base de Données et l'application Sondix.

Le composant Base de données est caractérisé par notre base MySQL. Une base de données est pour faire simple un regroupement de données structurées et organisées dans le but d'en faciliter la récupération et l'utilisation. Les données collectées peuvent être diverses et variées mais dans notre cas nous nous sommes limités à des champs de texte c'est-à-dire varchar en MySQL.

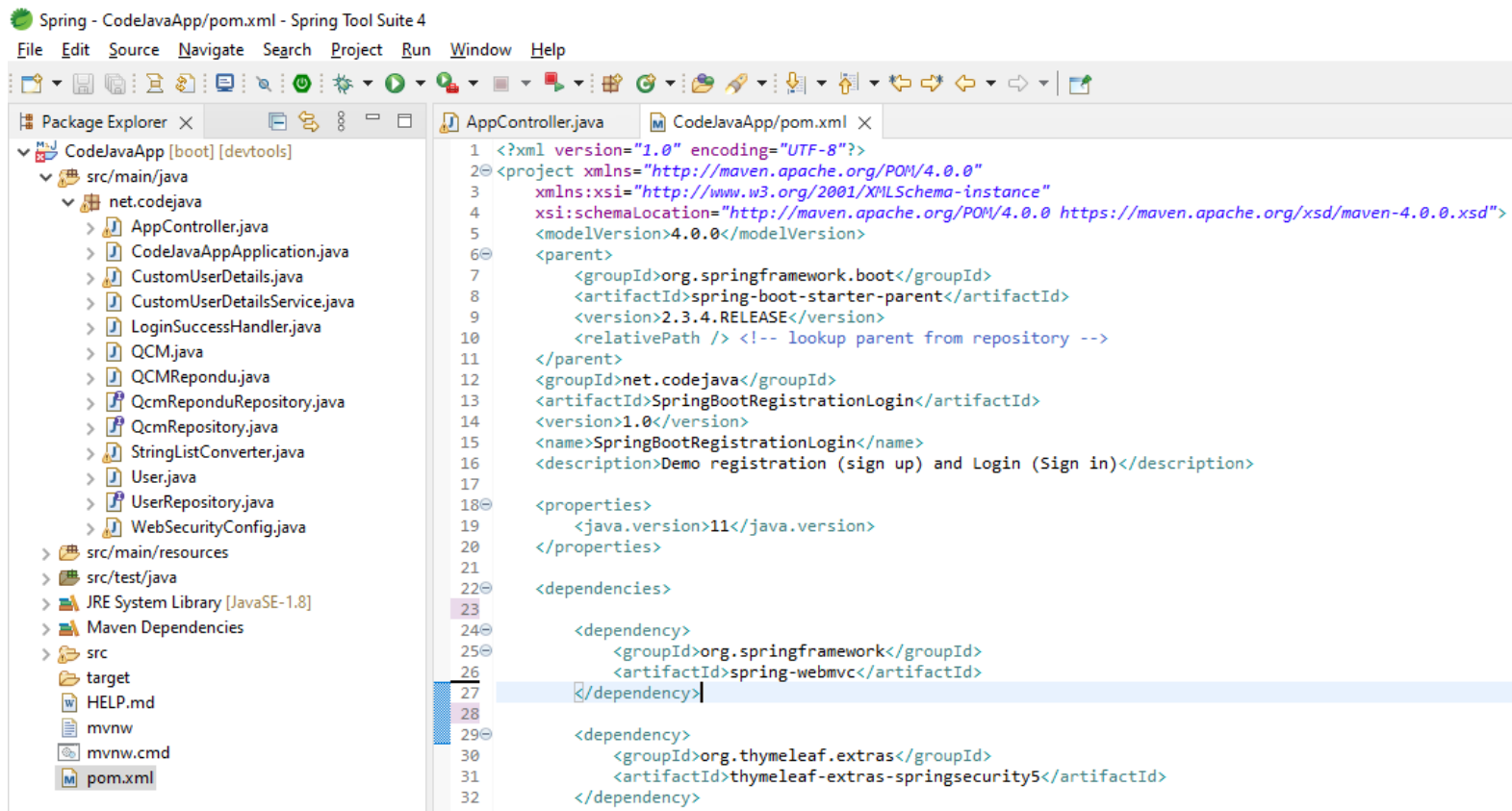
Pour totalement appréhender le fonctionnement de MySQL, il faut connaître 2 concepts liés qui sont :

- Base de données relationnelle : Au sein d'une base de données relationnelle comme MySQL, les données sont divisées en plusieurs zones de stockage nommées "Tables" au lieu de regrouper toutes les données dans une seule et unique zone de stockage. Cela dans le but de départager les données afin de les organiser, d'éviter leurs duplications et aussi de surcharger la base de données inutilement. Pour avoir les relations entre les différentes "Tables" l'utilisation d'une clé est nécessaire. Grâce à celle-ci on peut lier les données de plusieurs tables entre elles pour les manipuler ou les récupérer. Une clé est obligatoirement 100% unique comme par exemple un numéro d'identification numérique ou alors le numéro de votre carte vitale.
- Modèle client-serveur : La partie serveur correspond au lieu où réside réellement les données. Pour pouvoir avoir accès à ces données il faut en faire la demande via le client. Pour faire cela, l'on doit utiliser SQL (Structured Query Language) qui est un langage informatique utilisé pour exploiter des bases de données. Concrètement le client envoie une requête au serveur de base de données et si la requête est valide le serveur l'exécute.

Pour réaliser notre projet nous faisons tourner localement sur nos machines un serveur MySQL. Sur ce serveur est présente la base "sondixdb" utilisée pour l'application Sondix. Sur ce serveur nous avons 3 Tables différentes reliées entre elles grâce à des clés. Les Tables sont : users, qcm, qcmrepondu. Pour pouvoir accéder à ces données, notre autre composant qui est l'application Sondix en Spring utilise le SQL.

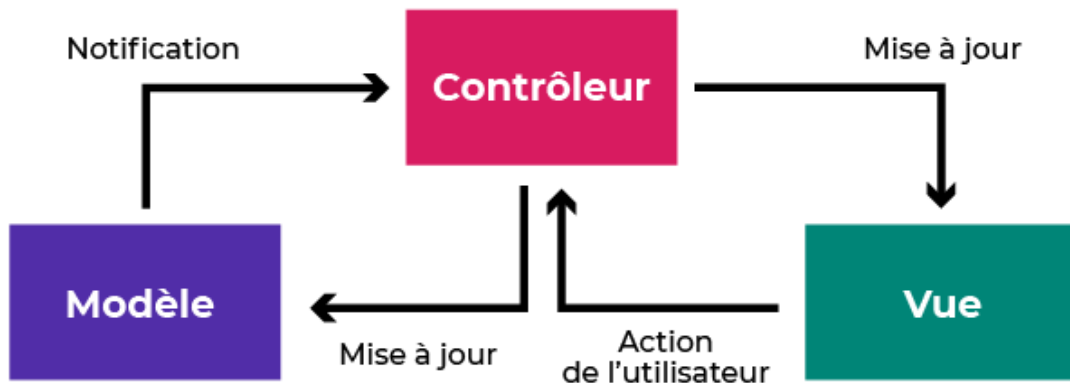
La composante application Sondix est réalisée en Spring. Le projet a été créé grâce à Spring Boot et l'équipe utilise le Framework Spring Tool Suite 4 pour le développement. L'application Sondix est une application Spring MVC qui permet de construire l'application Web en Java. Le principe du MVC est celui du Modèle / Vue / Contrôleur qui est en lien direct avec le IoC (Inversion of Control) du Spring Framework.

Pour utiliser Spring MVC, il faut déclarer sa dépendance dans le fichier pom.xml de notre projet :



Le MCV se divise en 3 parties :

- Le modèle représente la structure des données dans une application. Par exemple pour le projet Sondix nous avons une Table QCM. Cela est représenté en Spring par la classe QCM.java qui a comme annotation au-dessus de la déclaration de la classe : `@entity` et `@Table(name = "QCM")`.
- La vue qui représente l'interface de l'utilisateur c'est-à-dire tout ce qu'il voit à l'écran et avec lequel il peut interagir. Les vues dans le projet Sondix sont présentes dans le dossier "src/main/resources" et sont appelées par le contrôleur du nom de AppController.java qui est le contrôleur principal de notre application.
- Le contrôleur représente les classes qui se connectent au modèle et à la vue. Celui-ci permet de communiquer entre la vue et le modèle. Un contrôleur en Spring est une classe Java portant la notation `@controller` au-dessus de la déclaration de la classe. Pour que le contrôleur soit averti lors du traitement d'une requête, il faut ajouter l'une des annotations suivantes : `@GetMapping`, `@PostMapping` ou `@RequestMapping`.



D) Lien avec la partie conception

Pour bien expliquer le lien qui existe avec les 2 composants qui sont l'application et la base de données, je vais parler de la partie de l'employé. Voici la page sur laquelle arrive un employé une fois qu'il s'est connecté :

Créer un QCM
Mon tableau de QCM
X

Deconnexion

Vincent Pisano, bienvenue sur votre page d'employé

Outils pour la création et supervision de votre QCM

Sélectionner l'onglet que vous souhaitez dans la barre de navigation, cela vous permettra d'accéder aux différentes sections vous permettant de créer un QCM et les gérer.

L'employé a la possibilité d'utiliser la barre de navigation pour aller Créer un QCM :

Créer un QCM Mon tableau de QCM X Deconnexion

Vincent Pisano, bienvenue sur votre page d'employé

Outils pour la création et supervision de votre QCM

Sélectionner l'onglet que vous souhaitez dans la barre de navigation, cela vous permettra d'accéder aux différentes sections vous permettant de créer un QCM et les gérer.

Nombres de questions : 1
Créer QCM

L'application Sondix affiche à l'employé la liste de tous les QCM présents dans la base de données dans l'onglet "Mon tableau de QCM" :

Créer un QCM Mon tableau de QCM X Deconnexion

Vincent Pisano, bienvenue sur votre page d'employé

Outils pour la création et supervision de votre QCM

Sélectionner l'onglet que vous souhaitez dans la barre de navigation, cela vous permettra d'accéder aux différentes sections vous permettant de créer un QCM et les gérer.

Titre	Catégorie	Statut	Commentaire	Suppression	Modification	Mise en ligne
les simpson	Culture	Créé		Supprimer	Modifier	Demande
Avengers	Culture	Créé		Supprimer	Modifier	Demande

Pour comprendre cela nous devons aller voir dans le controller AppController.java :

```

AppController.java ×  employe.html
248         model.addAttribute("QCM", myQCM);
249
250         return "faireQcm";
251     }
252
253     @GetMapping("/responsable")
254     public String viewResponsablePage(Model model) {
255         List<QCM> listQcm = qcmRepo.findAllNotStatutCreate();
256
257         model.addAttribute("listQcm", listQcm);
258         return "responsable";
259     }
260
261     @GetMapping("/employee")
262     public String viewEmployeePage(Model model) {
263         List<QCM> listQcm = qcmRepo.findAll();
264
265         model.addAttribute("listQcm", listQcm);
266         return "employee";
267     }

```

Le `@GetMapping("/employee")` permet d'avertir le contrôleur quand l'utilisateur arrive sur le page "<http://localhost:8080/employee>" et de savoir quoi faire. En l'occurrence on déclare une liste de QCM qui récupère toutes les instances présentes dans le "qcmRepo" qui correspond au repository des QCM grâce à JpaRepository qui nous permet de communiquer avec notre base de données(ligne 255). Une fois la liste de tous les QCM de l'application récupérée nous la transmettons au modèle en l'ajoutant à ses attributs avec comme premier argument le nom de l'attribut puis sa valeur. Enfin pour terminer on redirige la route vers "employee" ce qui permet à l'application de comprendre qu'il faut afficher le contenu du fichier "employee.html".

La méthode utilisée pour récupérer tous les QCM qui est `findAll()` est une méthode propre à JpaRepository.

Pour ce projet nous avons aussi développé nos propres méthodes pour communiquer avec notre base de données. Par exemple pour communiquer avec la table QCM présente dans notre base de données “sondixdb” nous utilisons l’interface QcmRepository.java que voici :

```

AppController.java  employe.html  QCM.java  QcmRepository.java X
1  package net.codejava;
2
3  import java.util.List;
7
8  public interface QcmRepository extends JpaRepository<QCM, Long> {
9      @Query("SELECT u FROM QCM u WHERE u.titre = ?1 ")
10     public QCM findByTitre(String titre);
11
12     @Query("delete FROM QCM WHERE idQCM = ?1")
13     public QCM deleteById(int id);
14
15     @Query("SELECT u FROM QCM u WHERE u.titre = ?1 and u.statut ='Créé'")
16     public QCM findByTitreAndCreateStatut(String titre);
17
18     @Query( value = "SELECT * FROM QCM u  WHERE  u.statut !='Créé'",
19             nativeQuery = true)
20     public List<QCM> findAllNotStatutCreate();
21

```

Les repository sont des classes ou des composants qui encapsulent la logique requise pour accéder aux sources de données. Ils centralisent les fonctionnalités d'accès aux données communes (la base de données), offrant une meilleure maintenabilité et découplant l'infrastructure ou la technologie utilisée pour accéder aux bases de données à partir de la couche de modèle(=les entités).

Dans cette interface nous déclarons des méthodes en commençant par leurs données une visibilité par exemple “public”.

Puis nous avons le type de retour comme “QCM”, et le nom de la méthode et ses paramètres. Juste au-dessus de la méthode l’on indique @Query(“SELECT u FROM QCM u WHERE u.titre = ?1”) qui correspond à la requête SQL à exécuter.

Le “?1” correspond au premier paramètre de la méthode.

Enfin l'affichage des données de la base de données, dans notre cas des QCM se fait grâce à Thymeleaf comme voici ci-dessous :

```
<tbody>
  <tr th:each="qcm: ${ListQcm}">
    <td th:text="${qcm.idQCM}"></td>
    <td th:text="${qcm.titre}"></td>
    <td th:text="${qcm.categorie}"></td>

    <td th:text="${qcm.statut}"></td>
    <td th:text="${qcm.commentaire}"></td>

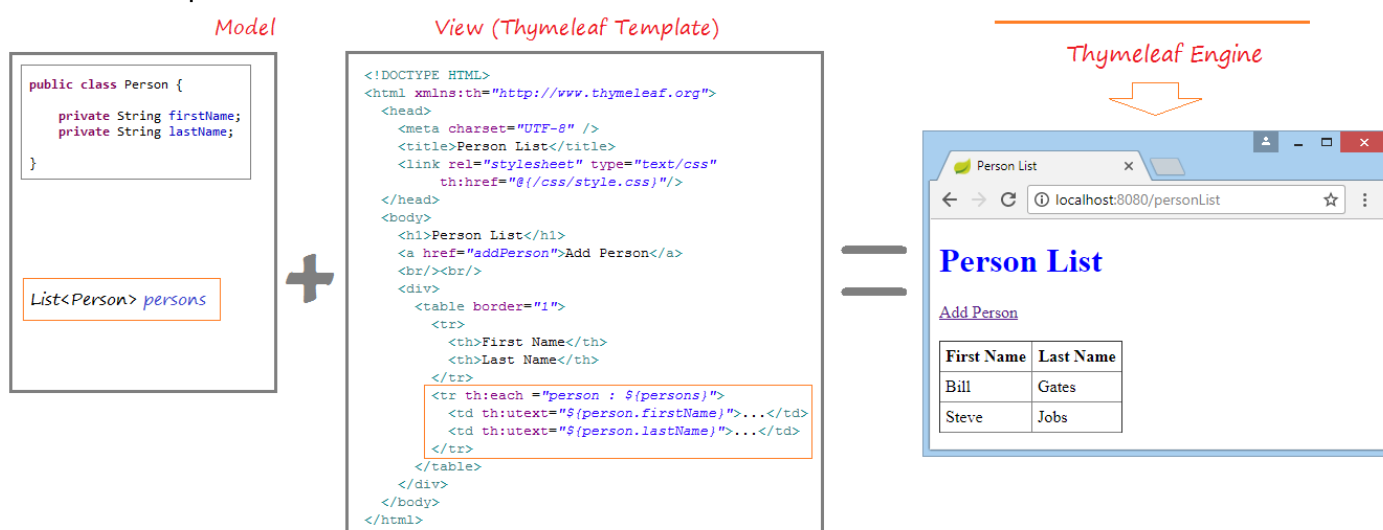
  </tr>
```

Thymeleaf est un Java XML/XHTML/HTML5 Template Engine qui peut travailler à la fois dans des environnements Web et non Web. Il est mieux adapté pour diffuser XHTML/HTML5 sur View (View Layer) des applications Web basées sur MVC. Il fournit une intégration complète de Spring Framework.

Thymeleaf est un logiciel à code source ouvert (open source) sous licence Apache 2.0.

Thymeleaf Engine (le moteur de Thymeleaf) va lire un fichier modèle et le combiner avec des objets Java pour générer un autre document.

Voici un exemple concret :



E) Exemple de fonctionnement de l'application

L'acteur "employé" dispose d'actions mises à sa disposition par l'application Sondix. Il peut créer un QCM, le modifier, le supprimer ou en demander sa mise en ligne comme vous pouvez le voir pour le QCM Avengers ci-dessous :

Créer un QCM Mon tableau de QCM X Deconnexion

Vincent Pisano, bienvenue sur votre page d'employé

Outils pour la création et supervision de votre QCM

Sélectionner l'onglet que vous souhaitez dans la barre de navigation, cela vous permettra d'accéder aux différentes sections vous permettant de créer un QCM et les gérer.

Titre	Catégorie	Statut	Commentaire	Suppression	Modification	Mise en ligne
les simpson	Culture	Demande de mise en ligne				
Avengers	Culture	Créé		Supprimer	Modifier	Demande

Je présenterai plus en détail la création d'un QCM dans la partie suivante de ce rapport. La suppression, modification et demande de mise en ligne d'un QCM est possible seulement si le QCM est dans le statut "créé" ou "Refusé". Sinon les boutons "Supprimer", "Demande" et "Modifier" seront invisibles. L'action d'appuyer sur le bouton "Supprimer" transmettra au AppController.java l'identifiant du QCM pour pouvoir le supprimer. Le contrôleur à partir de cet identifiant supprimera le QCM de la base de donnée par le biais du "qcmRepo" qui est le repository de la table QCM.

```
AppController.java x QcmRepository.java employe.html
409
410 @PostMapping("/deleteData")
411 public String processDeleteData(@RequestParam("idQCM") Long id, Model model) {
412     //Suppression du QCM
413     qcmRepo.deleteById(id);
414     //Récupération de tout les QCM après la suppression
415     List<QCM> listQcm = qcmRepo.findAll();
416     //Passage des QCM à la vue
417     model.addAttribute("listQcm", listQcm);
418     //redirection vers la vue employe
419     return "redirect:/employe";
420 }
421
```

La méthode "deleteById" a été implémentée comme ceci :

```

AppController.java × QcmRepository.java × employe.html
1 package net.codejava;
2
3 import java.util.List;
4
5
6
7
8 public interface QcmRepository extends JpaRepository<QCM, Long> {
9
10 @Query("delete FROM QCM WHERE idQCM = ?1")
11 public QCM deleteById(int id);
12

```

En plus de supprimer, l'employé peut modifier le QCM en cliquant sur le lien "Modifier" présent dans la colonne "Modification" ce qui lui permet de changer le titre, la catégorie, les énoncés des questions et des réponses.

[Retour](#)

Modification du QCM

Titre :

Catégorie :

Question 1 :

Réponses 1 :

[Valider](#)

Une fois les modifications faites, l'action de cliquer sur le bouton valider met à jour le QCM à travers la méthode "processModificationUpload" du AppController.java ci-dessous :

```

AppController.java X employe.html
453 //permet la modification d'un QCM
454 @PostMapping("/process_modification")
455 public String processModificationUpload(@RequestParam("titreancien") String titreancien,
456 @RequestParam("titre") String titre, @RequestParam("categorie") String categorie,
457 @RequestParam("questions") List<String> questions, @RequestParam("reponses1") List<String> reponses1,
458 @RequestParam(value = "reponses2", required = false, defaultValue = "") List<String> reponses2,
459 @RequestParam(value = "reponses3", required = false, defaultValue = "") List<String> reponses3,
460 @RequestParam(value = "reponses4", required = false, defaultValue = "") List<String> reponses4,
461 @RequestParam(value = "reponses5", required = false, defaultValue = "") List<String> reponses5,
462 @RequestParam(value = "reponses6", required = false, defaultValue = "") List<String> reponses6,
463 @RequestParam(value = "reponses7", required = false, defaultValue = "") List<String> reponses7,
464 @RequestParam(value = "reponses8", required = false, defaultValue = "") List<String> reponses8,
465 @RequestParam(value = "reponses9", required = false, defaultValue = "") List<String> reponses9,
466 @RequestParam(value = "reponses10", required = false, defaultValue = "") List<String> reponses10,
467 Model model) {
468
469 //Récupération du QCM à partir de son ancien Titre
470 QCM myModifyQCM = qcmRepo.findByTitre(titreancien);
471
472 //Mise a jour du QCM
473 myModifyQCM.setTitre(titre);
474 myModifyQCM.setCategorie(categorie);
475 myModifyQCM.setQuestions(questions);
476 myModifyQCM.setReponses1(reponses1);
477 myModifyQCM.setReponses2(reponses2);
478 myModifyQCM.setReponses3(reponses3);
479 myModifyQCM.setReponses4(reponses4);
480 myModifyQCM.setReponses5(reponses5);
481 myModifyQCM.setReponses6(reponses6);
482 myModifyQCM.setReponses7(reponses7);
483 myModifyQCM.setReponses8(reponses8);
484 myModifyQCM.setReponses9(reponses9);
485 myModifyQCM.setReponses10(reponses10);
486
487 //sauvegarde de la mise a jour dans la base de données
488 qcmRepo.save(myModifyQCM);
489
490 //redirection vers la vue employe
491 return "redirect:/employe";
492 }
493

```

La dernière action possible par l'employé en dehors de la création d'un QCM est de demander la mise en ligne d'un QCM en cliquant sur le bouton "Demande" présent dans la colonne ayant pour titre "Mise en Ligne". Cela changera le Statut du QCM "les simpson" de "créé" au nouveau statut "Demande de mise en ligne" :

Vincent Pisano, bienvenue sur votre page d'employé

Outils pour la création et supervision de votre QCM

Sélectionner l'onglet que vous souhaitez dans la barre de navigation, cela vous permettra d'accéder aux différentes sections vous permettant de créer un QCM et les gérer.

Titre	Catégorie	Statut	Commentaire	Suppression	Modification	Mise en ligne
les simpson	Culture	Demande de mise en ligne				
Avengers	Culture	Créé		Supprimer	Modifier	Demande

Cette action fait appel à la méthode "processUpdateData" du AppController.java :

```

493
494 //quand l'employe demande la mise en ligne d'un QCM cela change l'état du QCM en : Demande de mise en ligne
495 @RequestMapping("/processOnline")
496 public String processUpdateData(@RequestParam("titre") String titre, Model model) {
497     //On récupère le QCM a partir de son titre
498     QCM myUpdateQCM = qcmRepo.findByTitre(titre);
499
500     //On change son statut pour Demande de mise en ligne
501     myUpdateQCM.setStatut("Demande de mise en ligne");
502
503     //On sauvegarde le changement dans la base de données
504     qcmRepo.save(myUpdateQCM);
505
506     //On récupère tout les QCM
507     List<QCM> listQcm = qcmRepo.findAll();
508
509     //On passe la liste de tout les QCM dans la vue
510     model.addAttribute("listQcm", listQcm);
511     viewEmployePage(model);
512
513     //On redirige vers la vue employe
514     return "redirect:/employe";
515 }

```

Un autre exemple de fonctionnement de l'application Sondix concerne le rôle du responsable. Celui-ci arrive sur cette page une fois connecté :

Deconnexion

Bouchra Bnik, bienvenue sur votre page de responsable

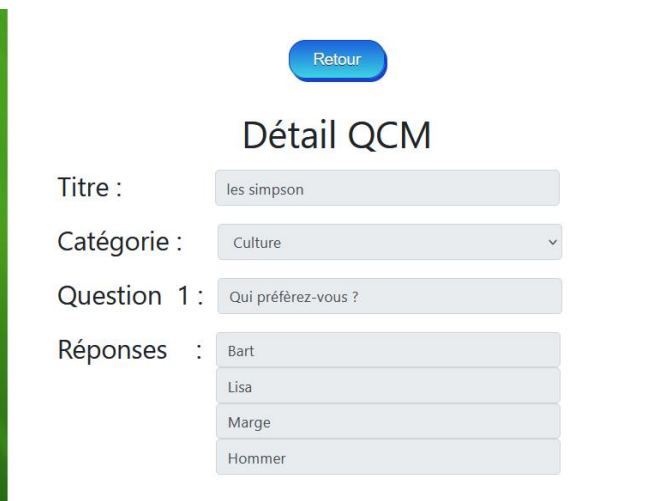
Liste des demandes de mise en ligne des QCM

Dans le tableau ci-dessous, vous retrouverez la liste de tous les QCM que vous avez validés ou qui sont en attentes de validation. Veuillez mettre un commentaire pour préciser les raisons de vos décisions.

Titre	Catégorie	Commentaire	Statut	Détail	Décision
les simpson	Culture		Demande de mise en ligne	Voir	<div>Commentaire</div> <div>Validé</div> <div>Refusé</div>

Le but du responsable est de valider ou refuser les QCM qui ont été fait par l'employé et qui sont dans le statut "Demande de mise en ligne" avec un commentaire.

Pour ce faire le responsable à la possibilité de voir en détail le QCM en question :



Retour

Détail QCM

Titre : les simpson

Catégorie : Culture

Question 1 : Qui préférerez-vous ?

Réponses :
Bart
Lisa
Marge
Homer

Une fois le QCM validé, alors, les internautes auront la possibilité de pouvoir le faire. Sinon s'il est refusé, alors l'employé peut le supprimer ou le modifier pour ensuite re demander sa mise en ligne.

Enfin, l'un des éléments principaux pour la création de ce projet était l'analyse des réponses aux QCM. Donc le rôle de Directeur a été mis en place pour cette partie. Le directeur une fois connecté arrive sur cette page :

Statistiques générales
Archivage de QCM
X
Deconnexion

Hacene Mengueliti, bienvenue sur votre page de directeur

Outils pour le suivi et supervision des statistiques des QCM de Sondix

Sélectionner l'onglet que vous souhaitez dans la barre de navigation, cela vous permettra d'accéder aux différentes sections vous permettant de voir les statistiques générales et l'archivage des QCM.

Il peut aller dans l'onglet "Statistiques générales" :

Statistiques générales
Archivage de QCM
X
Deconnexion

Hacene Mengueliti, bienvenue sur votre page de directeur

Outils pour le suivi et supervision des statistiques des QCM de Sondix

Sélectionner l'onglet que vous souhaitez dans la barre de navigation, cela vous permettra d'accéder aux différentes sections vous permettant de voir les statistiques générales et l'archivage des QCM.

Statistiques générales du site web Sondix

Nombre QCM en Ligne	2
QCM le plus populaire	les simpson
QCM le moins populaire	Avengers
Total internaute Inscrit	3
Total QCM Répondu	3

Ce tableau de bord lui permet de connaître d'un seul coup d'œil le total de QCM répondu, le nombre d'internautes inscrit à l'application Sondix, le nombre de QCM en ligne et le QCM le plus / le moins populaire.

Il peut aller dans l'onglet "Archivage QCM" :

Statistiques générales Archivage de QCM X Deconnexion

Hacene Menguelti, bienvenue sur votre page de directeur

Outils pour le suivi et supervision des statistiques des QCM de Sondix

Sélectionner l'onglet que vous souhaitez dans la barre de navigation, cela vous permettra d'accéder aux différentes sections vous permettant de voir les statistiques générales et l'archivage des QCM.

Tableaux contenant tout les QCM de site Sondix

Titre	Catégorie	Détails	Statut	Action
les simpson	Culture	Voir	Validé	Archivé
Avengers	Culture	Voir	Validé	Archivé

De plus, le Directeur dispose du pouvoir d'archiver des QCM quand il pense que celui-ci a assez duré ou que les statistiques recueillies représentent un nombre conséquent de données.

Enfin il peut aller voir en détails sur chaque QCM la répartition des réponses des internautes comme le montre la page ci-dessous :

[Retour](#)

Statistiques du Sondage :

Titre : les simpson

Nombre d'internautes ayant répondu : 2

Catégorie : Culture

Question 1 : Qui préférez-vous ?

Réponses :

Bart	50.0 %
Lisa	0.0 %
Marge	0.0 %
Hommer	50.0 %

F) Code source des parties significatives

L'une des deux parties clé au sein de l'application Sondix est la création d'un QCM par l'employé. Pour ce faire l'employé après s'être connecté choisit le nombre de questions qu'il veut dans son QCM (par défaut 1 seule) et appuie sur le bouton "Créer QCM".

Créer un QCM

Mon tableau de QCM X

Deconnexion

Vincent Pisano, bienvenue sur votre page d'employé

Outils pour la création et supervision de votre QCM

Sélectionner l'onglet que vous souhaitez dans la barre de navigation, cela vous permettra d'accéder aux différentes sections vous permettant de créer un QCM et les gérer.

Nombres de
questions :

1

Créer QCM

Cette action spécifique appelle la méthode viewCreationQCMPage du AppController.java

:

```
AppController.java X employee.html creation_qcm.html
369
370 // page de création d'un QCM ou on passe un qcm vierge a remplir
371 @PostMapping("/creation_qcm")
372 public String viewCreationQCMPage(Model model, @RequestParam("nbqts") String nbQuestions) {
373
374     // Création d'une liste pour récupérer le nombre de question a afficher
375     List<String> nbq = new ArrayList<String>();
376
377     // a partir du nombre de question donner par l'employeur dans l'interface employe
378     // j'augmente la taille de ma liste
379     for (int i = 0; i < Integer.valueOf(nbQuestions); i++) {
380         nbq.add("");
381     }
382
383     // je passe ma liste a la vue pour savoir combien de question créer
384     model.addAttribute("nbqts", nbq);
385
386     // je passe l'objet QCM a ma vue qui s'occupera de le remplir
387     model.addAttribute("qcm", new QCM());
388
389     // je redirige vers la vue creation_qcm
390     return "creation_qcm";
391 }
```

Cette méthode nous redirige vers la vue "création_qcm" qui permet à l'employé de remplir le QCM :

Le fait de cliquer sur le bouton "Créer" une fois les champs remplis va grâce à thymeleaf sauvegarder le titre, la catégorie, les questions et les réponses dans l'objet QCM qui a été passé à la vue.

Pour plus de détails sur l'utilisation de thymeleaf dans un formulaire aller voir : <https://devstory.net/12385/thymeleaf-th-object-et-syntaxe-asterisk>

Le fait de cliquer sur le bouton "Créer" va appeler la méthode processCreation du contrôleur "AppController.java" :

```

AppController.java X  employee.html  creation_qcm.html
392
393 // enregistrement du QCM ou le qcm est unique en fonction de son titre
394 @PostMapping("/process_creation")
395 public String processCreation(QCM qcm) {
396
397     if (qcmRepo.findByTitre(qcm.getTitre()) != null) {
398         return "creation_failure";
399     } else {
400
401         qcmRepo.save(qcm);
402
403         return "redirect:/employee";
404     }
405 }
    
```

Cette méthode vérifie qu'il n'y ait pas déjà un QCM qui a le même titre. Si cela n'est pas le cas alors on sauvegarde le QCM dans la table "QCM" de la base de données "sondixdb". S'il existe un QCM ayant le même titre, alors l'application Sondix préviendra l'employé que la création de ce QCM est annulée avec un message et le redirigera vers la vue "employé".

La deuxième partie clé de l'application Sondix est le fait de répondre à des QCM par les internautes. Un internaute une fois connecté arrive à cette page :

Jean Gautier, bienvenue sur votre page d'internaute

Informations sur votre compte client

Sélectionner l'onglet que vous souhaitez dans la barre de navigation. Cela vous permettra d'accéder à vos informations personnelles, aux QCM disponibles / réalisés ainsi qu'à vos points de fidélité.



Il peut aller dans l'onglet "Informations personnelles" :

Jean Gautier, bienvenue sur votre page d'internaute

Informations sur votre compte client

Sélectionner l'onglet que vous souhaitez dans la barre de navigation. Cela vous permettra d'accéder à vos informations personnelles, aux QCM disponibles / réalisés ainsi qu'à vos points de fidélité.

Vos informations personnelles :

Nom : Jean

Prénom : Gautier

Email : jean@live.fr

Pseudonyme : Inter

Numéro de téléphone : 0459905036

Date de naissance : 07/12/2001

Catégorie socio-professionnel : Employés



Il peut aller dans l'onglet "Points de fidélité" :

Jean Gautier, bienvenue sur votre page d'internaute

Informations sur votre compte client

Sélectionner l'onglet que vous souhaitez dans la barre de navigation. Cela vous permettra d'accéder à vos informations personnelles, aux QCM disponibles / réalisés ainsi qu'à vos points de fidélité.

Vos informations concernant vos points de fidélité

Points de fidélité : **200**



Il peut aller dans l'onglet "QCM réalisés" :

Jean Gautier, bienvenue sur votre page d'internaute

Informations sur votre compte client

Sélectionner l'onglet que vous souhaitez dans la barre de navigation. Cela vous permettra d'accéder à vos informations personnelles, aux QCM disponibles / réalisés ainsi qu'à vos points de fidélité.

Voici les QCM que vous avez réalisés

Titre	Catégorie	Action
les simpson	Culture	Voir
Avengers	Culture	Voir



Il peut aller dans l'onglet "QCM disponibles" :

Mike Thyson, bienvenue sur votre page d'internaute

Informations sur votre compte client

Sélectionner l'onglet que vous souhaitez dans la barre de navigation. Cela vous permettra d'accéder à vos informations personnelles, aux QCM disponibles / réalisés ainsi qu'à vos points de fidélité.

Voici les QCM qui sont encore disponibles

Titre	Catégorie	Action
les simpson	Culture	Faire
Avengers	Culture	Faire

Une fois qu'il choisit un QCM à faire il est redirigé vers la vue "faireqcm" :

[Retour](#)

QCM : Avengers

Question 1 : Qui est le plus fort ? ☒ Hulk ☐ Iron Man ☐ Thor

Question 2 : Qui détester vous ? ☐ Thanos ☒ Loki

[Envoyer](#)

Lorsque l'on arrive sur cette page l'AppController.java exécute la méthode "viewFaireQCMPage" qui transmet à la vue le QCM que l'internaute veut faire et un objet QCMRepondu pour que la vue sauvegarde le numéro de la réponse choisie. Avec l'image ci-dessus cela sauvegardera la valeur 1 qui correspond à la réponse Hulk puis la valeur 2 qui correspond à Loki . Quand on clique sur le bouton "Envoyer" cela exécute la méthode processregisterReponses du AppController.java :

```

AppController.java × employee.html creation_qcm.html faireQcm.html
524
525 // On sauvegarde le changement dans la base de données
526 qcmRepo.save(myUpdateQCM);
527
528 // On récupère tout les QCM
529 List<QCM> listQcm = qcmRepo.findAll();
530
531 // On passe la liste de tout les QCM dans la vue
532 model.addAttribute("listQcm", listQcm);
533 viewEmployePage(model);
534
535 // On redirige vers la vue employe
536 return "redirect:/employe";
537 }
538
539 // enregistre les réponses de l'internaute a un QCM en gardant l'idQCM et id de
540 // l'utilisateur
541 @PostMapping("/registerResponses")
542 public String processregisterResponses(Model model,
543     @CurrentSecurityContext(expression = "authentication?.name") String username,
544     @RequestParam("titre") String titre, QCMRepondu QCMRepondu) {
545
546     // on récupère l'internaute ayant répondu au QCM
547     User myUser = userRepo.findByEmail2(username);
548
549     // on augmente ses points de fidélité par 100
550     myUser.setFidelite(myUser.getFidelite() + 100);
551
552     // on récupère le QCM auquel il a répondu grâce a son titre
553     QCM myQCM = qcmRepo.findByTitre(titre);
554
555     // on sauvegarde dans l'identifiant du user et du QCM dans le QCMRepondu sachant
556     // que ses réponses qu'il a choisi on était sauvegarder par la vue faireQcm
557     QCMRepondu.setIdUser(myUser.getId().toString());
558     QCMRepondu.setIdQCM(myQCM.getIdQCM().toString());
559     qcmReponsesRepo.save(QCMRepondu);
560
561     return "redirect:/internaute";
562 }
563

```

Cette méthode enregistre dans la Table "QCMRepondu" de la base "sondixdb" l'identifiant de l'internaute, du QCM et les réponses de l'internaute (séparées par un caractère spécifique s'il y en a plusieurs) et le redirige vers la vue "internaute".

G) Le modèle de la base de données

La base de données de l'application est "sondixdb". Elle est composée de 3 tables.

La table "users" est la table où sont présents les utilisateurs de l'application.

Table: users

Columns:

id	bigint AI PK
birth	varchar(20)
email	varchar(45)
fidelite	int
first_name	varchar(20)
last_name	varchar(20)
password	varchar(64)
phone	varchar(20)
pseudo	varchar(20)
role	varchar(20)
sexe	varchar(20)
socio	varchar(255)

Elle est composée d'un "id" qui est la clé primaire (= index) de la table. Cet identifiant est auto incrémenté. De plus, nous avons décidé que les utilisateurs de l'application seront départagés en fonction de leur adresse email. Ce qui signifie qu' un utilisateur ne pourra se créer qu'un seul compte sur Sondix par mail. Donc "email" est unique dans la table "users".

La table “qcm” est la table où sont présents les sondages réalisés par les employés.

Table: qcm

Columns:

idqcm	bigint AI PK
categorie	varchar(50)
commentaire	varchar(50)
questions	varchar(512)
reponses1	varchar(255)
reponses10	varchar(255)
reponses2	varchar(255)
reponses3	varchar(255)
reponses4	varchar(255)
reponses5	varchar(255)
reponses6	varchar(255)
reponses7	varchar(255)
reponses8	varchar(255)
reponses9	varchar(255)
statut	varchar(50)
titre	varchar(50)

Elle est composée d’un “idqcm” qui est la clé primaire (= index) de la table. Cet identifiant est auto incrémenté. De plus, lorsque les employés valident la création d’un QCM, une vérification est faite pour vérifier si le titre du QCM en cours de création est déjà présent dans la table “qcm”. Si un QCM avec le même titre est déjà présent alors nous prévenons l’employé et empêchons la création de ce QCM. Le champ “questions” est une string qui correspond à la concaténation des questions d’un QCM séparé par un séparateur spécifique.

Nous avons décidé de stocker les questions et les réponses sous formes de chaînes de caractères séparés par le caractère spécial “;” afin de limiter le nombre de requête réalisée sur la base de données; aussi bien lors de la création d’un QCM ou lors des statistiques réalisées par Sondix sur les QCM et les QCMRépondu. En effet l’autre solution aurait été de faire une table Question et une table réponses; cela aurait plus sollicité notre base de données par le biais de nombreuses jointures afin d’obtenir les QCM et d’en faire des statistiques.

Exemple:

questions = "Est-ce que tu aimes Noël ?; Où as-tu envie de partir en vacances cet été ?; Travailles-tu l’été ?"

Les champs “reponses1” à “reponses10” correspondent quant à eux à la concaténation des réponses de toutes les questions du QCM séparées par un séparateur spécifique.

Exemple :

reponses1 = “Oui;Non” reponses2 = “Italie,Russie;USA” reponses3 = "Non;Peut Être;Oui"

La dernière table de notre base de données est la table “qcmrepondu” qui stocke les réponses des internautes des QCM qu’ils ont réalisés.

Table: qcmrepondu

Columns:	
idqcmrepondu	bigint AI PK
idqcm	varchar(50)
id_user	varchar(50)
reponses	varchar(255)

Elle est composée d’un “idqcmrepondu” qui est la clé primaire (= index) de la table. Cet identifiant est auto incrémenté. L’identifiant “id_user” est une clé étrangère vers l’identifiant de la table “users” pour retrouver qui a répondu. De plus l’identifiant “idqcm” permet de savoir quel QCM a été fait. Enfin “réponses” est une string correspondant à la concaténation des réponses à un QCM représentée sous la forme de compteurs séparés par un séparateur spécifique.

Exemple :

reponses = “1;3;2” signifie que l’internaute a répondu 1 à la question 1 puis 3 à la question 2 et enfin 2 à la question 3.

A chaque lancement de l’application Sondix, elle se connecte à la base de données “sondixdb” avec les paramètres qui sont détaillés dans le fichier “application.properties” :

```

application.properties X
1 spring.jpa.hibernate.ddl-auto=none
2 spring.datasource.url=jdbc:mysql://localhost:3306/sondixdb
3 spring.datasource.username=root
4 spring.datasource.password=password
5 spring.jpa.properties.hibernate.format_sql=true
  
```

A chaque fois que l’on arrête l’application, la connexion à la base est fermée par le framework.

H) Limites et évolutions possibles

Les limites de l'application Sondix sont :

- L'employé ne peut pas mettre d'image dans les QCM.
- Les internautes ne peuvent pas utiliser leurs points de fidélité.
- L'employé ne peut pas rechercher au sein de la base de données de QCM existantes des questions et des réponses pour les réutiliser.

Les évolutions possibles :

- Pour les internautes, intégrer une barre de recherche par nom des QCM quand il y en aura beaucoup.
- Pour le directeur, afficher les statistiques des réponses d'un QCM sous forme de graphique (camembert) pour une lecture plus simple et rapide.
- Rendre possible l'ajout d'images dans la base de données pour les mettre dans les QCM.
- Ajouter pour les internautes un classement par rapport à leurs points de fidélités pour ajouter un petit aspect compétitif.
- Rendre possibles pour les internautes de modifier leurs réponses à des QCM qu'ils ont déjà finis.
- Ajouter un magasin permettant aux internautes d'utiliser leurs points de fidélité.

I) Code source

Le code source de l'application est disponible sur ce dépôt GITHUB :
<https://github.com/chrabou/projetqcm>

J) Conclusion

C'est la première fois que nous travaillons tous les trois ensemble sur un projet qui possède un but bien défini, un cahier des charges clair et qui est plus orienté vers le monde du travail.

Le projet de l'entreprise Sondix consiste à mettre en place un nouveau système informatique permettant de créer et de mettre en ligne des QCM sur le web de manière générique.

De l'avis général, nous avons consolidé nos connaissances en JAVA, HTML, CSS, SQL et appris à utiliser le Framework MVC Spring.

De plus, nous avons appliqué la méthode Agile durant ce projet afin bien nous répartir les tâches pour réaliser les objectifs dans les temps.

La cohésion au sein du groupe était bonne.

Une bonne expérience à renouveler !