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 : 18.05.2018

Superviseurs : Prof. Houda Chabbi Drissi

Prof. Pierre Kuonen

Prof. Omar Abou Khaled

Client: Prof. Pierre Kuonen

Table des matières

1 Implémentation SGBD

1.1 Vos choix en tant qu'administrateur de votre SGBD

1.1.1 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.

1.1.2 Comptes sur votre SGBD

Cette partie est documentée dans le chapitre 1.4.4.

1.2 Vos choix pour l'optimisation et la concurrence/synchronisation

1.2.1 Choix d'index pour chaque relation

Nous n'avons pas implémenté d'index dans notre base pour le moment. Actuellement, nous n'avons pas assez de données pour démontrer l'apport d'un index dans le temps d'exécution, nous attendons d'avancer un peu plus conséquemment le développement avant d'en rajouter un ou plusieurs.

Dans notre projet, globalement, il serait intéressant et bénéfique d'utiliser un index dans les champs dates de la table challenges. Souvent, les comparaisons y sont utilisées et l'indexation des colonnes seraient profitable car une fois triée la recherche sera simplifiée.

1.2.2 Argumentation sur la dénormalisation

Aucune démoralisation n'a été faite.

Une modification peut être intéressante dans notre projet. Les Rôles sont définis dans une autre table et nécessitent une jointure à chaque fois que l'on souhaite récupérer les informations d'un account. Plutôt que de faire une table avec une FK liée, on pourrait directement utiliser le champ de la FK avec le type de ROLE et supprimer la table respective. Les valeurs seraient dupliquées mais économiseraient une jointure à chaque requête sur le compte.

1.2.3 Choix pour la concurrence

Afin de gérer la concurrence, deux champs de types TIMESTAMP supplémentaires ont été ajoutés à chaque table. Le premier préserve la date et l'heure de création de l'enregistrement et la deuxième conserve la date et l'heure de la dernière modification. Ces champs sont automatiquement remplis et mis à jour.

Lorsqu'une requête de modification ou de suppression d'un enregistrement est exécutée, la colonne conservant la dernière modification est comparée entre l'enregistrement récupéré et l'enregistrement présent dans la table.

Si les valeurs sont les mêmes, il n'y a pas eu de changement entre la requête récupérant l'enregistrement et celle modifiant / supprimant, donc elle peut s'exécuter sans problème.

Si les valeurs ne sont pas les mêmes, il y a alors un problème de concurrence qu'il faut relever, le traitement de ce problème varie.

```
ALTER TABLE [nom de la table]
ADD COLUMN created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON CREATE
CURRENT_TIMESTAMP;

ALTER TABLE [nom de la table]
ADD COLUMN updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP;
```

1.2.4 Les procédures stockées et vues

Des procédures stockées ont été implémentées mais pas dans le but d'économiser des jointures. Une procédure stockée qui récupère toutes les informations d'un concours peut être utile car elle économiserait différente jointure. Aucune vue n'a été utilisée.

1.3 Etude d'un plan d'exécution

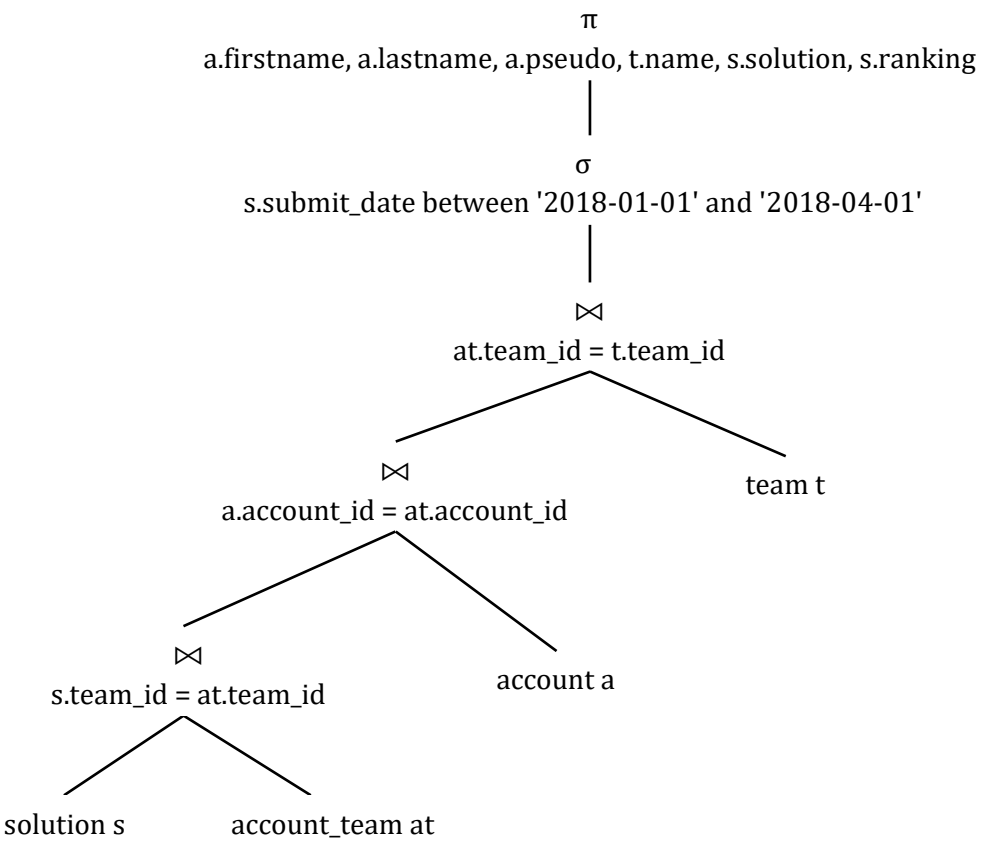
La requête présentée ci-dessous ne sera pas utilisée dans notre projet. Elle sert juste à générer un plan d'exécution intéressant à analyser.

```
select account.firstname, account.lastname, account.pseudo, team.name, solution.solution, solution.ranking
from account inner join account_team
on account.account_id = account_team.account_id
inner join solution
on solution.team_id = account_team.team_id
inner join team
on account_team.team_id = team.team_id
where solution.submit_date between '2018-01-01' AND '2018-04-01';
```

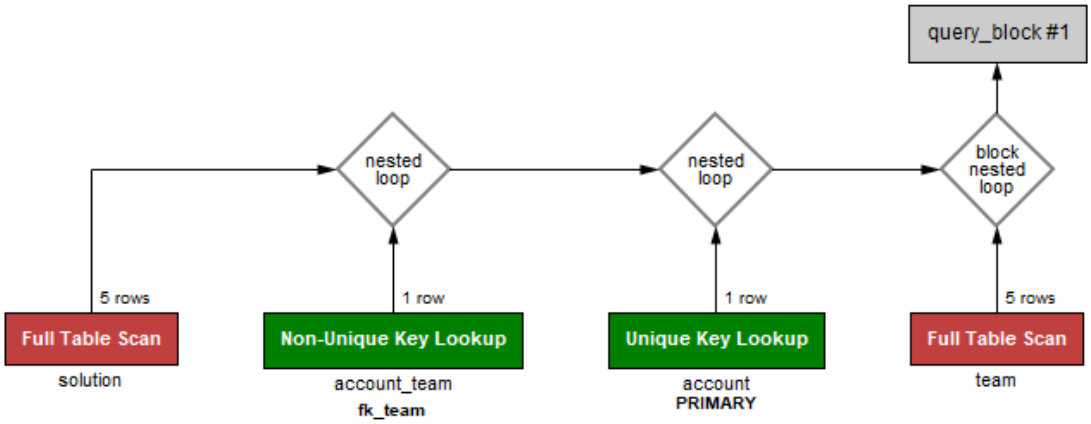
Voici le résultat que l'on obtient après l'exécution de la requête précédemment présentée.

	firstname	lastname	pseudo	name	solution	ranking
	Valdimir	Meier	valdm	L'homme de l'est	lien.vers.solution	6
	Valdimir	Meier	valdm	L'homme de l'est	lien.vers.solution	7.5

On peut voir ci-dessous notre plan d'exécution en AR.



Ci-dessous, le plan d'exécution généré par le SGBD.



Un full scan est lancé sur la table solution car aucune entrée particulière n'est recherchée. Par contre, le Non-Unique Key Lookup s'explique du fait que la clé étrangère fk_team peut apparaître plusieurs fois dans la table account_team et que l'on effectue une recherche sur cette clé. Le Unique Key Lookup s'explique du fait que l'on recherche la clé primaire d'un account que l'on sait unique. Pour le dernier full scan, nous n'avons pas réussi à en déterminer la raison.

1.4 Code SQL dans cet ordre

Contrainte d'intégrité

- **C1 :** Afin de pouvoir identifier le compte de chaque utilisateur, les pseudos doivent être unique dans la base de données.
- **C2 :** Afin de pouvoir identifier chaque utilisateur grâce à leur token, ceux-ci seront unique dans la base de données.
- **C3 :** Afin de pouvoir garder les concours équitables, les organisateurs ne pourront pas participer à leurs propres concours.
- **C4 :** Afin de pouvoir préserver une chronologie logique, la date de fin du concours devra être plus ancienne que la date de début.
- **C5 :** Afin de pouvoir préserver une chronologie logique, la date du début du concours devra être plus ancienne que la date de la fin des inscriptions.
- **C6 :** Afin de pouvoir garantir qu'une solution soit déposée dans les délais, la date de la soumission d'une solution devra être plus ancienne que la date de début du concours.
- **C7 :** Afin de pouvoir garantir qu'une solution soit déposée dans les délais, la date de la soumission d'une solution devra être plus récente que la date de fin du concours.
- **C8 :** Afin de pouvoir prévenir que les utilisateurs puissent uniquement participer à un concours donné avec une et une seule équipe, une vérification sera effectuée.

- **C9 :** Afin de pouvoir garantir qu'une solution soit déposée par une personne qui fasse partie de l'équipe à laquelle cette solution est liée, une vérification sera effectuée.

Contraintes relationnelles

On part du principe que par défaut tous les champs du modèle sont not null.

- **CR1 :** Le champ name dans la table « role » est de type {Admin, Organisateur validé, Organisateur en attente, User validé, User en attente}
- **CR2 :** Un enregistrement dans la tableau team ne peut pas exister sans challenge
- **CR3 :** Les champs token et image de la table « Account » sont facultatifs
- **CR4 :** Le champ ranking de la table « Solution » est facultatif
- **CR5 :** Un team ne peut pas exister sans « Account », en cas d'ajout d'une team un enregistrement doit être ajouté dans la table «account-team »
- **CR6 :** Un concours ne peut pas exister sans minimum un organisateur, lors de l'ajout d'un concours un enregistrement doit être ajouté dans la table « challenge-organizer »
- **CR7 :** Il faut vérifier que le leader d'une équipe est membre de cette équipe. Il doit exister dans la table « account-team »

Pour information, la contrainte relationnelle CR1 n'a volontairement pas été implémentée. La table Role n'étant jamais insérée, modifiée ou supprimée, elle nous a semblée inutile.

La situation relevée par la contrainte d'intégrité CR7 est rendu impossible par la procédure stockée « create_team » (inutile de vérifier car la procédure rajoute directement dans les deux tables concernées).

Lorsque qu'une personne dépose une solution, c'est spécifiquement pour son équipe. Il n'a pas la possibilité de choisir. Pour cette raison, la contrainte C9 nous a semblé peu utile à être implémentée.

1.4.1 Code de l'implémentation des tables

```
USE `hashcodedb`;

CREATE TABLE IF NOT EXISTS `hashcodedb`.`role` (
  `role_id` INT         NOT NULL AUTO_INCREMENT,
  `name`    VARCHAR(30) NOT NULL,
  PRIMARY KEY (`role_id`)
   -- CHECK (name IN ('Admin', 'validated_organizer',
'waiting_organizer', 'validated_user', 'waiting_user')) -- Contrainte
relationnelles CR1

)
  ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `hashcodedb`.`account` (
  `account_id` INT          NOT NULL AUTO_INCREMENT,
  `firstname`  VARCHAR(100) NOT NULL,
  `lastname`   VARCHAR(100) NOT NULL,
  `email`      VARCHAR(100) NOT NULL,
```

```
  `pseudo`      VARCHAR(100) NOT NULL UNIQUE, --  UNIQUE : Contrainte
intégrité C1
  `password`    VARCHAR(100) NOT NULL,
  `token`       VARCHAR(100) NULL UNIQUE, -- NULL : Contrainte
relationnel CR3  + UNIQUE : Contrainte intégrité C2
  `image`       LONGTEXT     NULL, -- NULL : Contrainte relationnel
CR3
  `role_id`     INT,
  PRIMARY KEY (`account_id`),
  CONSTRAINT `fk_role` FOREIGN KEY (`role_id`) REFERENCES `role`
(`role_id`)
)
  ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `hashcodedb`.`challenge` (
  `challenge_id`     INT          NOT NULL AUTO_INCREMENT,
  `name`             VARCHAR(100) NOT NULL,
  `nb_teams`         INT          NOT NULL,
  `inscription_date` DATETIME     NOT NULL,
  `begin`            DATETIME     NOT NULL,
  `end`              DATETIME     NOT NULL,
  `media_xml`        TEXT         NOT NULL,
  PRIMARY KEY (`challenge_id`)
  CHECK (inscription_date<begin),  -- Contrainte intégrité C5
  CHECK (begin<end) -- Contrainte intégrité C4
)
  ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `hashcodedb`.`challenge_account` (
  `challenge_id` INT NOT NULL,
  `account_id`   INT NOT NULL,
  PRIMARY KEY (`challenge_id`, `account_id`),
  CONSTRAINT `fk_challenge` FOREIGN KEY (`challenge_id`) REFERENCES
`challenge` (`challenge_id`),
  CONSTRAINT `fk_organizer` FOREIGN KEY (`account_id`) REFERENCES
`account` (`account_id`)
);

CREATE TABLE IF NOT EXISTS `hashcodedb`.`team` (
  `team_id`      INT          NOT NULL AUTO_INCREMENT,
  `name`         VARCHAR(100) NOT NULL,
  `challenge_id` INT          NOT NULL, -- FK Not Null : Contrainte
intégrité CR2

  `leader_id`    INT          NOT NULL,
  PRIMARY KEY (`team_id`),
  CONSTRAINT `fk_challenge2` FOREIGN KEY (`challenge_id`) REFERENCES
challenge (`challenge_id`),
  CONSTRAINT `fk_leader` FOREIGN KEY (`leader_id`) REFERENCES account
(`account_id`)
);

CREATE TABLE IF NOT EXISTS `hashcodedb`.`account_team` (
  `account_id` INT NOT NULL,
  `team_id`    INT NOT NULL,
  PRIMARY KEY (`account_id`, `team_id`),
  CONSTRAINT `fk_account` FOREIGN KEY (`account_id`) REFERENCES
`account` (`account_id`),
```

```
  CONSTRAINT `fk_team` FOREIGN KEY (`team_id`) REFERENCES `team`
(`team_id`)
);

CREATE TABLE IF NOT EXISTS `hashcodedb`.`solution` (
  `solution_id` INT          NOT NULL AUTO_INCREMENT,
  `name`        VARCHAR(100) NOT NULL,
  `language`    VARCHAR(100) NOT NULL,
  `solution`    VARCHAR(100) NOT NULL,
  `version`     FLOAT        NOT NULL,
  `ranking`     FLOAT        NULL, -- CR4 : Un ranking null est
considéré comme pas évalué
  `submit_date` DATETIME     NOT NULL,
  `account_id`  INT          NOT NULL,
  `team_id`     INT          NOT NULL,
  PRIMARY KEY (`solution_id`),
  CONSTRAINT `fk_account_team` FOREIGN KEY (`account_id`, `team_id`)
REFERENCES `account_team` (`account_id`, `team_id`)
);

CREATE TABLE IF NOT EXISTS `hashcodedb`.`data` (
  `data_id`      INT          NOT NULL AUTO_INCREMENT,
  `file`         VARCHAR(100) NOT NULL,
  `challenge_id` INT          NOT NULL,
  PRIMARY KEY (`data_id`),
  CONSTRAINT `fk_challenge3` FOREIGN KEY (`challenge_id`) REFERENCES
`challenge` (`challenge_id`)
);

-- Tables needed for spring session

CREATE TABLE IF NOT EXISTS `hashcodedb`.`spring_session` (
  `primary_id`            CHAR(36) NOT NULL,
  `session_id`            CHAR(36) NOT NULL,
  `creation_time`         BIGINT   NOT NULL,
  `last_access_time`      BIGINT   NOT NULL,
  `max_inactive_interval` INT      NOT NULL,
  `expiry_time`           BIGINT   NOT NULL,
  `principal_name`        VARCHAR(100),
  CONSTRAINT `spring_session_pk` PRIMARY KEY (`primary_id`)
)
  ENGINE = InnoDB;

CREATE UNIQUE INDEX `spring_session_ix1`
  ON `spring_session` (`session_id`);
CREATE INDEX `spring_session_ix2`
  ON `spring_session` (`expiry_time`);
CREATE INDEX `spring_session_ix3`
  ON `spring_session` (`principal_name`);

CREATE TABLE IF NOT EXISTS `hashcodedb`.`spring_session_attributes` (
  `session_primary_id` CHAR(36)     NOT NULL,
  `attribute_name`     VARCHAR(200) NOT NULL,
  `attribute_bytes`    BLOB         NOT NULL,
  CONSTRAINT `spring_session_attributes_pk` PRIMARY KEY
(`session_primary_id`, `attribute_name`),
  CONSTRAINT `spring_session_attributes_fk` FOREIGN KEY
(`session_primary_id`) REFERENCES `spring_session` (`primary_id`)
    ON DELETE CASCADE
)
```

```
  ENGINE = InnoDB;

CREATE INDEX `spring_session_attributes_ix1`
  ON `spring_session_attributes` (`session_primary_id`);

-- Minimum data

INSERT INTO role (name) VALUES ('ADMIN');
INSERT INTO role (name) VALUES ('VALIDATED_ORGANIZER');
INSERT INTO role (name) VALUES ('PENDING_ORGANIZER');
INSERT INTO role (name) VALUES ('VALIDATED_USER');
INSERT INTO role (name) VALUES ('WAITING_USER');
```

1.4.2 Code des triggers

```
-- Containte C3

delimiter |
CREATE TRIGGER C3
BEFORE INSERT ON account_team
FOR EACH ROW

 BEGIN

          DECLARE v_firstname, v_lastname, v_email varchar(50);
    DECLARE idChallenge INT;
    DECLARE otherIdAccount INT;
          Select team.challenge_id
          INTO idChallenge
    FROM team
          inner join challenge on  team.challenge_id = challenge.challenge_id
          where team.team_id = NEW.team_id;

          Select firstname, lastname, email
    INTO v_firstname, v_lastname, v_email
    FROM account
    where account_id = NEW.account_id;

    Select account_id
    INTO otherIdAccount
    FROM account
    where account_id != NEW.account_id
                    and firstname = v_firstname
                    and  lastname = v_lastname
        and email = v_email;

          IF  EXISTS (Select * FROM challenge_account where
challenge_account.challenge_id=idChallenge and challenge_account.account_id=otherIdAccount)
    THEN
                    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Un organisateur ne peut pas
participer à son propre concours';
    END IF;
END |

delimiter ;
```

```
-- Containte C6 ET C7

delimiter |
CREATE TRIGGER C6C7
BEFORE INSERT ON solution
FOR EACH ROW
 BEGIN
          DECLARE  v_date_begin DATETIME;
          DECLARE v_date_end DATETIME;

    SELECT challenge.begin, challenge. end
    INTO v_date_begin, v_date_end
        FROM challenge
      inner join team on challenge.challenge_id = team.challenge_id
      inner join solution on  team.team_id = solution.team_id
      where team.team_id = NEW.team_id
      group by begin;

    IF(v_date_begin > NEW.submit_date) THEN
    -- La date de début du concours est plus récente que la date de soumission de la solution
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =  'Il est trop tot pour soumettre une solution';
```

```
    END IF;

    IF(v_date_end < NEW.submit_date) THEN
    -- La date de fin du concours est plus récente que la date de soumission de la solution
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Il est trop tard pour soumettre une solution';

    END IF;
END |

delimiter |
```

```
-- Contrainte C8
delimiter |
CREATE TRIGGER C8
BEFORE INSERT ON account_team
FOR EACH ROW
 BEGIN
        DECLARE  idChallenge INT;

    SELECT challenge_id
    INTO idChallenge
        FROM team
      where team.team_id = NEW.team_id;

      IF EXISTS(Select * from team inner join account_team on
team.team_id = account_team.team_id where
team.challenge_id=idChallenge and account_team.account_id =
NEW.account_id)
      THEN
                        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
'L\'utilisateur participe d�j� au concours';
    END IF;
END |

delimiter ;
```

1.4.3 Code des procédures stockées et des fonctions implémentées

Aucune fonction n'a été implémentée.

```
-- Procedure stocker pour CR5
DELIMITER |

CREATE PROCEDURE create_team  (name_team VARCHAR(100), challenge_id
INT, leader_id INT)
BEGIN

INSERT INTO team (`name`, `challenge_id`, `leader_id`) VALUES
(name_team, challenge_id, leader_id);
SELECT team_id INTO @idTeam
FROM team
Where `name` = name_team and  `challenge_id`=  challenge_id and
leader_id=  `leader_id`  ;
INSERT INTO account_team ( `account_id`, `team_id`)
VALUES (leader_id, @idTeam);

END |

DELIMITER ;
```

```
-- Procedure stocker pour CR6
CREATE PROCEDURE create_challenge AS
DELIMITER |

CREATE PROCEDURE create_challenge  (name_challenge VARCHAR(100),
nb_teams_challenge INT,
        inscription_date DATETIME, begin_challenge DATETIME,
end_challenge DATETIME, media_xml TEXT, id_organizer INT)
BEGIN
INSERT INTO challenge ( `name`, `nb_teams`, `inscription_date`,
`begin`, `end`, `media_xml`)
VALUES (name_challenge, nb_teams_challenge, inscription_date,
begin_challenge, end_challenge, media_xml);
SELECT challenge_id INTO @idChallenge
FROM challenge
WHERE `name` = name_challenge and  `nb_teams` = nb_teams_challenge
                                and  `inscription_date` =
inscription_date and `begin` = begin_challenge
                and `end` = end_challenge and `media_xml` =
media_xml;
INSERT INTO challenge_account (`challenge_id`, `account_id`) VALUES
(@idChallenge, id_organizer);
END |
DELIMITER ;
```

1.4.4 Code de création des utilisateurs avec leurs privileges

L'utilisateur est créé directement avec le framework, nous n'avons pas encore implémenté les privilèges car nous souhaitions éviter les restrictions lors du développement. Cependant, l'idée voulue pour les privilèges est, afin de respecter les contraintes CR5 et CR6, qu'il faut désactiver l'ajout dans les tables challenge et team pour qu'on soit obligé d'utiliser les procédures stockées.

```
GRANT Insert, Select, Update, Delete ON `hashcodedb`.account TO
'user'@'localhost'  ;
GRANT Insert, Select, Update, Delete ON `hashcodedb`.account_team TO
'user'@'localhost' ;
GRANT Insert, Select, Update, Delete ON
`hashcodedb`.challenge_account TO 'user'@'localhost' ;
GRANT Insert, Select, Update, Delete ON `hashcodedb`.data TO
'user'@'localhost' ;
GRANT Insert, Select, Update, Delete ON `hashcodedb`.solution TO
'user'@'localhost' ;
GRANT Insert, Select, Update, Delete ON `hashcodedb`.role TO
'user'@'localhost' ;
GRANT Insert, Select, Update, Delete ON `hashcodedb`.solution TO
'user'@'localhost' ;
GRANT Select, Update, Delete ON `hashcodedb`.challenge TO
'user'@'localhost' ;
GRANT Select, Update, Delete ON `hashcodedb`.team TO
'user'@'localhost' ;

GRANT Execute On Procedure create_team TO 'user'@'localhost' ;
GRANT Execute On Procedure  create_challenge TO 'user'@'localhost' ;
```

1.5 Caractéristiques SGBD

- NOM du SGBD : MySQL
- Version : 5.7
- Références d'où proviennent vos commentaires :
  - https://openclassrooms.com/courses/administrez-vos-bases-de-donnees-avec-mysql/triggers
  - https://www.supinfo.com/articles/single/1689-transactions-niveaux-isolation-mysql

1.5.1 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 :

- Aucune 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.

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

Quatre niveaux d'isolation sont disponibles avec MySQL :

- Serializable : mode d'isolation le plus haut qui offre un équivalent sériel. Il résoud tous les problèmes mais rend l'exécution beaucoup plus lente.
- Repeatable Read : mode d'isolation qui résoud la lecture sale, la lecture non reproductible mais pas la lecture fantôme.
- Committed Read : mode d'isolation qui résoud le problème de lecture sale.
- Uncommitted Read : mode d'isolation le plus bas qui expose le SGBD à tous les riques tels que la lecture fantôme, la lecture sale et la lecture non reproductible.

Le niveau d'isolation par défaut de MySQL est du repeatable read.

1.5.3 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.

La validation du fichier XML se fait sur le serveur avec un fichier XSD.

1.5.4 Les types d'index disponibles

Il existe 4 types d'index dans MySQL :

- `UNIQUE` : permet d'assurer qu'il n'y aura jamais de doublons dans la colonne
- `INDEX` : permet d'indexer des colonnes qui sont susceptibles d'avoir des doublons
- `FULLTEXT` : permet d'effectuer des recherches efficaces dans du texte. Il peut être utilisé qu'avec des champs de type `CHAR`, `VARCHAR` et `TEXT`.
- `SPATIAL` : permet d'indexer des données spatiales tels que des points, des lignes, des polygones,…

2 Implémentation SI

2.1 Environnement spécifique de développement

2.1.1 Framework

2.1.1.1 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.

2.1.1.2 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.

2.1.1.3 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.

2.1.1.4 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.

2.2 Éléments architecturaux

2.2.1 Couche présentation

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.

2. 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

3. 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 systèmes d'information 2.

4. 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.

2.2.2 Couche métier

Dans cette partie, nous affichons sous forme d'un tableau la structure du projet en terme de dossiers. Une explication par type de classe est donnée.

Nom package	Liste des classe	Explications
Frontend		
app	app.component.ts app.module.ts app-routing.module.ts	app.component.ts est le composant principal du système. Il est comparable au Main d'un programme java standard. app.module.ts s'occupe d'énumérer tous les composants, services et modules dont l'application a besoin. app-routing.module.ts est un controlleur responsable de la gestion des routes pour la redirection vers les bonnes vues d'Angular.
component	admin.component.ts challenges-details.component.ts challenges.component.ts challenges-list.component.ts confirm.component.ts confirm.component.spec.ts profile.component.ts profile.component.spec.ts header.component.ts home.component.ts person.component.ts	Ces "classes" sont tous les composants qui forment les pages de l'application. Une page peut contenir plusieurs composants. Certains composants sont utilisés dans plusieurs pages comme par exemple le composant header.
model	Account.ts Challenge.ts	Les modèles sont des classes manipulables par les

	Role.ts Person.ts Team.ts	services et les composants d'Angular. Celui-ci est capable de travailler avec des objets.
service	account.service.ts authentication.service.ts challenge.service.ts confirm.service.ts person.service.ts	Les services sont utilisés par les composants pour accéder à l'API REST exposée par le serveur Hashcode. Ajax est utilisée par ces services.
interceptor	authentication.interceptor.ts	Cette classe sert à intercepter chaque requête http lancée au sein de l'application pour y ajouter dans l'entête un token.
helper	zip-collector.helper.ts	Cette classe permet de récupérer les données d'un concours et de les placer dans un zip par la suite téléchargeable par l'utilisateur.
Backend		
src	application.java	C'est le point d'entrée du serveur, similaire au composant app de la partie frontend.
service	ITeamService.java ISolutionService.java IChallengeService.java IAccountService.java	Ces interfaces ont été créées pour une éventuelle extension du projet.
service.impl	AccountDetailsService.java AccountService.java ChallengeService.java SolutionService.java TeamService.java	Les services implémentant les interfaces ci-dessus permettent de faire le lien avec les repositories.
repository	AccountRepository.java ChallengeRepository.java SolutionRepository.java TeamRepository.java	Les repositories sont des classes formant l'ORM. Ce sont ces classes qui s'occupent d'accéder à la base de données et travaillent avec des objets.
model	Account.java Challenge.java Role.java Solution.java Team.java	Les modèles se sont les classes pour les données qui passent de la vue à la base de données. Ces modèles ont le même rôle que pour la partie frontend.
controller	AccountController.java AuthenticationController.java ChallengeController.java FileController.java SolutionController.java TeamController.java	Les controlleurs, de la même manière que le composant app-routing.module.ts, s'occupent de la route de chaque requête http reçue par le serveur.

config	SecurityConfig.java SessionConfig.java	La classe SecurityConfig s'occupe de l'authentification, la sécurité des requêtes http, le chiffrement de mots de passes et les droits d'accès. La classe SessionConfig s'occupe elle du token pour la session.
constante	RoleConst.java	Cette classe sert à économiser les requêtes en exposant les différents rôles disponibles au lieu de devoir les chercher dans la base de données.
helper	XSDValidatorHelper.java	Cette classe permet de vérifier la validité d'une donnée XML.

2.2.3 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'ajouter 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`

2. 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 sont gérés 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.

3. 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.

2.3 Éléments technologiques

2.3.1 Outils de gestion de versionning (CVS / SVN)

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.

2.3.2 Outils de modélisation

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. Nous avons également utilisé DBMain pour les schémas de la base de données.

2.3.3 Outils de documentation du code

Pour la documentation côté serveur, nous avons utilisé JavaDoc car il s'agit de l'outil de documentation officiel pour la langage Java. Voici comment documenter une méthode grâce à JavaDoc :

```java
/**
 * Cette méthode permet de recupérer un compte grâce à son pseudo.
 *
 * @param pseudo Pseudo du compte recherché
 * @return Objet Account contenant les informations relatives au compte
 */
@Override
public Account getAccountByPseudo(String pseudo) {
  return accountRepository.findByPseudo(pseudo);
}
```

Voici le résultat après la génération de la documentation :

getAccountByPseudo

```
public Account getAccountByPseudo(java.lang.String pseudo)
```

Cette méthode permet de recupérer un compte grâce à son pseudo.

Specified by:

`getAccountByPseudo in interface IAccountService`

Parameters:

`pseudo - Pseudo du compte recherché`

Returns:

`Objet Account contenant les informations relatives au compte`

Pour la documentation côté client, nous avons utilisé TypeDoc car il s'agit de l'outil de documentation officiel pour la langage TypeScript. Voici comment documenter une méthode grâce à TypeDoc :

```typescript
/**
* Cette méthode permet de récupérer l'image de profil grâce à son ID de compte.
* @param {number} accountId ID du compte duquel l'image doit être récupérée
* @returns {string} Image de profil au format base64

*/
```

```
public getProfilePicture(accountId: number): string {
  const account: Account = this.getAccountFromAccountId(accountId);
  if (account !== null && account.image !== null) {
    return account.image;
  }
  return 'assets/default_profile_picture.png';
}
```

Voici le résultat après la génération de la documentation :

getProfilePicture

`getProfilePicture(accountId: number): string`

Defined in component/challenge-details/challenge-details.component.ts:55

Cette méthode permet de récupérer l'image de profil grâce à son ID de compte.

Parameters

- **accountId:** *number*
  ID du compte duquel l'image doit être récupérée

Returns *string*

Image de profil au format base64

2.3.4 Outils de journalisation de prototype

Nous n'avons utilisé aucun outil de journalisation de prototype par nous-même. Le framework Spring implémente déjà une variante de Commons Logging API. Log4j et slf4j sont utilisés dans ce contexte.

2.3.5 Outils de test

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.

2.3.6 Outils de gestion de licences

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

Dans le cadre de ce projet intégré, nous n'avons pas eu le temps nécessaire pour s'intéresser à ce sujet en particulier. Les licences des logiciels utilisés pour la réalisation de ce projet sont toutes fournies personnellement par le compte de la HEIA-FR. Comme nous ne sommes que trois sur un projet à court terme, il n'y a pas de raison suffisante pour utiliser un outil de gestion de licences. Ce genre d'outil est surtout utile pour les grandes entreprises qui gèrent beaucoup d'employés ayant besoin de licences de logiciels.

2.3.7 Outils d'installation et de packaging

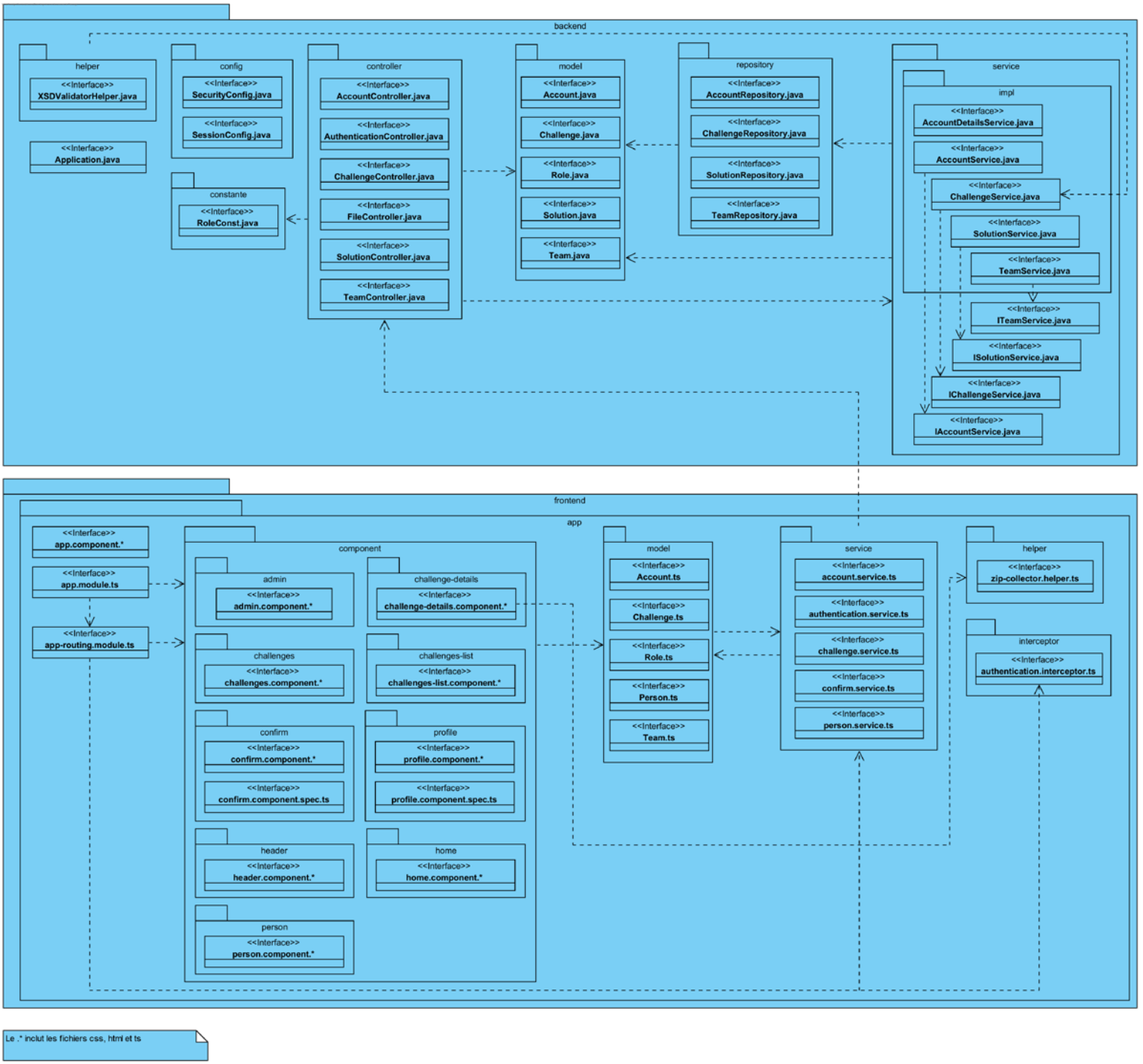
Nous prévoyons d'utiliser docker comme conteneur de notre projet pour qu'il puisse tourner sur les machines requises. A l'heure actuelle, nous ne l'avons pas encore

implémenté. Bien que ce ne soit pas réellement un outil d'installation, il permet de faire tourner sur une autre machine un projet local sans y passer trop de temps.

2.4 Diagrammes d'architecture

2.4.1 Diagramme(s) de package

Le diagramme de paquetage de la page suivante sépare le projet en deux parties : le frontend et le backend. Les différentes classes utilisées sont organisées dans des dossiers définissant les fonctions des classes. La base de données par contre n'apparaît pas dans ce diagramme.



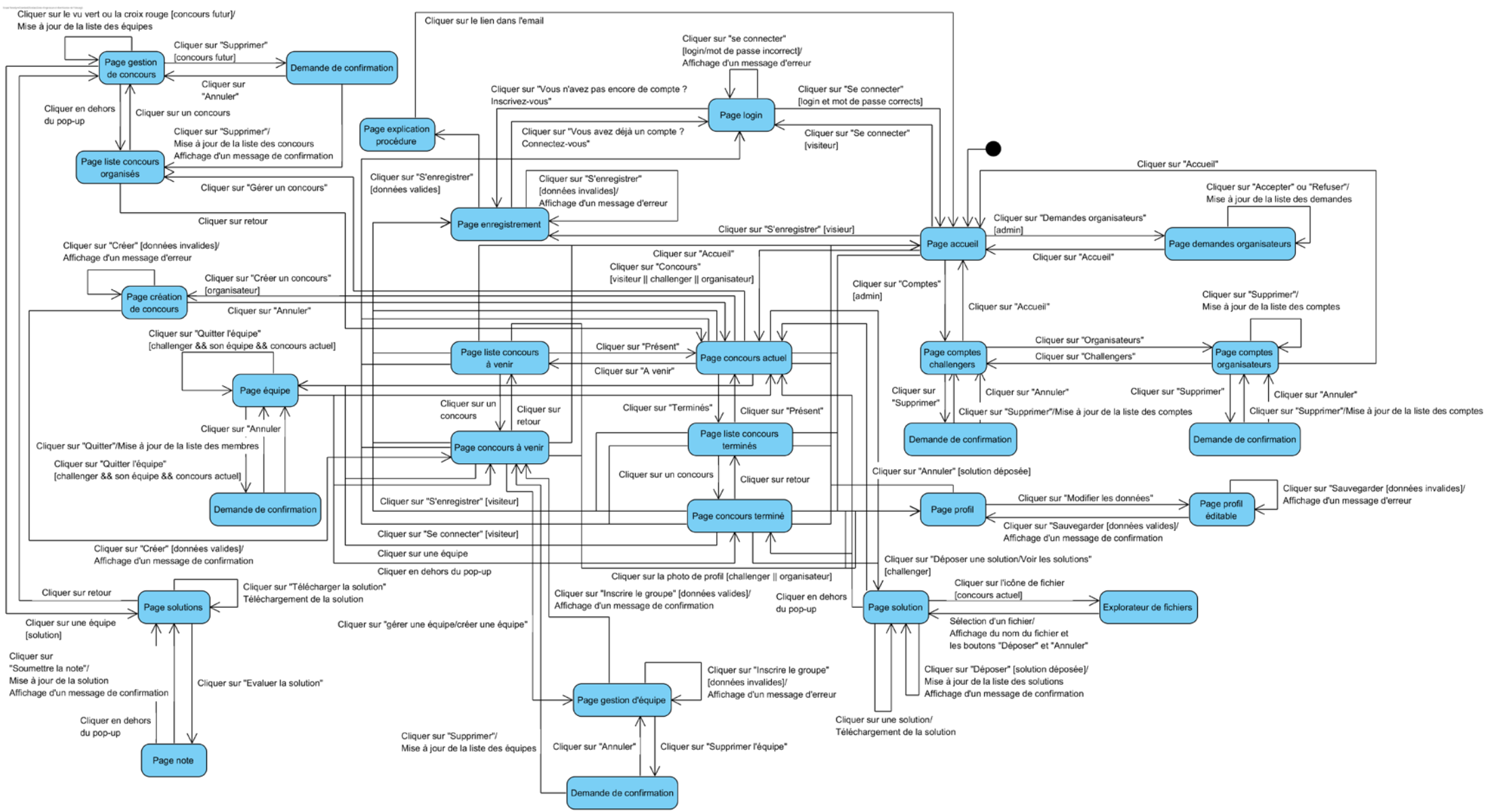
2.4.2 Diagramme(s) d'état-transition

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 (page d'accueil).

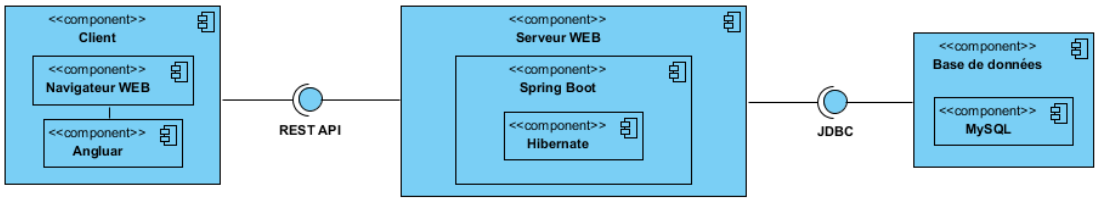
Sachant que la navigation sur notre application web est libre (menu, onglet, ...), nous n'avons pas pu modéliser tous les événements entre les tous états pour une question de lisibilité et d'importance de l'information.

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.

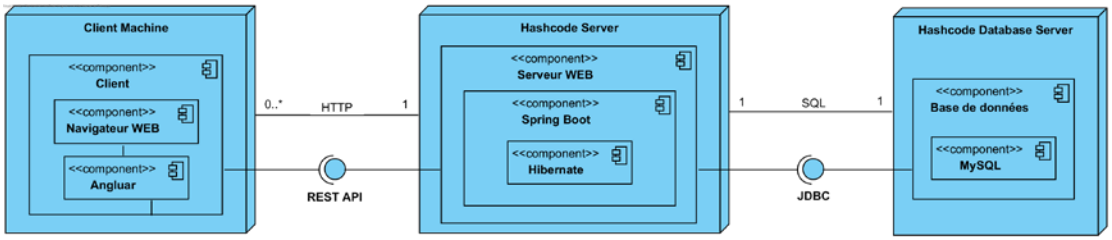

Cliquer sur le vu vert ou la croix rouge [concours futur]/
Mise à jour de la liste des équipes
Page gestion de concours
Cliquer sur "Supprimer" [concours futur]
Demande de confirmation
Cliquer sur "Annuler"
Cliquer en dehors du pop-up
Cliquer sur un concours
Cliquer sur "Supprimer"/
Mise à jour de la liste des concours
Affichage d'un message de confirmation
Page liste concours organisés
Cliquer sur "Gérer un concours"
Cliquer sur retour
Cliquer sur le lien dans l'email
Cliquer sur "se connecter" [login/mot de passe incorrect]/
Affichage d'un message d'erreur
Cliquer sur "Vous n'avez pas encore de compte ? Inscrivez-vous"
Page login
Cliquer sur "Se connecter" [login et mot de passe corrects]
Page explication procédure
Cliquer sur "Vous avez déjà un compte ? Connectez-vous"
Cliquer sur "Se connecter" [visiteur]
Cliquer sur "S'enregistrer" [données valides]
Cliquer sur "S'enregistrer" [données invalides]/
Affichage d'un message d'erreur
Page enregistrement
Cliquer sur "S'enregistrer" [visiteur]
Cliquer sur "Accueil"
Page accueil
Cliquer sur "Demandes organisateurs" [admin]
Page demandes organisateurs
Cliquer sur "Accepter" ou "Refuser"/
Mise à jour de la liste des demandes
Cliquer sur "Créer" [données invalides]/
Affichage d'un message d'erreur
Page création de concours
Cliquer sur "Créer un concours" [organisateur]
Cliquer sur "Annuler"
Cliquer sur "Concours" [visiteur || challenger || organisateur]
Cliquer sur "Comptes" [admin]
Cliquer sur "Quitter l'équipe" [challenger && son équipe && concours actuel]
Page équipe
Page liste concours à venir
Cliquer sur "Présent"
Cliquer sur "A venir"
Page concours actuel
Page comptes challengers
Cliquer sur "Organisateurs"
Cliquer sur "Challengers"
Page comptes organisateurs
Cliquer sur "Supprimer"/
Mise à jour de la liste des comptes
Cliquer sur un concours
Cliquer sur retour
Cliquer sur "Terminés"
Cliquer sur "Présent"
Page liste concours terminés
Page concours à venir
Cliquer sur "Supprimer"
Cliquer sur "Annuler"
Cliquer sur "Supprimer"/Mise à jour de la liste des comptes
Demande de confirmation
Cliquer sur "Annuler"
Cliquer sur "Quitter"/Mise à jour de la liste des membres
Cliquer sur "Quitter l'équipe" [challenger && son équipe && concours actuel]
Demande de confirmation
Cliquer sur "Annuler" [solution déposée]
Cliquer sur "S'enregistrer" [visiteur]
Page concours terminé
Page profil
Cliquer sur "Modifier les données"
Page profil éditable
Cliquer sur "Sauvegarder [données invalides]/
Affichage d'un message d'erreur
Cliquer sur "Sauvegarder [données valides]/
Affichage d'un message de confirmation
Cliquer sur "Se connecter" [visiteur]
Cliquer sur une équipe
Cliquer en dehors du pop-up
Cliquer sur "Créer" [données valides]/
Affichage d'un message de confirmation
Cliquer sur la photo de profil [challenger || organisateur]
Cliquer sur "Déposer une solution/Voir les solutions" [challenger]
Cliquer sur retour
Page solutions
Cliquer sur "Télécharger la solution"
Téléchargement de la solution
Cliquer sur "Inscrire le groupe" [données valides]/
Affichage d'un message de confirmation
Cliquer en dehors du pop-up
Page solution
Cliquer sur l'icône de fichier [concours actuel]
Explorateur de fichiers
Sélection d'un fichier/
Affichage du nom du fichier et
les boutons "Déposer" et "Annuler"
Cliquer sur une équipe [solution]
Cliquer sur "gérer une équipe/créer une équipe"
Cliquer sur "Soumettre la note"/
Mise à jour de la solution
Affichage d'un message de confirmation
Cliquer sur "Evaluer la solution"
Page gestion d'équipe
Cliquer sur "Inscrire le groupe" [données invalides]/
Affichage d'un message d'erreur
Cliquer sur "Déposer" [solution déposée]/
Mise à jour de la liste des solutions
Affichage d'un message de confirmation
Cliquer sur une solution/
Téléchargement de la solution
Cliquer en dehors du pop-up
Page note
Cliquer sur "Supprimer"/
Mise à jour de la liste des équipes
Cliquer sur "Annuler"
Cliquer sur "Supprimer l'équipe"
Demande de confirmation

2.4.3 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.



2.5 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		Inclus dans le HttpClient d'Angular	
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 v5.7
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		

2.6 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	L'API que nous avons mise en place est de type REST. C'est très rapidement implémenté et disponible dans 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 ne générons pas de document à partir du XML.
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		Nous n'utilisons pas le langage Xquery car des méthodes sont déjà fournies par le DOM Parser de javascript.
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.

2.7 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.		Nous n'avons pas séparé les tâches précisément, nous les avons définies par oral.

3 Conclusion

3.1 Objectifs atteints

Actuellement, le login a été implémenté correctement. La persistance du token dans le localstorage du navigateur a été réalisée afin d'éviter la perte de celui lors du rafraîchissement de la page. A cela s'ajoute l'affichage des informations pour les concours passés, actuel et futurs. La modification des informations de profil d'un compte challenger a été en grande partie implémentée. Le téléchargement des figures et vidéos est également disponible.

Objectifs	
UseCase1 – Définir les privilèges	70%
UseCase2 – Supprimer un compte	0%
UseCase3 – Consulter les informations d'un concours	90%
UseCase4 – S'enregistrer sur la plateforme	20%
UseCase5 - Login	100%
UseCase6 – Consulter/Modifier les données de son compte	80%
UseCase7 – Consulter les équipes	0%
UseCase8 – Gérer une équipe	0%

UseCase9 – Logout	100%
UseCase10 – Supprimer une équipe	0%
UseCase11 – Supprimer un membre	0%
UseCase12 – Ajouter un membre	0%
UseCase13 – Vérifier la session	100%
UseCase14 – Consulter les solutions déposées	0%
UseCase15 – Déposer une solution	0%
UseCase16 – Consulter les concours	70%
UseCase17 – Créer un concours	0%
UseCase18 – Gérer un concours	0%
UseCase19 – Supprimer un concours	0%
UseCase20 – Evaluer les solutions	0%
UseCase21 – Valider les inscriptions	0%

3.2 Problèmes rencontrés

3.2.1 MySQL et XML

Le SGBD qui nous a été imposé, c'est-à-dire MySQL, ne supporte pas le type XML. Ceci implique que nous avons du valider notre XML au niveau applicatif.

3.2.2 Versions de librairies spring

Durant l'implémentation de notre application, nous avons été à maintes reprises confrontés à la difficulté de trouver la bonne version des librairies de spring ainsi que la documentation qui s'y rattache.

3.2.3 Visibilité des diagrammes UML

Du au fait que nos diagrammes UML était assez précis et détaillés, la taille des diagrammes UML était souvent conséquente. C'est la raison pourquoi la lisibilité des diagrammes a été souvent problématique.

3.3 Problèmes non résolus

Tous les problèmes que nous avons rencontrés jusqu'à maintenant ont été pour la majorité résolus, sachant bien que l'implémentation n'est pas aussi avancée que prévu.

3.4 Perspectives futures

Le concept de l'application est intéressant et pourrait être utile dans un cadre pédagogique. Cependant, comme plusieurs groupes ont travaillé sur ce même projet, il serait intelligent d'utiliser le projet le plus aboutit pour y donner suite.

3.5 Synthèse

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 framewors avec lesquels nous allions mener à bien ce projet. Cela nous a aussi permis de pouvoir travailler avec le framework Spring qui est très populaire dans le monde professionnel mais qui 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 eu beaucoup trop de modélisation. Nous doutons qu'une entreprise budgétise de quoi faire autant de fiches descriptives, diagrammes de séquence et diagrammes de communication. Hormis les diagrammes de classes, le diagramme use case et le diagramme d'états-transitions, nous n'avons pas vu l'utilité des autres diagrammes demandés. 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 a 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.

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.

4 Webographie

4.1.1 Spring

[S1], Spring documentation : https://docs.spring.io/spring/docs

4.1.2 MySQL

[S1], Cours MySQL : https://openclassrooms.com/courses/administrez-vos-bases-de-donnees-avec-mysql

[S2], MySQL documentation : https://dev.mysql.com

[S3], Mise en forme MySQL : https://tohtml.com/mysql/

4.1.3 Angular

[S1], Angular documentation : https://angular.io

4.1.4 AJAX

[S1], Tutoriels et exemples : https://www.w3schools.com