Hashcode

**Groupe No :** 6

Rapport d’élaboration : variante avec composants

Joé Butty

Fuchs Nicolas

Rial Johnatan

**Filière :** Informatique

**Technologies :** JavaEE, MySQL.

**Date du rendu :** 18 avril 2018

**Superviseurs :** Prof. Houda Chabbi Drissi

Prof. Pierre Kuonen

Prof. Omar Abou Khaled

**Client:** Prof. Pierre Kuonen

**Table des matières**

1 Changements par rapport à la phase de création 5

1.1 Base de données 5

1.1.1 Modèle Relationnel 5

1.1.2 Contrainte d’intégrité 5

1.1.3 Contraintes relationnelles 6

1.2 Nouveau cas d’utilisation 6

1.2.1 Fiche descriptive de valider le compte 7

1.3 Modification des fiches descriptives 7

1.3.1 Cas 2 : Supprimer un compte 7

1.3.2 Cas 4 : S’enregistrer sur la plateforme 7

1.3.3 Cas 6 : Consulter / Modifier les données de son compte 7

1.3.4 Cas 15 : Déposer une solution 8

1.3.5 Cas 17 : Créer un concours 8

1.3.6 Cas 18 : Gérer un concours 9

2 Présentation des composants utilisés 10

1.4 Côté client 10

1.5 Côté serveur 10

1.6 Côté base de données 11

3 Diagrammes de séquence et de communication 12

3.1 Cas 1 : Définir les privilèges 12

3.1.1 Diagramme de séquence 12

3.1.2 Diagramme de communication 13

3.1.3 Transactions 13

3.2 Cas 2 : Supprimer un compte 14

3.2.1 Diagramme de séquence 14

3.2.2 Diagramme de communication 15

3.2.3 Transactions 15

3.3 Cas 3 : Consulter les informations d’un concours 17

3.3.1 Diagramme de séquence 17

3.3.2 Diagramme de communication 18

3.3.3 Transaction 18

3.4 Cas 4 : S’enregistrer sur la plateforme 20

3.4.1 Diagramme de séquence 20

3.4.2 Diagramme de communication 21

3.4.3 Transaction 21

3.5 Cas 5 : Login 22

3.5.1 Diagramme de séquence 22

3.5.2 Diagramme de communication 22

3.6 Cas 6 : Consulter / Modifier les données de son compte 23

3.6.1 Diagramme de séquence 23

3.6.2 Diagramme de communication 24

3.6.3 Transaction 24

3.7 Cas 7 : Consulter les équipes 25

3.7.1 Diagramme de séquence 25

3.7.2 Diagramme de communication 26

3.7.3 Transaction 26

3.8 Cas 8 : Gérer une équipe 27

3.8.1 Diagramme de séquence 27

3.8.2 Diagramme de communication 28

3.8.3 Transaction 28

3.9 Cas 9 : Logout 30

3.9.1 Diagramme de séquence 30

3.9.2 Diagramme de communication 30

3.10 Cas 10 : Supprimer les équipes 31

3.10.1 Diagramme de séquence 31

3.10.2 Diagramme de communication 32

3.10.3 Transaction 32

3.11 Cas 11 : : Supprimer un membre 33

3.11.1 Diagramme de séquence 33

3.11.2 Diagramme de communication 33

3.11.3 Transaction 33

3.12 Cas 12 : : Ajouter un membre 34

3.12.1 Diagramme de séquence 34

3.12.2 Diagramme de communication 35

3.12.3 Transaction 35

3.13 Cas 13 : Vérifier la session 36

3.13.1 Diagramme de séquence 36

3.13.2 Diagramme de communication 36

3.14 Cas 14 : Consulter les solutions déposées 37

3.14.1 Diagramme de séquence 37

3.14.2 Diagramme de communication 38

3.14.3 Transaction 38

3.15 Cas 15 : Déposer une solution 39

3.15.1 Diagramme de séquence 39

3.15.2 Diagramme de communication 40

3.15.3 Transaction 40

3.16 Cas 16 : Consulter les concours 41

3.16.1 Diagramme de séquence 41

3.16.2 Diagramme de communication 42

3.16.3 Transaction 42

3.17 Cas 17 : Créer un concours 43

3.17.1 Diagramme de séquence 43

3.17.2 Diagramme de communication 44

3.17.3 Transaction 44

3.18 Cas 18 : Gérer un concours 45

3.18.1 Diagramme de séquence 45

3.18.2 Diagramme de communication 46

3.18.3 Transaction 46

3.19 Cas 19 : Supprimer un concours 47

3.19.1 Diagramme de séquence 47

3.19.2 Diagramme de communication 48

3.19.3 Transaction 48

3.20 Cas 20 : Evaluer les solutions 49

3.20.1 Diagramme de séquence 49

3.20.2 Diagramme de communication 50

3.20.3 Transaction 50

3.21 Cas 21 : Valider les inscriptions 51

3.21.1 Diagramme de séquence 51

3.21.2 Diagramme de communication 52

3.21.3 Transaction 52

3.22 Cas 22 : Valider le compte 53

3.22.1 Diagramme de séquence 53

3.22.2 Diagramme de communication 54

3.22.3 Transaction 54

4 Liste des objets et composants 54

5 Diagramme de classes 54

6 Concurrence 55

6.1 Récapitulatif sur les transactions 55

6.2 Présentation d’un scénario concurrentiel 56

7 Diagramme de composants 56

8 Signatures 57

# Changements par rapport à la phase de création

Nous avons réalisé quelques changements par rapport à l’ancien rapport, ce chapitre décrit en détail chaque changement apporté.

## Base de données

Nous avons apporté plusieurs modifications par rapport au rendu précédents, nous avons corrigé le modèle relationnel et listé toutes les contraintes.

### Modèle Relationnel

account(id\_account, firstname, lastname, email, pseudo, password, token, image, #id\_role)

role(id\_role, name)

team(id\_team, t\_name, #challenge, #leader)

challenge(id\_challenge, c\_name, nb\_teams, date\_inscription, begin, end, mediaXML)

solution(id\_solution, s\_name, language, solution, version, ranking, submit\_date, #(account-team) )

data(id\_data, file, #challenge)

account-team(#id\_account, #id\_team)

challenge-organizer(#id\_challenge, #id\_organizer)

### 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.

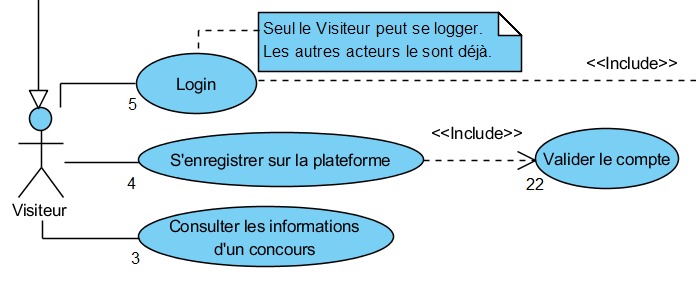
### Contraintes relationnelles

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 »

## Nouveau cas d’utilisation

Nous avons ajouté un nouveau cas d’utilisation dans notre diagramme Use Case. Il s’agit de « Valider le compte » qui inclus le cas « S’enregistrer » lié au visiteur. La fiche descriptive est disponible à la page suivante.



### Fiche descriptive de valider le compte

**Résumé** **:** Une fois inscrit, le visiteur peut récupérer un lien dans son adresse Email qui permettra de valider son compte

**Acteurs** **:** Visiteur

#### Description des enchaînements :

**Pré conditions :** -

**Scénario nominal :**

1. L'acteur arrive sur le lien permettant de valider son compte
2. Le système met à jour le compte dans la BD
3. Le système affiche une page de bienvenue
4. Fin du cas

**Enchainements alternatifs :**

A1 : L'acteur arrive une deuxième fois sur le lien de validation

Démarre au point 2 du scénario nominal

1. Le système affiche une page expliquant que le compte n’existe pas ou a déjà été validé.
2. Fin du cas

**Enchainements d’exception :** -

**Post conditions :** -

## Modification des fiches descriptives

Durant la modélisation des diagrammes de séquence, nous avons apporté quelques modifications sur les fiches descriptives. Elles sont toutes répertoriées dans ce chapitre.

### Cas 2 : Supprimer un compte

La terminologie de “Challenger” est fausse, il faut la remplacer par “Persone loggées”.

### Cas 4 : S’enregistrer sur la plateforme

Les lignes de11 à 13 (comprises) sont supprimées.

### Cas 6 : Consulter / Modifier les données de son compte

**Résumé :** Une personne loguée sur la plateforme peut paramétrer son profil

**Acteurs** **:** Personne loguée, Organisateur

**Scénario nominal :**

1. L'acteur clique sur la photo de profil
2. <<UC : Vérifier la session>>
3. Le système récupère les informations de l'acteur dans la BD
4. Le système affiche une page avec les informations de l'acteur
5. Fin du cas

**Enchaînements alternatifs :**A1 : L’acteur souhaite modifier ses données

Démarre à la place du point 5 du scénario nominal

1. L'acteur clique sur le bouton « Modifier les données »
2. Le système affiche les champs en mode éditables
3. L’acteur modifie ses champs
4. L'acteur clique sur le bouton "Sauvegarder"
5. <<UC : Vérifier la session>>
6. Le système vérifie la validité et le format des informations entrées
7. Le système met à jour le compte dans la BD
8. Le système affiche un message de confirmation de modification
9. Fin du scénario alternatif

A2 : Les formats des données ne sont pas valides

Démarre au point 7 du scénario alternatif A1

1. Le système affiche un message d’erreur
2. Fin du scénario alternatif

### Cas 15 : Déposer une solution

Dans le scénario nominal, il faut rajouter une ligne entres le n° 16 et le n° 17. Cette ligne est : “Le système affiche une confirmation de déposition de la solution”

Le résultat donne :

16. Le système crée une solution dans la BD

17. Le système affiche une confirmation de déposition de la solution

18. Fin du cas

### Cas 17 : Créer un concours

Dans le scénario nominal, la ligne 6 doit être placée en dessous de la ligne 8.

Il manque un scénario alternatif :

A4 : Le système ne trouve pas l’organisateur recherché par l’acteur

Démarre au point 4 du scénario alternatif A2

1. L’organisateur efface le nom entré dans la recherche
2. Fin du scénario alternatif

### Cas 18 : Gérer un concours

La ligne 10 est remplacé par « UC : Vérifier la session » et une nouvelle ligne entre la 11 et 12 est rajouté « UC : Vérifier la session »

Le résultat donne :

9. L'acteur clique pour fermer le pop-up avec les informations d’un concours

10. « UC : Vérifier la session »

11. L’acteur clique pour fermer le pop-up avec les concours liés par l’organisateur

12. « UC : Vérifier la session »

13. Fin du cas

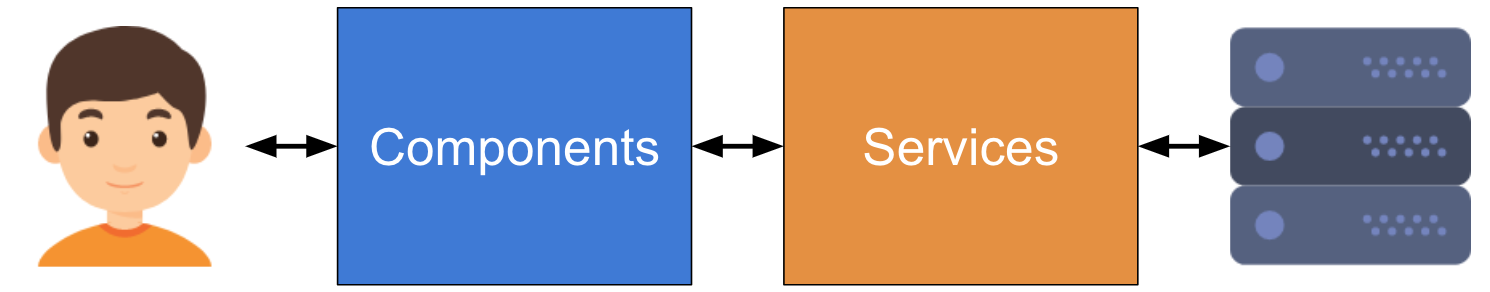
# Présentation des composants utilisés

Plusieurs composants seront utilisés pour réaliser la plateforme. Cette plateforme sera séparée en trois : la partie cliente, la partie serveur et la base de données. Pour la partie client, Angular sera utilisé. Côté serveur Spring Framework permettra de facilement créer une API et côté base de données, MySQL a été imposé pour ce projet. Voici un schéma qui résume l’interaction entre ces différentes parties :



## Côté client

Pour le client, le choix a été fait d’utiliser Angular comme framework. Celui-ci permet de créer des applications web riches grâce au langage TypeScript. Ce langage est développé par Microsoft. Il s’agit d’une surcouche de Javascript qui introduit la notion de type. Il permet donc de programmer d’une manière plus proche de ce qu’il se fait en Java ou C#. Angular quant à lui permet de réaliser facilement du binding entre le modèle et la vue. Il impose également une séparation de l’application en composants indépendants qui utilisent des services qui peuvent être partagés. Ces services servent notamment à communiquer avec une API. Voici un schéma qui résume l’interaction entre l’utilisateur, les composants, les service et l’API REST de l’application :



## Côté serveur

Pour le côté serveur, le choix s’est porté sur Spring Boot. Celui-ci permet de rapidement créer une application base sur le framework Spring qui s’appuie sur le langage Java. Le framework Spring mets à dispositions un nombre assez grand de librairies qui simplifie le développement d’une application. Pour ce projet, Spring Web sera utilisé afin de réaliser l’API, Spring Data JPA permettra de communiquer avec la base de données en s’appuyant sur le très répandu ORM Hibernate, Spring Session permettra gérer facilement les sessions des utilisateurs alors que Spring Security gérera l’authentification. Spring impose une séparation de l’application en trois parties : les controllers, les services et les repositories. Les controllers permettent de réaliser le mapping entre les requêtes http et les méthodes Java. Les services contiennent l’intelligence du programme comme par exemple réaliser la validation d’un XML ou l’envoie d’un email. Les repositories permettent l’interfaçage entre le code et la base de données.

Voici un schéma qui résume les interactions entre ceux-ci :



Concernant la persistance des données, Spring Data JPA et Hibernate sont assez flexibles et, de ce fait, n’influencent pas la modélisation de la base de données. Cependant certaines librairies de Spring nécessitent parfois d’intégrer des tables supplémentaires dans la base de données. C’est le cas notamment pour Spring Session qui a besoin d’une table « Session » pour persister l’état des sessions. Pour l’accès à la base, JDBC sera utilisé grâce au pilote officiel fournit par MySQL. La gestion des transactions peut être réalisée grâce à l’annotation @Transaction dans un service.

## Côté base de données

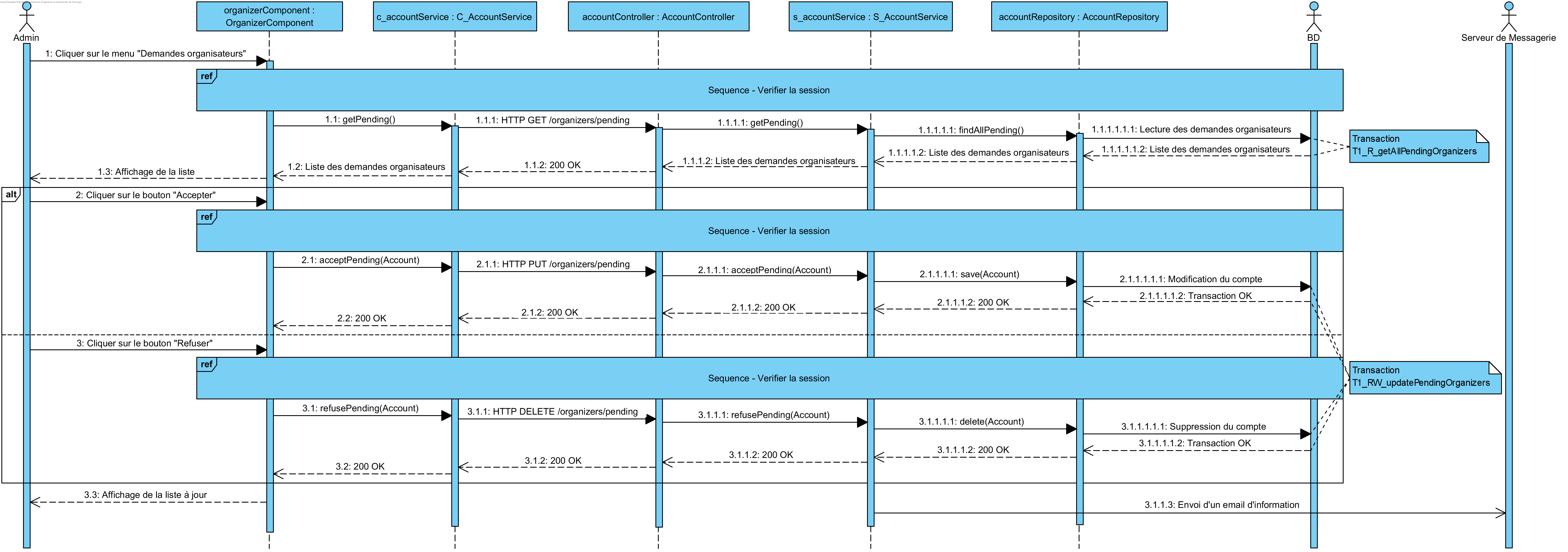
La technologie de la base de données a été imposée. Pour ce projet, il est obligatoire d’utiliser MySQL. La version communautaire publiée sous licence publique GNU sera utilisée. Ce SGBD est très répandu et profite d’une grosse communauté ce qui facilite la résolution des problèmes qui pourraient être rencontrés. Cependant, MySQL ne permet la validation d’un XML grâce à un XML Schema contrairement à SQLServer. Pour pallier à ce problème, cette validation est reportée au niveau applicatif.

# Diagrammes de séquence et de communication

## Cas 1 : Définir les privilèges

Un administrateur peut accepter la demande d'une personne souhaitant devenir organisateur.

### Diagramme de séquence



### Diagramme de communication

### Transactions

**T1\_R\_getAllPendingOrganizers**

La transaction va chercher la liste des personnes inscrites souhaitant devenir des organisateurs.

**Type de la transaction :** Lecture

**Isolation :** READ COMMITED

La transaction va simplement chercher la liste des organisateurs dans la base de données. Il n’est pas nécessaire d’avoir un niveau d’isolation plus élévée.

**Table(s) touchée(s) :** Account, Role

**Fréquence :** Rare

**Période(s) :** -

**T1\_RW\_updatePendingOrganizers**

La transaction va changer le rôle d’un compte souhaitant devenir organisateur en cas de validation, sinon le compte sera supprimé.

**Type de la transaction :** Lecture et Ecriture

**Isolation :** SERIALIZABLE

Dans le cas où deux admins (A1, A2) sont connectées, les deux ont chargé la liste des personnes qui souhaitent devenir un organisateur, si A1 a validé (ajouté) ou invalidé (supprimé) un organisateur après que A2 ait récupéré la liste, dans la vue de A2, il y aura une personne n’existant plus dans la base de données.

**Table(s) touchée(s) :** Account, Role

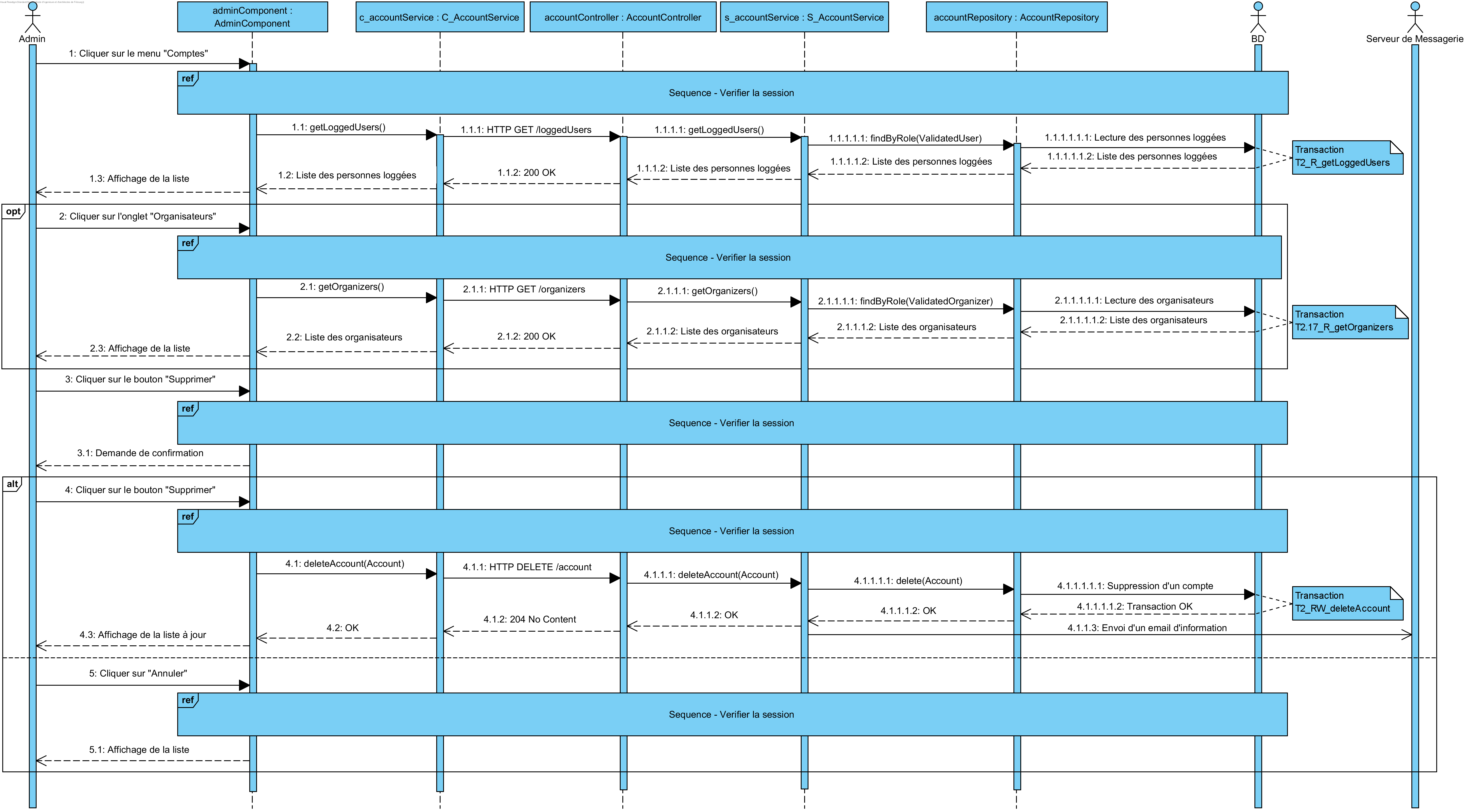
**Fréquence :** Rare

**Période(s) :** -

## Cas 2 : Supprimer un compte

Résumé du cas.

### Diagramme de séquence



### Diagramme de communication

### Transactions

**T2\_R\_getLoggedUsers**

La transaction va chercher la liste des personnes loggées.

**Type de la transaction :** Lecture

**Isolation :** READ COMMITED

La transaction va simplement chercher la liste des personnes loggées. Il n’est pas nécessaire d’avoir un niveau d’isolation plus élevé.

**Tables touchées :** Account, Role

**Fréquence :** Rare

**Période :** -

**T2\_R\_getOrganizers**

La transaction va chercher la liste des organisateurs.

**Type de la transaction :** Lecture

**Isolation :** READ COMMITED

La transaction va simplement chercher la liste des organisateurs. Il n’est pas nécessaire d’avoir un niveau d’isolation plus élevé.

**Tables touchées :** Account, Role

**Fréquence :** Rare

**Période :** -

**T2\_RW\_deleteAccount**

La transaction va chercher la liste des organisateurs.

**Type de la transaction :** Lecture et Ecriture

**Isolation :** SERIALIZABLE

Dans le cas où deux admins (A1, A2) sont connectées, les deux ont chargé la liste des organisateurs, si A1 a supprimé un organisateur après que A2 ait récupéré la liste, dans la vue de A2, il y aura une personne n’existant plus dans la base de données.

**Tables touchées :** Account, Role

**Fréquence :** Rare

**Période :** -

## Cas 3 : Consulter les informations d’un concours

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

**T3.7.8.14.15.16.18\_R\_getChallenge**

La transaction va chercher toutes les informations d’un challenge spécifique.

**Type de la transaction :** Lecture

**Isolation :** READ COMMITED

La transaction va chercher toutes les informations nécessaire d’un challenge précis. Il n’est pas nécessaire d’avoir un niveau d’isolation supplémentaire

**Tables touchées :** Toutes les tables sauf solution

**Fréquence :** Elevé

**Période :** -

**T3.7.8.14.16\_R\_getPastChallenges**

La transaction va chercher la liste de tous les challenges passés.

**Type de la transaction :** Lecture

**Isolation :** READ COMMITED

La transaction va simplement chercher la liste des challenges déjà passé. Il n’est pas nécessaire d’avoir un niveau d’isolation supplémentaire.

**Tables touchées :** Challenge

**Fréquence :** Elevé

**Période :** -

**T3.7.8.16\_R\_getFuturChallenges**

La transaction va chercher la liste de tous les challenges futurs.

**Type de la transaction :** Lecture

**Isolation :** READ COMMITED

La transaction va simplement chercher la liste des prochains challenges. Il n’est pas nécessaire d’avoir un niveau d’isolation supplémentaire.

**Tables touchées :** Challenge

**Fréquence :** Elevé

**Période :** -

## Cas 4 : S’enregistrer sur la plateforme

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

**T4\_RW\_createAccount**

La transaction va ajouter un account dans la base de données, elle vérifie que l’utilisateur n’existe pas déjà.

**Type de la transaction :** Lecture et Ecriture

**Isolation :** SERIALIZABLE

Il est nécessaire de verrouiller la table car dans le cas où deux visiteurs (V1, V2) souhaitent s’inscrire en même temps avec le même pseudo, il ne faut pas que V1 et V2 vérifie l’existence du pseudo en même temps et écrive en même temps dans la base de donnée.

**Tables touchées :** Account, Role

**Fréquence :** Moyenne

**Période :** -

## Cas 5 : Login

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

## Cas 6 : Consulter / Modifier les données de son compte

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

**T6\_R\_getInfoAccount**

La transaction va chercher les informations d’un account.

**Type de la transaction :** Lecture

**Isolation :** READ COMMITTED

La transaction va simplement chercher les informations d’un compte précis. Il n’est pas nécessaire d’avoir un niveau d’isolation supplémentaire.

**Tables touchées :** Account

**Fréquence :** Moyenne

**Période :** -

**T6\_RW\_updateAccount**

La transaction va modifier un compte dans la base de données.

**Type de la transaction :** Lecture et Ecriture

**Isolation :** SERIALIZABLE

Il est nécessaire de verrouiller la table car dans le cas où deux visiteurs (V1, V2) souhaitent modifier leur pseudo en même temps et mettre la même valeur, il ne faut pas que V1 et V2 vérifie l’existence du pseudo en même temps et écrive en même temps dans la base de donnée. Il ne faut pas qu’un visiteur créé un compte en même temps.

**Tables touchées :** Account

**Fréquence :** Faible

**Période :** -

## Cas 7 : Consulter les équipes

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

**T7\_R\_getInfoTeam**

La transaction va chercher toutes les informations d’une équipe ainsi que leur membre

**Type de la transaction :** Lecture

**Isolation :** READ COMMITED

La transaction va simplement chercher les informations d’une équipe précise. Il n’est pas nécessaire d’avoir un niveau d’isolation supplémentaire.

**Tables touchées :** Team, Account,, Account-Team

**Fréquence :** Elevé

**Période :** -

**T7\_RW\_deleteMemberTeam**

La transaction va supprimer un membre d’une équipe.

**Type de la transaction :** Lecture et Ecriture

**Isolation :** REPEATABLE READ

Il faut éviter que des modifications aient lieu sur les lignes qui seront sélectionnée et modifier.

**Tables touchées :** Team, Account,, Account-Team

**Fréquence :** Faible

**Période :** -

## Cas 8 : Gérer une équipe

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

**T3.7.8. 14.15.16.18\_R\_getChallenge :** Cette transaction est décrite dans le chapitre « 3.3.3 transactions » du cas d’utilisation « Consulter les informations d’un concours »

**T3.7.8. 14.16\_R\_getFutureChallenges**: Cette transaction est décrite dans le chapitre « 3.3.3 transactions » du cas d’utilisation « Consulter les informations d’un concours »

**T8\_R\_getTeamsOfChallenge**

La transaction va récupérer toutes les équipes d’un challenges précis.

**Type de la transaction :** Lecture

**Isolation :** READ COMMITED

La transaction va simplement chercher les équipes d’un challenge précis. Il n’est pas nécessaire d’avoir un niveau d’isolation supplémentaire.

**Tables touchées :** Team, Account,, Account-Team

**Fréquence :** Elevé

**Période :** -

**T8\_W\_createTeam**

La transaction va créer une équipe dans la base de données et y ajouter ses membres.

**Type de la transaction :** Lecture et Ecriture

**Isolation :** SERIALISABLE

Il ne faut pas que la même équipe soit créer en même temps, que les utilisateurs modifient leur pseudo ou que le concours soit modifié durant la transaction. Il est donc nécessaire d’avoir une isolation niveau 3.

**Tables touchées :** Account,, Team, Challenge, Account-Team

**Fréquence :** Elevé

**Période :** -

**T8\_RW\_updateTeam**

La transaction va modifier une équipe dans la base de données, il est possible que la transaction ajoute ou supprime des membres de cette équipe.

**Type de la transaction :** Lecture et Ecriture

**Isolation :** SERIALISABLE

Il ne faut pas qu’une même équipe soit créer en même temps, que les utilisateurs modifient leur pseudo ou que le concours soit modifié durant la transaction. Il est donc nécessaire d’avoir une isolation niveau 3.

**Tables touchées :** Account,, Team, Challenge, Account-Team

**Fréquence :** Elevé

## Cas 9 : Logout

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

## Cas 10 : Supprimer les équipes

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

**T10\_W\_deleteTeam**

La transaction va supprimer une équipe d’un concours

**Type de la transaction :** Lecture et Ecriture

**Isolation :** SERIALISABLE

Il ne faut pas qu’une même équipe soit créer en même temps, que les utilisateurs modifient leur pseudo ou que le concours soit modifié durant la transaction. Il ne faut pas qu’une équipe ayant le même nom dans un autre concours soit créé en même temps Il est donc nécessaire d’avoir une isolation niveau 3.

**Tables touchées :** Account,, Team, Challenge, Account-Team

**Fréquence :** Elevé

## Cas 11 : : Supprimer un membre

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 12 : : Ajouter un membre

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

**T12\_R\_getChallengers**

La transaction va chercher la liste des personnes pouvant participer au concours

**Type de la transaction :** Lecture

**Isolation :** READ COMMITTED

La transaction va simplement chercher la liste des personne pouvant s’inscrire au challenge. Il n’est pas nécessaire d’avoir un niveau d’isolation supplémentaire.

**Tables touchées :** Account, Role, Team, Challenge, Account-Team

**Fréquence :** Moyenne

## Cas 13 : Vérifier la session

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

## Cas 14 : Consulter les solutions déposées

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

**T3.7.8. 14.15.16.18\_R\_getChallenge :** Cette transaction est décrite dans le chapitre « 3.3.3 transactions » du cas d’utilisation « Consulter les informations d’un concours »

**T3.7.8.14\_R\_getPastChallenges :** Cette transaction est décrite dans le chapitre « 3.3.3 transactions » du cas d’utilisation « Consulter les informations d’un concours »

**T14.15.20\_R\_getSolutionsOfTeam**

La transaction va chercher toutes les informations liées à toutes les solutions d’une équipe.

**Type de la transaction :** Lecture

**Isolation :** READ COMMITTED

La transaction va uniquement chercher des informations en lecture. Il n’est pas nécessaire d’avoir un niveau d’isolation supplémentaire.

**Tables touchées :** Account, , Team, Challenge, Solution, Account-Team

**Fréquence :** Elevé

**T14\_R\_getSolutionOfTeam**

La transaction va chercher toutes les informations liées à une solution d’une équipe.

**Type de la transaction :** Lecture

**Isolation :** READ COMMITTED

La transaction va uniquement chercher des informations en lecture. Il n’est pas nécessaire d’avoir un niveau d’isolation supplémentaire.

**Tables touchées :** Account, , Team, Challenge, Solution Account-Team

**Fréquence :** Elevé

## Cas 15 : Déposer une solution

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

**T3.7.8. 14.15.16.18\_R\_getChallenge :** Cette transaction est décrite dans le chapitre « 3.3.3 transactions » du cas d’utilisation « Consulter les informations d’un concours »

**T14.15\_R\_getSolutionsOfTeam :** Cette transaction est décrite dans le chapitre « 3.14.3 transactions » du cas d’utilisation « Consulter les solutions déposées»

T15\_W\_createSolution

## Cas 16 : Consulter les concours

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

T3.7.8.14.15.16\_R\_getChallenge

T3.7.8.14.16\_R\_getPastChallenges

T3.7.8.16\_R\_getFutureChallenges

## Cas 17 : Créer un concours

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

T17\_R\_getChallengesDates

T2.17\_R\_getOrganizers

T17\_W\_createChallenge

## Cas 18 : Gérer un concours

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

T18\_R\_getChallengesOfOrganizer

T3.7.8.14.15.16.18\_R\_getChallenge

## Cas 19 : Supprimer un concours

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

T19\_W\_deleteChallenge

## Cas 20 : Evaluer les solutions

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

T7.20\_R\_getInfoTeam

T14.20\_R\_getSolutionsOfTeam

T20\_RW\_updateSolution

## Cas 21 : Valider les inscriptions

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

T21\_RW\_updateTeam

T21\_W\_deleteTeam

## Cas 22 : Valider le compte

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

Transaction

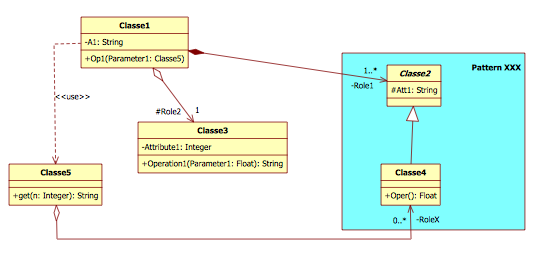
T22\_RW\_updatePendingTeam

# Liste des objets et composants

Dans ce chapitre indiquez la liste de tous les objets et composants (au format de la table ci-dessous) que vous utilisez dans vos diagrammes de séquence et de communication. Pour chaque objet indiquez dans quel(s) cas d'utilisation(s) il est utilisé et de quelle classe il est une instance.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Nom de l’objet ou du composant | Nome de la classe | No. Des Use Case |
| 1 | ObjetA | ClasseB | 1,3,6 |
| 2 | ComposantX |  | 2 |
| … | … | … | ... |

# Diagramme de classes



Commentez le diagramme et en particulier l’utilisation des patterns (si vous en avez utilisé). C'est un diagramme de classe de conception, vous devez y indiquer la multiplicité (par défaut =1) et les noms des rôles. Il ne devrait y avoir plus que des agrégations (simple ou composition), des généralisations (ou implémentation), éventuellement des classe associations des<<use>> ou des <<create>>. Si vous pensez que vous avez besoin d'autres relations, discutez-en avec votre professeur. Faites usage de la notions de «stéréotype» si vous pensez que certaines classes ne seront pas vraiment implémentées comme des classes (au sens habituel du terme). Si vous utilisez des composants proposez des solutions à votre professeur de GL pour les faire apparaître dans votre diagramme de classe.

**ATTENTION :** Les diagrammes de classe doivent être lisibles. Si nécessaire utilisé un format A3 ou éclatez votre diagramme en plusieurs parties.

# Concurrence

## Récapitulatif sur les transactions

Avoir une vue synthétique de toutes les transactions votre application. Les transactions pouvant attaquer la base de données pendant des périodes différentes de temps période, ceci apparaitra dans le tableau. Pour chaque transaction, définir quelles tables sont utilisées ainsi que le mode d’accès, lecture(R) ou écriture (W).

Si vous avez des périodes temporelles dans votre application, qui font qu’elle a des comportements différenciés suivant ces périodes, alors faire un tableau par période.

* Période1 - **avant inscription**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nom  Transactions | Table1 | … | … | … | Table\_n | Niveau Isolation |
| … | R | R | R |  |  | 1 |
| … | R | W |  |  |  | 0 |
| … | R | R | R/W | R | R | 3 |
| … |  | R |  |  |  |  |

* Période2 - **pendant inscription**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nom  Transactions | Table1 | … | … | … | Table\_n | Niveau Isolation |
| … | R | R | R |  |  | 1 |
| … | R | W |  |  |  | 0 |
| … | R | R | R/W | R | R | 3 |
| … |  | R |  |  |  |  |

## Présentation d’un scénario concurrentiel

Indiquez un scénario clair avec un problème de concurrence dans votre application. Indiquez clairement :

1. Les noms des transactions en jeu
2. Les problèmes qui risquent d’arriver
3. Expliquez alors clairement comment s’applique la philosophie de la gestion de concurrence que vous avez choisie sur ce cas et avec laquelle vous évitez les problèmes.

# Diagramme de composants

Il s’agit de représenter les composants et les dépendances explicites entre interfaces. ATTENTION : Les interfaces doivent être détaillées et, pour les composants que vous avez développés, il faut indiquer la liste des classes que contient le composant.

# Signatures

Fribourg, le ……………..............

Etudiant1 Etudiant2 Etudiant3