Hashcode

**Groupe No :** 6

Rapport de construction et de transition

Butty Joé

Fuchs Nicolas

Rial Jonathan

**Filière :** Informatique

**Technologies :** Java EE, MySQL

**Date de rendu :** 20.05.2018

**Superviseurs :** Prof. Houda Chabbi Drissi

Prof. Pierre Kuonen

Prof. Omar Abou Khaled

**Client:** Prof. Pierre Kuonen

**Table des matières**

1 Implémentation SGBD 4

1.1 Vos choix en tant qu’administrateur de votre SGBD 4

1.1.1 Estimation de la taille des données 4

1.1.2 Comptes sur votre SGBD 4

1.2 Vos choix pour l’optimisation et la concurrence/synchronisation 4

1.2.1 Choix d’index pour chaque relation 4

1.2.2 Argumentation sur la dénormalisation 4

1.2.3 Choix pour la concurrence 4

1.2.4 Les procédures stockées et vues 5

1.3 Etude d’un plan d’exécution 5

1.4 Code SQL dans cet ordre 5

1.4.1 Code de l’implémentation des tables 5

1.4.2 Code des triggers 5

1.4.3 Code des procédures stockées et des fonctions implémentés 5

1.4.4 Code de création des utilisateurs avec leurs privilèges 5

1.4.5 Remarques 5

1.5 Caractéristiques SGBD 5

1.5.1 Explicitez les triggers et leurs caractéristiques 5

1.5.2 Explicitez les niveaux d’isolations de votre SGBD et le protocole qu’il utilise 5

1.5.3 Le support de XML et XSD 5

1.5.4 Les types d’index disponibles 5

2 Implémentation SI 6

2.1 Environnement spécifique de développement 6

2.1.1 Framework 6

2.2 Éléments architecturaux 6

2.2.1 Couche présentation 6

2.2.2 Couche métier 7

2.2.3 Couche service 7

2.3 Éléments technologiques 8

2.3.1 Outils de gestion de versionning (CVS / SVN) 8

2.3.2 Outils de modélisation 8

2.3.3 Outils de documentation du code 8

2.3.4 Outils de journalisation de prototype 8

2.3.5 Outils de test 9

2.3.6 Outils de gestion de licences 9

2.3.7 Outils de d’installation et de packaging 9

2.4 Diagrammes d’architecture 10

2.4.1 Diagramme(s) de package 10

2.4.2 Diagramme(s) d’état-transition 10

2.4.3 Diagramme de déploiement 11

2.5 Choix technologique 12

2.6 Contraintes d’utilisation technologique 13

2.7 Outils administratifs 14

3 Conclusion 14

3.1 Objectifs atteints 14

3.2 Problèmes rencontrés 15

3.3 Problèmes non résolus 15

3.4 Perspectives futures 15

3.5 Synthèse 15

4 Références 16

4.1 Bibliographie 16

4.1.1 Thème : Reconnaissance vocale 16

4.2 Webographie 16

4.2.1 Thème : Reconnaissance vocale 16

4.3 Projet 16

4.3.1 Thème : Reconnaissance vocale 16

5 Annexes 17

5.1 Support numérique (une copie par rapport) 17

# Implémentation SGBD

## Vos choix en tant qu’administrateur de votre SGBD

### Estimation de la taille des données

L'estimation de la taille des données est basée sur une période d'environ 2 à 3 ans. L'unité de stockage utilisée dans ce tableau est le byte. Nous avons pris en compte les informations suivantes venant de la part du client :

* Grand maximum 100 équipes par concours
* 4-7 (max 10) personnes par équipe
* Une dizaine de concours sur 2-3 ans

Estimation moyenne :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table** | **NbTuples** | **Nbchamps** | **Formule Calcul** | **Taille tuple** | **Taille Table** |
| role | 5 | 2 | 5\*4 +6 + 20 + 18 + 15 + 13 | 10/24/22/19/17 | 92 |
| account | 5000 | 9 | 5000\*(7\*100+2\*4) | 708 | 3540000 |
| challenge | 10 | 7 | 10\*(2\*4+100+3\*8+65535) | 65667 | 656670 |
| team | 1000 | 4 | 1000\*(3\*4+100) | 112 | 112000 |
| account\_team | 7000 | 2 | 7000\*(2\*4) | 8 | 56000 |
| challenge\_account | 50 | 2 | 50\*(2\*4) | 8 | 400 |
| solution | 10000 | 9 | 10000\*(3\*4+3\*100+2\*4+8) | 328 | 3280000 |
| data | 20 | 3 | 20\*(2\*4+100) | 108 | 2160 |

Taille des champs utilisés :

|  |  |
| --- | --- |
| **Type** | **Taille** |
| int | 4 bytes |
| float | 4 bytes |
| datetime | 8 bytes |
| text | 65535 bytes |

La taille de la BD s'élève alors à 7647322 bytes, que l'on peut arrondir à 7650000 bytes (7.65 Mb).

Globalement, il n'y a pas de raison de se soucier d'une taille trop importante de données. Le scope de ce projet est bien plus petit que celui de Google Hashcode par exemple, qui s'étend celui-ci sur une échelle mondiale. L'estimation de la taille de la table est grossière dans le sens où une grande partie des champs utilisés sont de type varchar. La taille du champ dépend donc de la donnée qu'il contient. Par contre, dans la plupart des cas, la taille de la base de donnée est supérieure à la taille de son contenu.

### Comptes sur votre SGBD

Choix pour implémenter la sécurité (revoke et grant) : Lien entre vos vues externes (rapport analyse) et les comptes crées.

## Vos choix pour l’optimisation et la concurrence/synchronisation

### Choix d’index pour chaque relation

* Commencer par présenter les choix que vous avez fait pour l’implémentation des relations et des index à leur associer.
* Prenez une de vos relations et montrer l’apport de l’ajout d’un index en étudiant les temps d’exécution d’une requête que vous aurez choisi.

### Argumentation sur la dénormalisation

Étude des performances et choix de dénormalisation (duplication de l'information, fusion de tables...), choix relatifs aux données dérivées ajout de clés numériques, clustérisation…..

### Choix pour la concurrence

Décrire clairement les ajouts de tables/colonnes, …. Que vous avez dû faire pour assurer la concurrence.

### Les procédures stockées et vues

Les procédures stockées et vues mises en places pour optimiser les jointures etc. et gérer la sécurité…..

## Etude d’un plan d’exécution

Isoler une requête compliquée de votre application, ou créez-en une à la main.

1. Présenter la requête (SQL),
2. Votre plan d’exécution en AR
3. Le plan d’exécution généré par le SGBD
4. Commentez clairement ce qui se passe.

## Code SQL dans cet ordre

Attention aux CIs indiquées dans le modèle relationnel : il doit y avoir un commentaire qui indique l’endroit où elles ont été implémentées. **Utiliser leur nom.**

### Code de l’implémentation des tables

### Code des triggers

### Code des procédures stockées et des fonctions implémentés

### Code de création des utilisateurs avec leurs privilèges

### Remarques

Mettre ici toute astuce qui a dû être utilisée parce que le SGBD utilisé ne permet pas une gestion de quelque chose que vous pensiez utiliser : gestion des exceptions etc.

## Caractéristiques SGBD

Complétez les sections suivantes qui permettent de comprendre le SGBD utilisé et ses caractéristiques.

* NOM du SGBD : MySQL
* Version :
* Références d’où proviennent vos commentaires :
  + https://openclassrooms.com/courses/administrez-vos-bases-de-donnees-avec-mysql/triggers

### Explicitez les triggers et leurs caractéristiques

Avec MySQL, il n'est pas possible de générer des erreurs personnalisées. La seule solution pour générer une erreur est de lancer une requête qui va générer une exception.

Il n'est pas possible non plus d'utiliser des transactions ou des requêtes préparées depuis le code d'un trigger. Toutes les procédures ne sont pas appelables depuis un trigger, elles doivent respecter certaines conditions :

* Aucun information ne peut-être retournée au client MySQL mais l'information peut être retournée au trigger grâce aux paramètres INOUT et OUT.
* Les procédures utilisées dans un trigger doivent respecter elles aussi les restrictions des triggers (pas de transaction ni de requête préparée).

La restriction probablement la plus gênante pour ce projet est la suivante : Le code du trigger ne peut modifier la table de laquelle le trigger a été déclenché.

Si un changement de données (modification ou suppression) intervient depuis une clé étrangère (ON DELETE SET NULL dans la table dont un champ est référencé par la clé étrangère), le trigger de la table avec la clé étrangère n'est pas appelé.

### Explicitez les niveaux d’isolations de votre SGBD et le protocole qu’il utilise

### Le support de XML et XSD

Le SGBD MySQL n'offre pas de champ XML spécifique. On doit à la place utiliser un simple champ texte qui contiendra la donnée sous format XML.

### Les types d’index disponibles

Il existe 4 types d'index dans MySQL. Les index simples sont mis sur une ou plusieurs colonnes de type alphanumérique.

# Implémentation SI

## Environnement spécifique de développement

### Framework

#### Spring Session

Du côté serveur, nous avons ajouté Spring Session. Cette librairie permet de facilement gérer les informations des sessions des utilisateurs. Elle peut être utilisée de deux manières : avec des cookies ou avec des tokens. Dans notre cas, nous avons opté pour la méthode avec les tokens car après de nombreuses recherches sur internet, nous avons pu comprendre que c’est la méthode privilégiée lorsque le serveur est utilisé pour mettre à disposition une API REST.

#### Spring Security

Du côté serveur, nous avons également ajouté la librairie Spring Security. Ce framework permet l’authentification des utilisateurs ainsi que la gestion des droits sur les appels à l’API REST. Il a donc été assez facile d’ajouter ses fonctionnalités à notre application. Pour l’authentification des utilisateurs, nous avons donc dû configurer ce framework pour qu’il utilise les utilisateurs et les mots de passe qui sont stockés dans notre base de données. Cela a été facilement réalisé grâce à la création d’un service supplémentaire qui implémente l’interface « UserDetailsService ». Celui-ci permet de retourner les utilisateurs ainsi que leurs rôles. Les mots de passe sont stockés dans la base de données encodés avec la fonction de hachage Bcrypt. Ceci ajoute de la sécurité à notre application.

#### jQuery

Du côté client, nous avons dû ajouter la librairie jQuery afin de pouvoir interagir avec les modals Bootstrap. Cette librairie a été ajouté au projet grâce au gestionnaire de paquets NPM. Elle a également été « typée » ce qui permet une utilisation facile en TypeScript et donc également avec le framework Angular.

#### ngx-localstorage

Du côté client, nous avons également dû ajouter la librairie ngx-localstorage. Cette librairie permet d’interagir facilement avec le « local storage » du navigateur internet. Ceci nous permet de persister le token d’authentification afin de pouvoir le récupérer lors d’un éventuel rafraichissement de la page web.

## Éléments architecturaux

### Couche présentation

Il s’agit de remplir et compléter les informations suivantes :

1. Type de l’IHM

Client mixte.

Etant donné que l'on utilise Angular pour la partie cliente, toute la logique n'est pas uniquement implémentée du côté serveur, on manipule aussi des objets du côté client et des traitements y sont exécutés. Le serveur effectuant aussi des traitements, nous avons conclu à un client mixte. On a choisi cette option pour éviter de surcharger le serveur puisque plusieurs utilisateurs s'y connecteront en même temps. C'est un bon moyen de distribuer le travail sur les machines composants le système.

1. Support du IHM

Navigateur.

Nous avons choisi de réaliser une application web car cela permet d’accéder à notre application sans avoir à installer quoi que ce soit. Ceci permet également à notre application d’être accédée depuis n’importe quels appareils (ordinateur, tablette, smartphone).

Nous pouvons dire que le framework Angular est supporté par les versions les plus récentes des navigateurs internet parce que cela est explicitement écrit dans la documentation officielle disponible à cette page : <https://angular.io/guide/browser-support>

Les versions supportées sont les suivantes :

|  |  |
| --- | --- |
| Navigateur | Version supportées |
| Chrome | La plus récente |
| Firefox | La plus récente |
| Edge | Les deux plus récentes |
| IE | 11, 10, 9 |
| IE Mobile | 11 |
| Safari | Les deux plus récentes |
| iOS | Les deux plus récentes |
| Android | Nougat (7.0)  Marshmallow (6.0)  Lollipop (5.0, 5.1)  KitKat (4.4) |

Nous avons personnellement testé sous :

* IE : version 11.431.16299.0
* Firefox : version 59.0.2
* Chrome : version 66.0.3359

1. Explication sur le choix technologique :

Pour la couche présentation, l’approche utilisée est d’implémenter un client mixte avec le framework Angular parce que cette technologie permet de facilement créer des applications assez conséquentes. Nous avons également choisi d’utiliser cette technologie car nous l’avions déjà testée lors du cours de système d’information 2.

1. Réalisation de l’interface

L’interface IHM est réalisée « from scratch » parce que le framework Angular ne permet pas de générer les interfaces à partir d’une API REST. Cependant, nous nous basons sur la très connue librairie CSS Bootstrap afin de pouvoir développer une application avec un design correct.

### Couche métier

Il s’agit de remplir et compléter les informations suivantes :

1. Inventaire des servlets et des JSP / ASPX et CS / PhP avec un tableau après le diagramme de package avec une explication succincte de chaque classe. Ex. diagramme de package

|  |  |  |
| --- | --- | --- |
| Nom package | Liste des classe | Explications |
|  |  |  |
|  |  |  |
|  |  |  |

### Couche service

Il s’agit de remplir et compléter les informations suivantes :

1. L’accès à la base de données :

Nous avons utilisé le connecteur officiel pour Java fournit par MySQL. Son nom est « mysql-connector-java ». Nous avons utilisé la version 6.0.6 de celui-ci. Comme ce connecteur a été implémenté en Java, il est multiplateforme. Pour le configurer, il suffit d’ajout la ligne ci-dessous dans le fichier application.properties qui permet la configuration du framework Spring :

spring.datasource.url=jdbc:mysql://localhost:3306/hashcodedb?useSSL=false

1. La gestion de la politique et des droits d’accès :

Comme notre base de données sera accédée par une seule application, un utilisateur ayant les droits CREATE/READ/UPDATE/DELETE sera créé et utilisé par celle-ci.

Les droits de création, lecture, modification et suppression des utilisateurs est géré par le framework Spring Security. C’est donc à l’appel sur l’API que le contrôle est effectué. Pour configurer Spring Security, il suffit de créer une classe qui étend WebSecurityConfigurerAdapter et de redéfinir les méthodes configure(…) pour qu’elles utilisent un service qui implémente UserDetailsService afin d’obtenir les utilisateurs et leurs rôles.

1. La gestion des aspects de sécurité :

Pour des questions de sécurité, nous encodons les mots de passe des utilisateurs grâce à la fonction de hachage Bcrypt. Cet encodeur est fourni par Spring Security.

## Éléments technologiques

### Outils de gestion de versionning (CVS / SVN)

Il s’agit de donner les plus amples informations sur le système de gestion de version utilisé dans le cadre de votre projet : L’adresse du serveur, Le client utilisé, Le chemin d’accès, Synthèse sur les avantages et les facilités d’utilisation, Synthèse sur les désavantages et les difficultés d’utilisation.

L'outil de gestion de versionning que nous avons utilisé est git. C'est un outil très complet et efficace mais qui demande aussi un apprentissage en profondeur lorsque la structure du projet se complexifie. Il n'est pas évident non plus de gérer du code avec les fichiers temporaires et de configuration qui changent d'une machine à l'autre. Pour cela, l'utilisation du gitignore est requise.

### Outils de modélisation

Il s’agit de donner les plus amples informations sur l’utilisation de XXXX au niveau :

Tous les diagrammes de modélisation ont été créés grâce au programme Visual Paradigm. Cependant, aucun code n'a été généré à partir d'un outil de modélisation.

1. La génération du code (C#, java, ou PHP)
2. La génération des documents de documentation pour développeurs (Doc, Excel, et PowerPoint)
   1. Ex. de chaque Template
3. Plus d’infos & liens : http://staruml.sourceforge.net/en/

### Outils de documentation du code

Il s’agit de donner des informations sur l’outil(s) de génération de documentation du code :

1. Outil natif à l’environnement de développement
   1. Synthèse sur son utilisation
2. Outil externe. Lequel ? ex. javadoc, ndoc

Nous avons utilisé la javadoc car c'est un outil de documentation très répandu et efficace.

* 1. Synthèse sur son utilisation

Javadoc

1. Ex. de commentaire avec syntaxe complet (class, attribut, commentaires,)
2. Ex. de résultat de cette génération sur un fichier code
3. Plus d’infos & liens :
   1. <http://www.winnershtriangle.com/w/>
   2. <http://www.codeproject.com/KB/macros/kingstools.aspx>
   3. <http://sandcastle.codeplex.com/>

### Outils de journalisation de prototype

Il s’agit de donner des informations sur l’outil(s) de journalisation de prototype.

Nous n'avons utilisé aucun outil de journalisation de prototype

1. Outil natif à l’environnement de développement
   1. Synthèse sur son utilisation
2. Outil externe. Lequel ? ex. log4java
   1. Synthèse sur son utilisation
3. Ex. de résultats commentés
4. Plus d’infos :
   1. <http://logging.apache.org/>
   2. <http://en.wikipedia.org/wiki/Log4j>
   3. http://logging.apache.org/log4net/
   4. <http://lutecefalco.developpez.com/tutoriels/dotnet/log4net/introduction/>
   5. <http://www.dotnetspace.com/articles/general-articles/using-log4net---very-quick-start.html>
   6. http://www.kellermansoftware.com

### Outils de test

Il s’agit de donner des informations sur l’outil(s) de test « Fonctionnelle » et « unitaire ».

Dans le cadre de ce projet intégré, nous n'avons pas eu le temps nécessaire à l'application de tests fonctionnels ou unitaires. Si du temps supplémentaire y avait été consacré, les outils tels que junit ou encore karma auraient pu être utiles.

1. Outil natif à l’environnement de développement
   1. Synthèse sur son utilisation
2. Outil externe. Lequel ? ex. ncover, junit, rational functional Tester
   1. Synthèse sur son utilisation
3. Ex. de résultats commentés
4. Plus d’infos & liens:
   1. <http://fr.wikipedia.org/wiki/Test_unitaire>
   2. <http://en.wikipedia.org/wiki/Functional_test>
   3. <http://en.wikipedia.org/wiki/Unit_test>
   4. <http://www.geekzone.co.nz/vs2008/4819>
   5. <http://seleniumhq.org>

### Outils de gestion de licences

Il s’agit de donner des informations sur l’outil(s) de gestion de licence.

1. Outil natif à l’environnement de développement
   1. Synthèse sur son utilisation
2. Outil externe. Lequel ? ex. softwarekey
   1. Synthèse sur son utilisation
3. Ex. de résultats commentés
4. Plus d’infos & liens:
   1. [http://www.softwarekey.com](http://www.softwarekey.com/)
   2. <http://www.xheo.com/Products/Licensing>
   3. <http://www.mancosoftware.com/licensing/howto.htm>
   4. <http://www.gnu.org/licenses/gpl.html>
   5. <http://www.infralution.com/licensing.html>
   6. <http://www.jproductivity.com/products/protection/developer.htm>
   7. <http://members.chello.at/stefan-schneider/JavaLicenseKit.html>

### Outils d’installation et de packaging

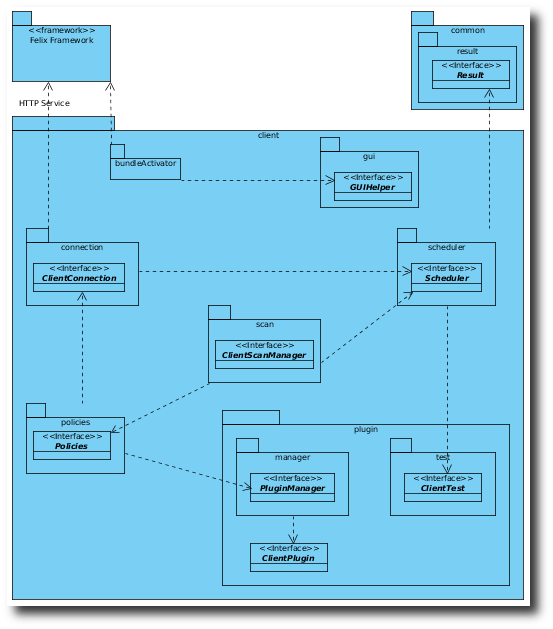
Il s’agit de donner des informations sur l’outil(s) d’installation et de packaging.

1. Outil natif à l’environnement de développement
   1. Synthèse sur son utilisation
2. Outil externe. Lequel ? ex. installshield
   1. Synthèse sur son utilisation
3. Ex. de résultats commentés
4. Plus d’infos & liens:
   1. <http://en.wikipedia.org/wiki/InstallShield>
   2. <http://www.innosetup.com/isinfo.php> (ex. pour ruby et autre)
   3. <http://www.pantaray.com/>

## Diagrammes d’architecture

### Diagramme(s) de package

Utilisé pour séparer le modèle en conteneurs logiques, et décrire leurs interactions à un haut niveau. Il s’agit de fournir un diagramme de paquetage au format UML commentée en détails. Ci-dessous un exemple de diagramme de paquetage.



### Diagramme(s) d’état-transition

Le diagramme d'état-transition, placé sur la page suivante, représente les diverses pages formant notre application web reliées par les événements déclenchés par l'utilisateur (click). Une même page avec un pop-up qui apparaît est considéré comme un nouvel état uniquement si celui-ci demande une action de l'utilisateur (bouton de confirmation "OK" par exemple). Si un message d'information disparaît après un certain timeout, il n'est pas considéré comme un nouvel état.

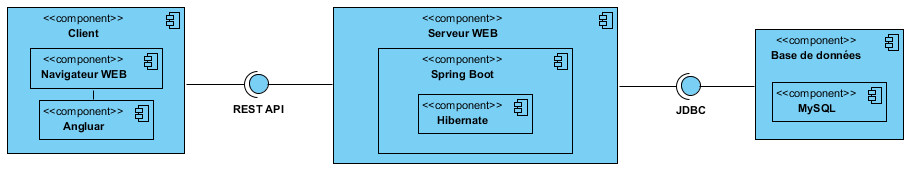
Une autre particularité de notre diagramme est qu'il n'a pas d'état final. Nous avons décidé que notre système n'avait pas d'état final, estimant que l'utilisateur peut réutiliser notre système à tout moment. On aurait pu, si désiré, ajouter un état final par exemple lorsque l'utilisateur ferme le navigateur. On aurait à ce moment-là utilisé un superstate qui aurait conduit à l'état final.

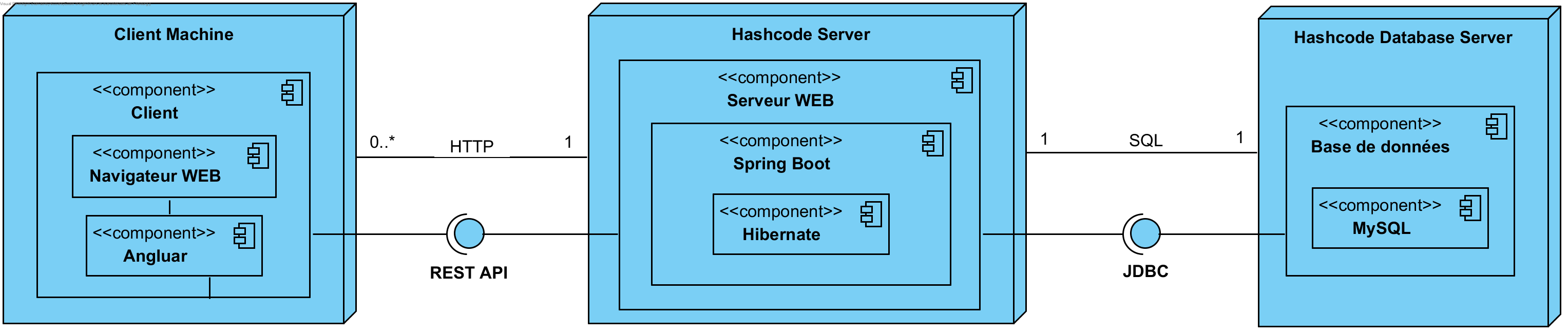
Sachant que la navigation sur notre application web est libre (menu, onglet, ...), nous n'avons pas pu modéliser tous les événements entre les tous états pour une question de lisibilité et d'importance de l'information.

Lorsque un utilisateur est connecté, il a la possibilité de se déconnecter depuis n'importe quelle page, pour autant qu'aucun pop-up ne soit affiché dans celle-ci.

### 

### Diagramme de déploiement

Des changements ont été apportés au diagramme de composants. Angular n'a aucun service ni composant qui se trouve directement sur le serveur. Voilà à quoi ressemble le nouveau diagramme de composants.

Pour ce qui est du diagramme de déploiement, il représente une architecture 3-tiers avec la machine du client (ordinateur/smartphone), le serveur Hashcode (logique métier) et le serveur de base de données Hashcode (persistance des données). Le navigateur utilise le composant Angular pour l'affichage des données. Du côté serveur, le framework spring boot est utilisé avec l'ORM hibernate. Comme base de données, MySQL est implémenté.

## Choix technologique

|  |  |  |  |
| --- | --- | --- | --- |
| **Approche technologique et langages** | **IHM** | **Logique / Middleware** | **Data** |
| JDBC – préciser la version |  | Spring-jdbc:5.0.5 |  |
| AJAX – préciser la plateforme ou les librairies utilisées | / | / | / |
| Web Services – préciser les librairies |  | javax.servlet-api:3.1.0 |  |
| DOM – préciser les librairies | / | Dom4j 1.6.1 | / |
| SAX– préciser les librairies | / | / | / |
| SQL |  |  | MySQL 5.6.30 |
| XML schéma, validation avant stockage dans la base |  | XML Schéma |  |
| Fichiers de configuration : Java EE (web.xml, etc. ) |  | Web.xml |  |
| Fichiers de configuration : XXXX (xxx.xxx, etc.) |  | SecurityConfig.java  SessionConfig.java |  |
| Browser - iexplorer | Version  11.431.16299.0 |  |  |
| Browser - firefox | Version  59.0.2 |  |  |
| Browser - chrome | Version 66.0.3359.139 |  |  |

## Contraintes d’utilisation technologique

|  |  |  |
| --- | --- | --- |
|  | OUI (justifier) | NON (justifier) |
| Utilisation une architecture 3 tiers au minimum | La plupart des applications actuelles fonctionnent de cette manière : partie cliente, partie serveur et partie base de données |  |
| Utilisation des web services dans votre architecture | Le serveur mis en place est de type REST. C'est très rapidement implémenté et c'est ce qui est proposé par défaut dans une application Spring Boot. |  |
| Utilisation des RIA dans votre architecture |  | Notre système requiert l'utilisation d'une base de données et d'un serveur qui ne peuvent être inclus dans une application fonctionnant uniquement dans un navigateur sans aucun besoin d'installation |
| Utilisation au minimum d’une feuille de style XSLT |  | Nous avons pas eu de partie assez conséquente d'XML pour avoir besoin d'utiliser une feuille de style |
| Utilisation au minimum d’une feuille de style XSL-FO |  | Aucun XML conséquent n'est affiché dans un format spécifique à lui |
| Utilisation du SQL | Les requêtes lancées sur notre base de données MySQL utilisent SQL |  |
| Utilisation du XQuery |  | Spring Boot offre déjà un parser DOM. Le document XML n'est pas assez large pour que l'utilisation du langage Xquery en vaille la peine |
| Utilisation au minimum d’un contrôle de validation sur la saisie des données | Un contrôle de saisie de données est effectué dans le formulaire d'inscription ainsi que dans la page de profil |  |
| Utilisation de DOM, SAX ou JDOM, JAXP |  | DOM, le parsing se fait d'un seul coup et on requête ensuite les attributs recherchés. Dom est largement suffisant pour la taille du XML et sa complexité |
|  |  |  |
|  |  |  |

## Outils administratifs

|  |  |  |
| --- | --- | --- |
|  | Oui (justifier) | Non (justifier) |
| Outil de gestion de planning |  | Le planning n'a pas été définit sur papier mais oralement au sein du groupe de travail. |
| Gestion tache, etc. | http://www.clockingit.com/ |  |
|  |  |  |
|  |  |  |

# Conclusion

## Objectifs atteints

La conclusion résume très succinctement ce qui a été réalisé. Cette sous-section se terminera par un tableau de la forme suivante :

|  |  |
| --- | --- |
| Objectifs |  |
| UseCase1 | 100% |
| UseCase2 | 90% |
| …. | 100% |
| Test (functional, unit, etc.) | 100% |
|  |  |

## Problèmes rencontrés

Joé qui emmerde

## Problèmes non résolus

Joé qui emmerde

## Perspectives futures

Joé qui emmerde

Cette sous-section contient des propositions pour des amélioration/évolution futur du produit.

## Synthèse

Cette sous-section contient une courte indication de l’intérêt (ou du non-intérêt) de ce PI sur le plan pédagogique et le point de vue sur l’organisation de ce laboratoire, …

Nous avons bien aimé ce projet dans son ensemble. C’était la première fois que nous pouvions mettre en pratique la matière apprise dans divers cours au sein d’un seul et même projet. Nous avons bien apprécié le fait que nous pouvions choisir les framework avec lesquels nous allions mener à bien ce projet. Cela nous a aussi permis de pouvoir utiliser le framework Spring qui est très populaire dans le monde professionnel mais qu’il n’avait pas été traité durant le cours de systèmes d’information.

Nous jugeons que ce projet intégré a été enrichissant en termes de connaissances. Nous n’avions par exemple jamais mis en place un système d’authentification avec gestion des droits selon les rôles des utilisateurs. Ce projet nous a donc permis de faire cela et de créer une application plus riche et plus complète qu’auparavant. Le sujet qui nous a été imposé était intéressant à réaliser.

Cependant, nous pensons qu’il y a beaucoup trop de modélisation. Nous doutons qu’une entreprise budgétise de quoi faire 21 fiches descriptives, 22 diagrammes de séquence et 22 diagrammes de communication. Cela nous a laissé le sentiment de « faire pour faire » qui nous a malheureusement suivi tout le long du projet. Nous estimons qu’autant de modélisation à prétérité la phase d’implémentation alors qu’il s’agit évidemment de la phase la plus intéressante et qui justifie le fait de faire de la modélisation. Ceci est particulièrement dommageable car nous avons l’impression d’avoir fait que de la modélisation et très peu d’implémentation alors que cette dernière compte double.

Hormis cela, ce projet nous a permis pas d’acquérir des connaissances que nous jugeons utiles pour notre avenir. Nous pensons que ce projet doit continuer à exister mais avec un petit rééquilibrage entre la modélisation et l’implémentation.

# Références

A remplir et modifier selon les besoins du projet

## Bibliographie

### Ex. Thème : Reconnaissance vocale

[B1], SALT Specification 1.0. 15 juillet 2002, Copyright Cisco Systems, Intel, Microsoft, Philips, SpeechWorks

**Votre conseil relativement à cette référence:** bonne référence pour résoudre le problème xxx

[B2], VoiceXML Specification 1.00. 07 mars 2000, Copyright VoiceXML Forum.

**Votre conseil relativement à cette référence:** à éviter les explicitons non claire

## Webographie

### Ex. Thème : Reconnaissance vocale

[S1], SALT: http://www.saltforum.org

**Votre conseil relativement à cette référence**: bonne référence pour résoudre le problème xxx

[S2], VoiceXML: http://www.voicexml.org

**Votre conseil relativement à cette référence:** bonne référence pour résoudre le problème xxx

## Projet

### Ex. Thème : Reconnaissance vocale

[P1], DynaVoice, Cédric Delléa. 2003, Projet de travaile de diplôme HES à l’HEVs

**Votre conseil relativement à cette référence**: bonne référence pour résoudre le problème xxx

[P2], Etude d’applications VoiceXML dans le cadre de PME avec des petites succursales, Christian Bardet, Sacao SA 2003, Projet de travail de diplôme EduSwiss

**Votre conseil relativement à cette référence**: bonne référence pour résoudre le problème xxx

# Annexes

## Support numérique (une copie par rapport)

[1]répertoire 1 : Répertoire de travail du PI

* Rapport, présentation, etc., Procès verbaux, les emails, etc.

[2]répertoire 2 : Développement logiciels

* Sources, Librairies (spécifique, communes), Serveurs ou autres produits.