

Hashcode

Groupe No : 6

Rapport d'élaboration : variante avec composants

Butty Joé

Fuchs Nicolas

Rial Johnatan

Filière : Informatique

Technologies : JavaEE, MySQL.

Date du rendu : 20 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 <i>Modèle Relationnel</i>	5
1.1.2 <i>Contrainte d'intégrité.....</i>	5
1.1.3 <i>Contraintes relationnelles.....</i>	6
1.2 Nouveau cas d'utilisation.....	6
1.2.1 <i>Fiche descriptive de valider le compte.....</i>	7
1.3 Nouveaux acteurs dans le diagramme Use Case.....	7
1.4 Modification des fiches descriptives.....	8
1.4.1 <i>Cas 2 : Supprimer un compte</i>	8
1.4.2 <i>Cas 4 : S'enregistrer sur la plateforme.....</i>	8
1.4.3 <i>Cas 6 : Consulter / Modifier les données de son compte</i>	8
1.4.4 <i>Cas 15 : Déposer une solution</i>	9
1.4.5 <i>Cas 17 : Créer un concours</i>	9
1.4.6 <i>Cas 18 : Gérer un concours</i>	9
2 Présentation des composants utilisés	10
1.5 Côté client.....	10
1.6 Côté serveur	10
1.7 Côté base de données	11
3 Diagrammes de séquence et de communication.....	12
3.1 Cas 1 : Définir les priviléges	13
3.1.1 <i>Diagramme de séquence</i>	13
3.1.2 <i>Diagramme de communication</i>	14
3.1.3 <i>Transactions</i>	14
3.2 Cas 2 : Supprimer un compte.....	16
3.2.1 <i>Diagramme de séquence</i>	16
3.2.2 <i>Diagramme de communication</i>	17
3.2.3 <i>Transactions</i>	17
3.3 Cas 3 : Consulter les informations d'un concours.....	19
3.3.1 <i>Diagramme de séquence</i>	19
3.3.2 <i>Diagramme de communication</i>	20
3.3.3 <i>Transaction</i>	20
3.4 Cas 4 : S'enregistrer sur la plateforme.....	22
3.4.1 <i>Diagramme de séquence</i>	22
3.4.2 <i>Diagramme de communication</i>	23
3.4.3 <i>Transaction</i>	23
3.5 Cas 5 : Login.....	24
3.5.1 <i>Diagramme de séquence</i>	24
3.5.2 <i>Diagramme de communication</i>	24
3.6 Cas 6 : Consulter / Modifier les données de son compte	25

3.6.1	<i>Diagramme de séquence</i>	25
3.6.2	<i>Diagramme de communication</i>	26
3.6.3	<i>Transaction</i>	26
3.7	Cas 7 : Consulter les équipes.....	28
3.7.1	<i>Diagramme de séquence</i>	28
3.7.2	<i>Diagramme de communication</i>	29
3.7.3	<i>Transaction</i>	29
3.8	Cas 8 : Gérer une équipe.....	31
3.8.1	<i>Diagramme de séquence</i>	31
3.8.2	<i>Diagramme de communication</i>	32
3.8.3	<i>Transaction</i>	32
3.9	Cas 9 : Logout.....	34
3.9.1	<i>Diagramme de séquence</i>	34
3.9.2	<i>Diagramme de communication</i>	34
3.10	Cas 10 : Supprimer les équipes.....	35
3.10.1	<i>Diagramme de séquence</i>	35
3.10.2	<i>Diagramme de communication</i>	36
3.10.3	<i>Transaction</i>	36
3.11	Cas 11 : : Supprimer un membre	37
3.11.1	<i>Diagramme de séquence</i>	37
3.11.2	<i>Diagramme de communication</i>	37
3.12	Cas 12 : : Ajouter un membre	38
3.12.1	<i>Diagramme de séquence</i>	38
3.12.2	<i>Diagramme de communication</i>	39
3.12.3	<i>Transaction</i>	39
3.13	Cas 13 : Vérifier la session	40
3.13.1	<i>Diagramme de séquence</i>	40
3.13.2	<i>Diagramme de communication</i>	41
3.14	Cas 14 : Consulter les solutions déposées.....	42
3.14.1	<i>Diagramme de séquence</i>	42
3.14.2	<i>Diagramme de communication</i>	43
3.14.3	<i>Transaction</i>	43
3.15	Cas 15 : Déposer une solution.....	45
3.15.1	<i>Diagramme de séquence</i>	45
3.15.2	<i>Diagramme de communication</i>	46
3.15.3	<i>Transaction</i>	46
3.16	Cas 16 : Consulter les concours	47
3.16.1	<i>Diagramme de séquence</i>	47
3.16.2	<i>Diagramme de communication</i>	48
3.16.3	<i>Transactions</i>	48
3.17	Cas 17 : Créer un concours	49
3.17.1	<i>Diagramme de séquence</i>	49
3.17.2	<i>Diagramme de communication</i>	50

3.17.3	<i>Transaction</i>	50
3.18	Cas 18 : Gérer un concours	52
3.18.1	<i>Diagramme de séquence</i>	52
3.18.2	<i>Diagramme de communication</i>	53
3.18.3	<i>Transaction</i>	53
3.19	Cas 19 : Supprimer un concours	54
3.19.1	<i>Diagramme de séquence</i>	54
3.19.2	<i>Diagramme de communication</i>	55
3.19.3	<i>Transaction</i>	55
3.20	Cas 20 : Evaluer les solutions	56
3.20.1	<i>Diagramme de séquence</i>	56
3.20.2	<i>Diagramme de communication</i>	57
3.20.3	<i>Transaction</i>	57
3.21	Cas 21 : Valider les inscriptions	58
3.21.1	<i>Diagramme de séquence</i>	58
3.21.2	<i>Diagramme de communication</i>	59
3.21.3	<i>Transaction</i>	59
3.22	Cas 22 : Valider le compte	61
3.22.1	<i>Diagramme de séquence</i>	61
3.22.2	<i>Diagramme de communication</i>	62
3.22.3	<i>Transaction</i>	62
4	Liste des objets et composants.....	63
5	Diagramme de classes	65
6	Concurrence	67
6.1	Récapitulatif sur les transactions.....	67
6.2	Présentation d'un scénario concurrentiel	69
7	Diagramme de composants.....	69
8	Signatures.....	71

1 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.

1.1 Base de données

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

1.1.1 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)
```

1.1.2 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.

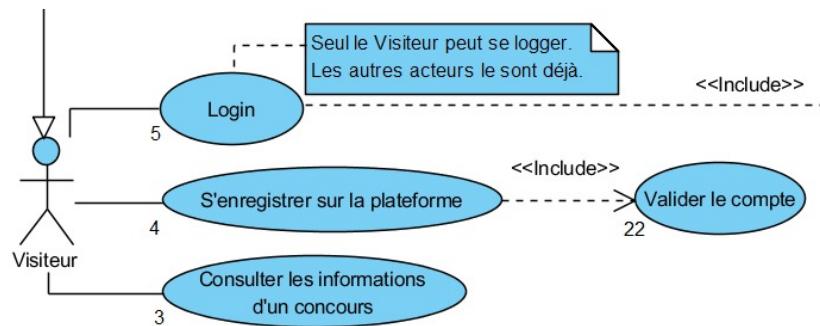
1.1.3 Contraintes relationnelles

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 »

1.2 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.



1.2.1 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

1.2.1.1 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 : -

1.3 Nouveaux acteurs dans le diagramme Use Case

Dans le diagramme UseCase, nous avons rajouté un nouvel acteur qui est le serveur de messagerie. Elle permet d'envoyer un Email.

L'acteur " Serveur de messagerie" est lié aux cas :

- Cas 1 : Definir les privileges
- Cas2 : Supprimer un compte
- Cas 10 : Supprimer une équipe
- Cas 19 : Supprimer un concours
- Cas 21 : Valider les inscriptions

L'acteur " Serveur de fichiers" est lié aux cas :

- Cas 14 : Consulter les solutions déposées
- Cas 15 : Déposer une solution
- Cas 17 : Créer un concours
- Cas 20 : Évaluer les solutions

1.4 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.

1.4.1 Cas 2 : Supprimer un compte

La terminologie de "Challenger" est fausse, il faut la remplacer par "Personne loggée".

1.4.2 Cas 4 : S'enregistrer sur la plateforme

Les lignes de 11 à 13 (comprises) sont supprimées.

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

Résumé : Une personne loguée sur la plateforme peut paramétriser 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

1.4.4 Cas 15 : Déposer une solution

Dans le scénario nominal, il faut rajouter une ligne entre 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

1.4.5 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

1.4.6 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

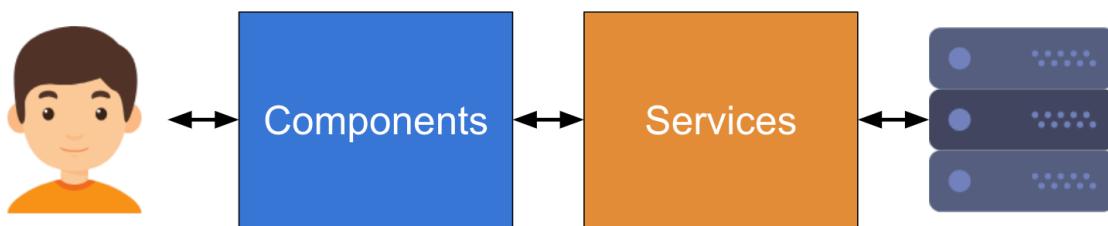
2 Présentation des composants utilisés

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 :



1.5 Côté client

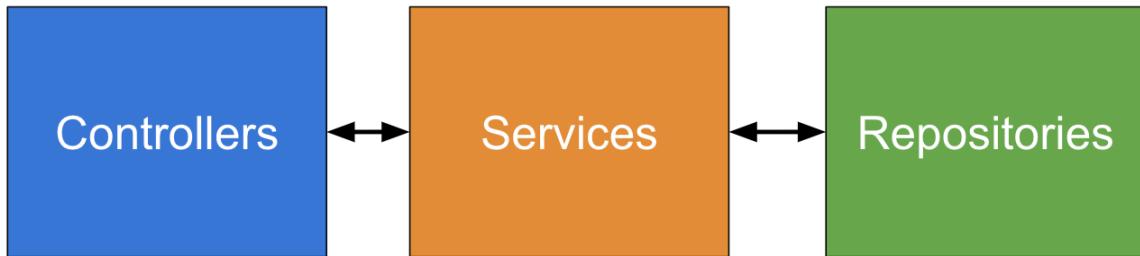
Pour le client, le choix a été fait d'utiliser Angular comme framework. Celui-ci permet de créer des applications web riches grâce au langage TypeScript. Ce langage est développé par Microsoft. Il s'agit d'une surcouche de Javascript qui introduit la notion de type. Il permet donc de programmer d'une 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 services et l'API REST de l'application :



1.6 Côté serveur

Pour le côté serveur, le choix s'est porté sur Spring Boot. Celui-ci permet de rapidement créer une application basée sur le framework Spring qui s'appuie sur le langage Java. Le framework Spring met à disposition 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 de 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'envoi 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 :



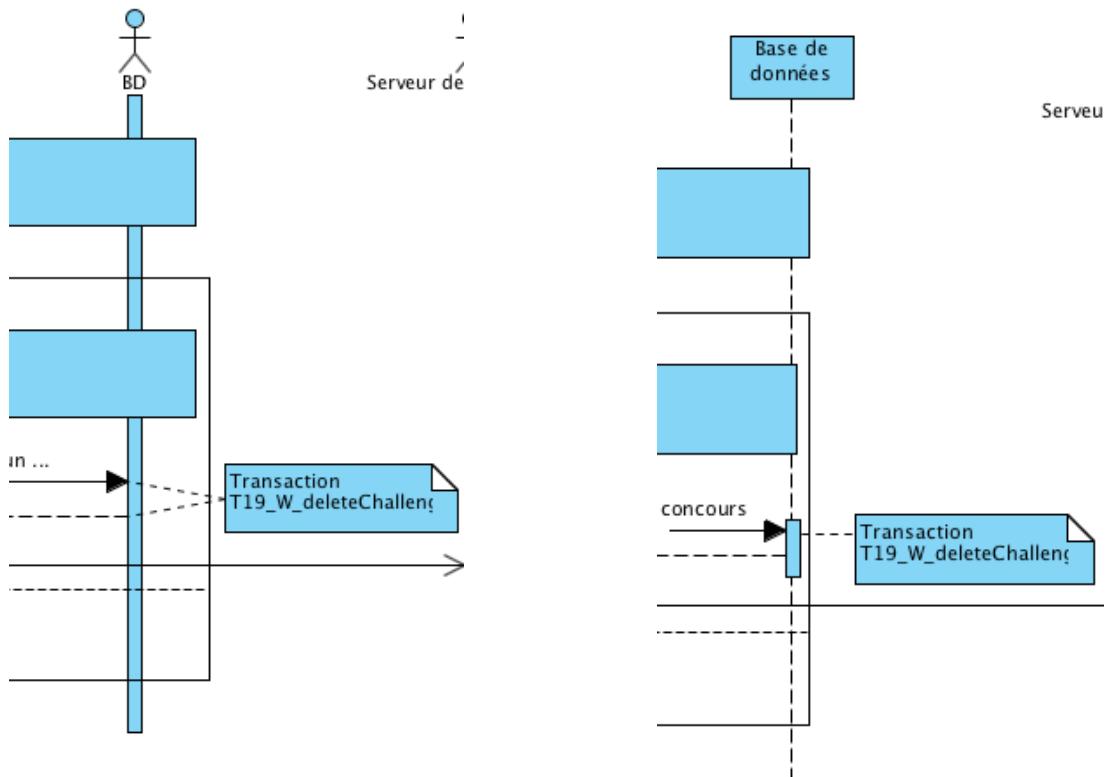
Concernant la persistance des données, Spring Data JPA et Hibernate sont assez flexibles et, de ce fait, n'influencent pas la modélisation de la base de données. Cependant certaines librairies de Spring nécessitent parfois d'intégrer des tables supplémentaires dans la base de données. C'est le cas notamment 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 fourni par MySQL. La gestion des transactions peut être réalisée grâce à l'annotation `@Transaction` dans un service.

1.7 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.

3 Diagrammes de séquence et de communication

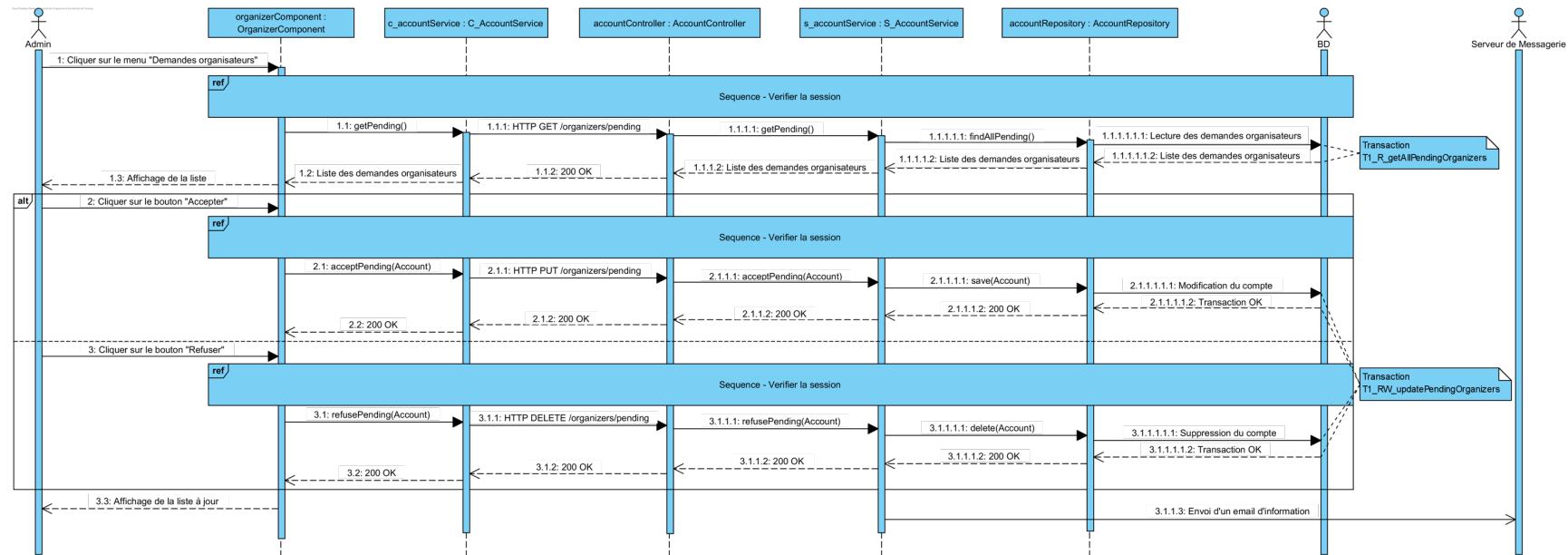
Dans notre modélisation de diagramme de séquence et communication, nous avons représenté la base de données comme acteurs alors qu'il s'agit d'une classe. Voici un exemple du changement qui devrait avoir lieu sur chaque diagramme de séquence.



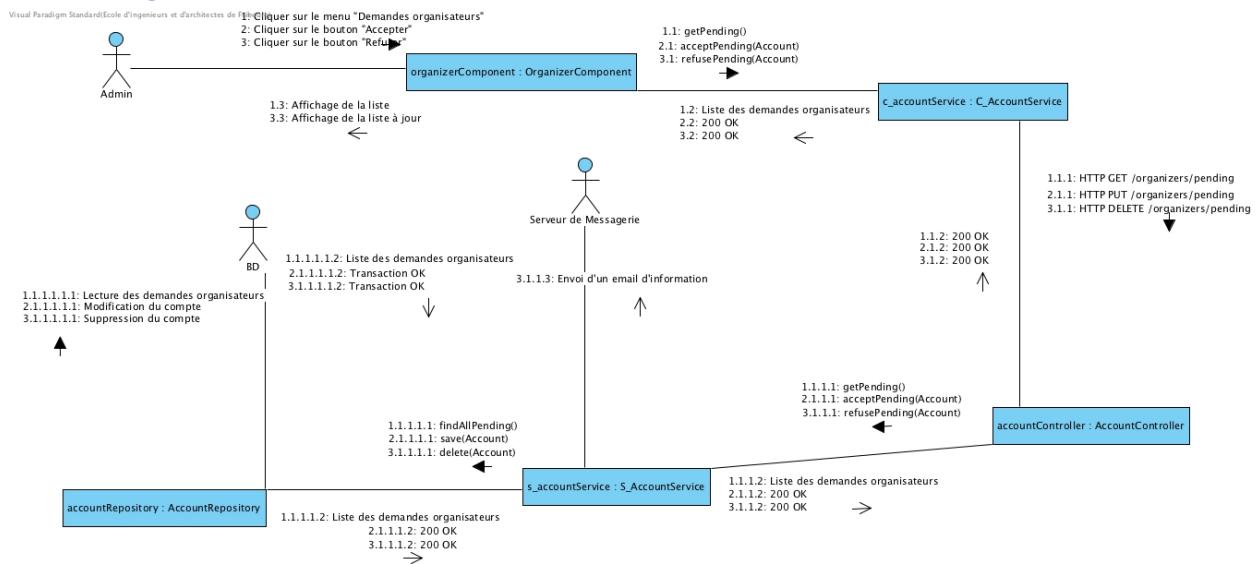
3.1 Cas 1 : Définir les privilèges

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

3.1.1 Diagramme de séquence



3.1.2 Diagramme de communication



3.1.3 Transactions

T1_R_getAllPendingOrganizers

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

Type de la transaction : Lecture

Isolation : READ COMMITTED

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 élevé.

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

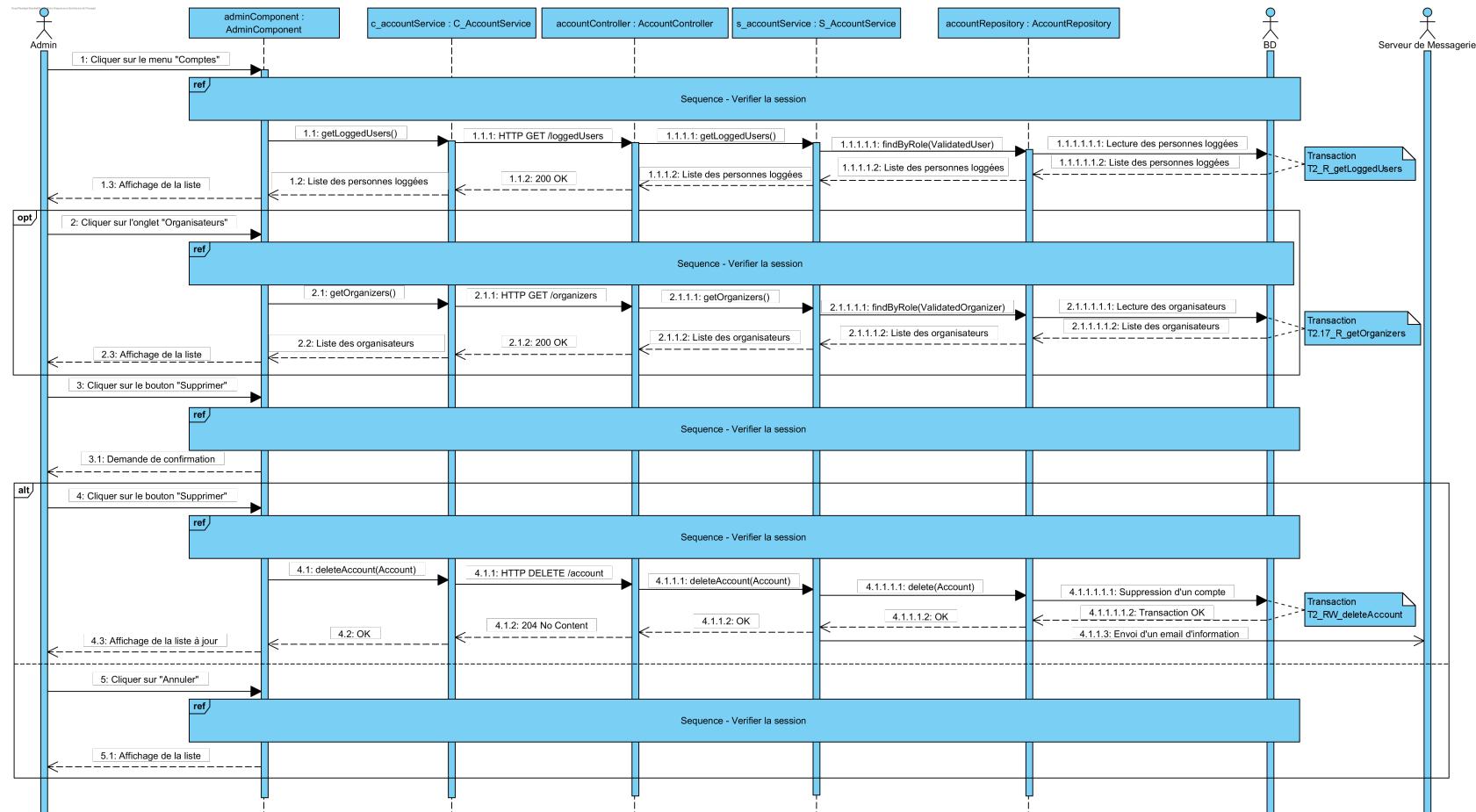
Fréquence : Rare

Période(s) : -

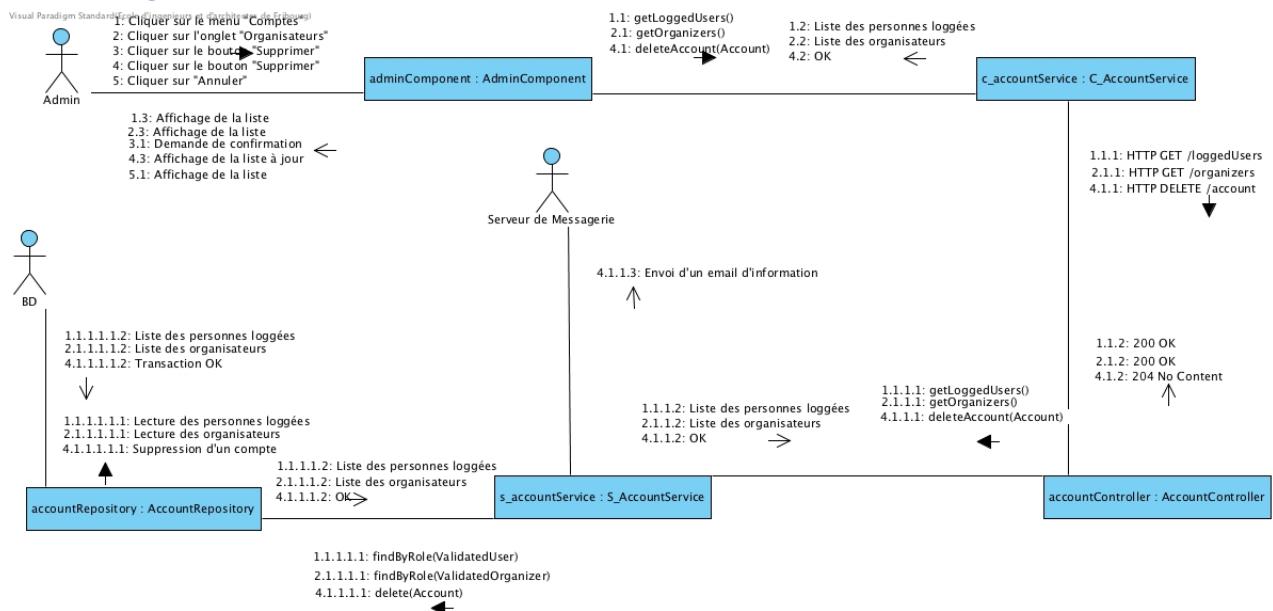
3.2 Cas 2 : Supprimer un compte

Un compte sur la plateforme peut être supprimé par un administrateur.

3.2.1 Diagramme de séquence



3.2.2 Diagramme de communication



3.2.3 Transactions

T2_R_getLoggedUsers

La transaction va chercher la liste des personnes loggées.

Type de la transaction : Lecture

Isolation : READ COMMITTED

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 : -

T2.17_R_getOrganizers

La transaction va chercher la liste des organisateurs.

Type de la transaction : Lecture

Isolation : READ COMMITTED

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

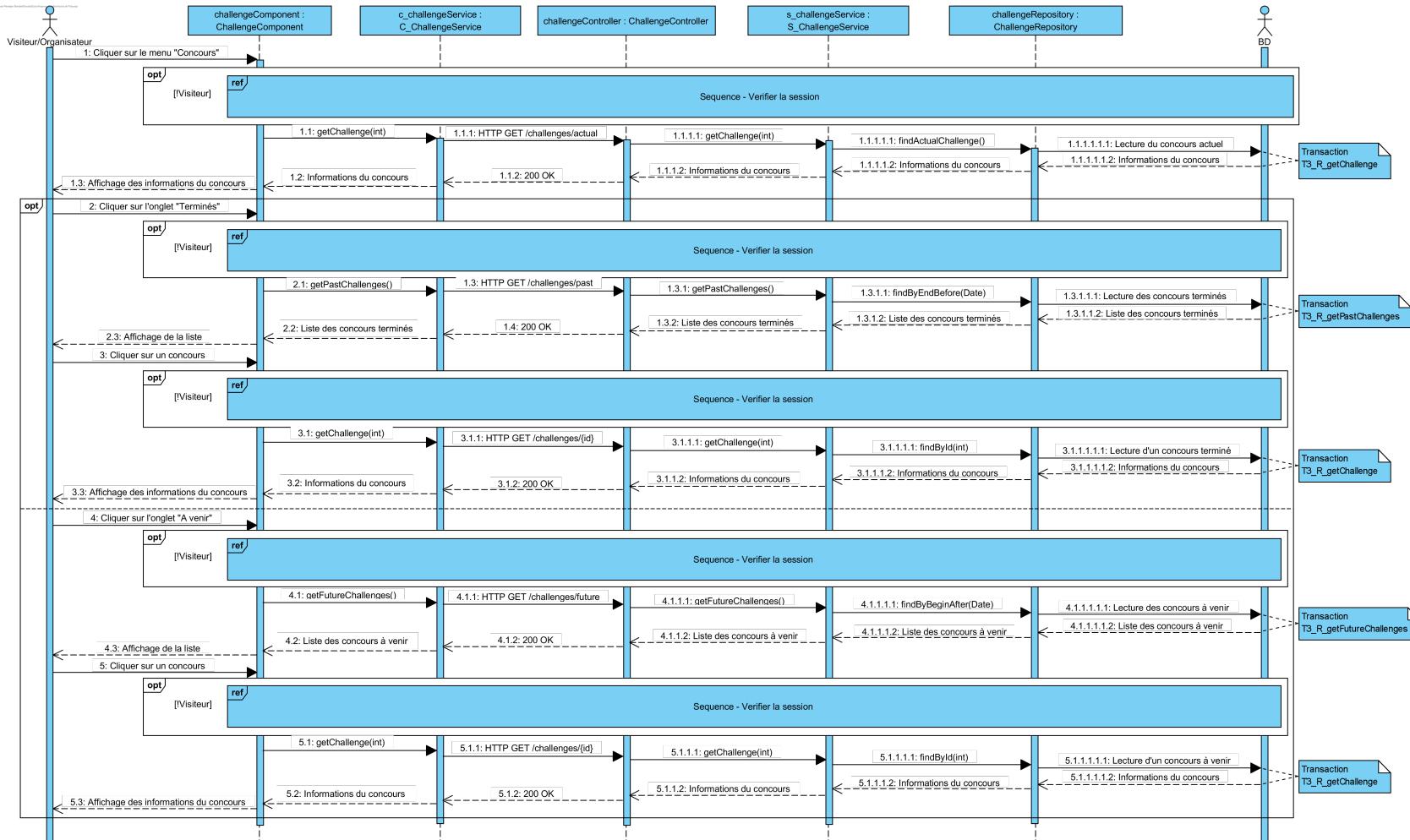
Fréquence : Rare

Période : -

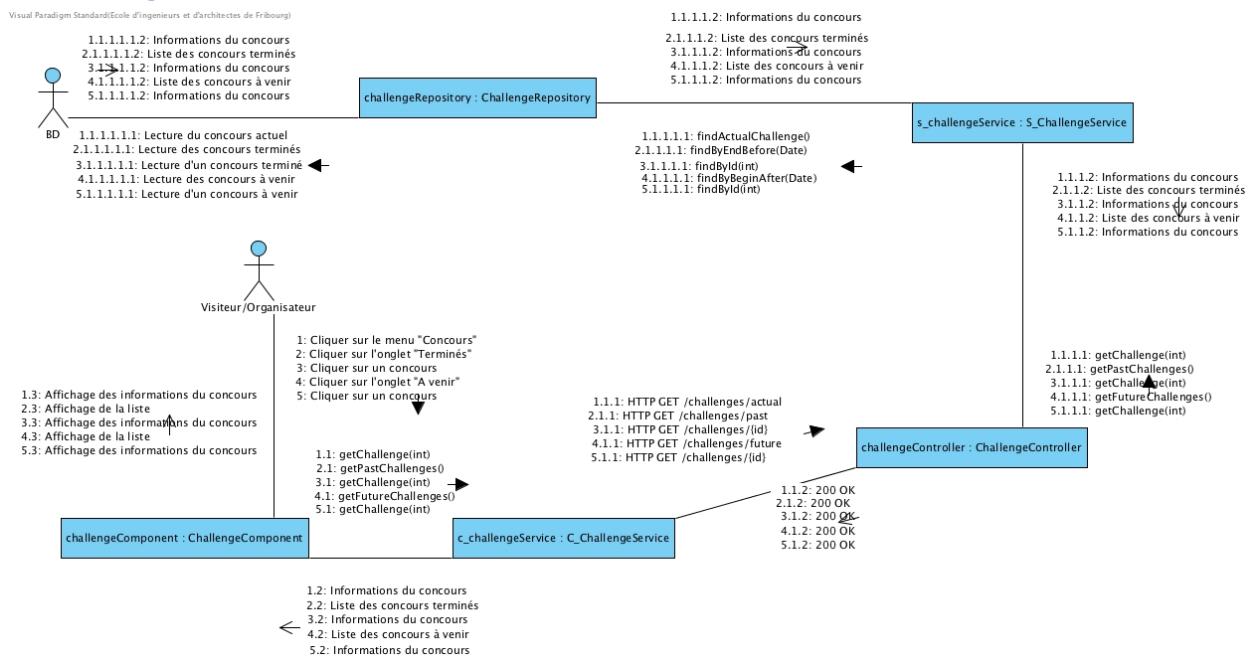
3.3 Cas 3 : Consulter les informations d'un concours

Toute personne peut consulter les informations des concours.

3.3.1 Diagramme de séquence



3.3.2 Diagramme de communication



3.3.3 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 COMMITTED

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 COMMITTED

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 COMMITTED

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

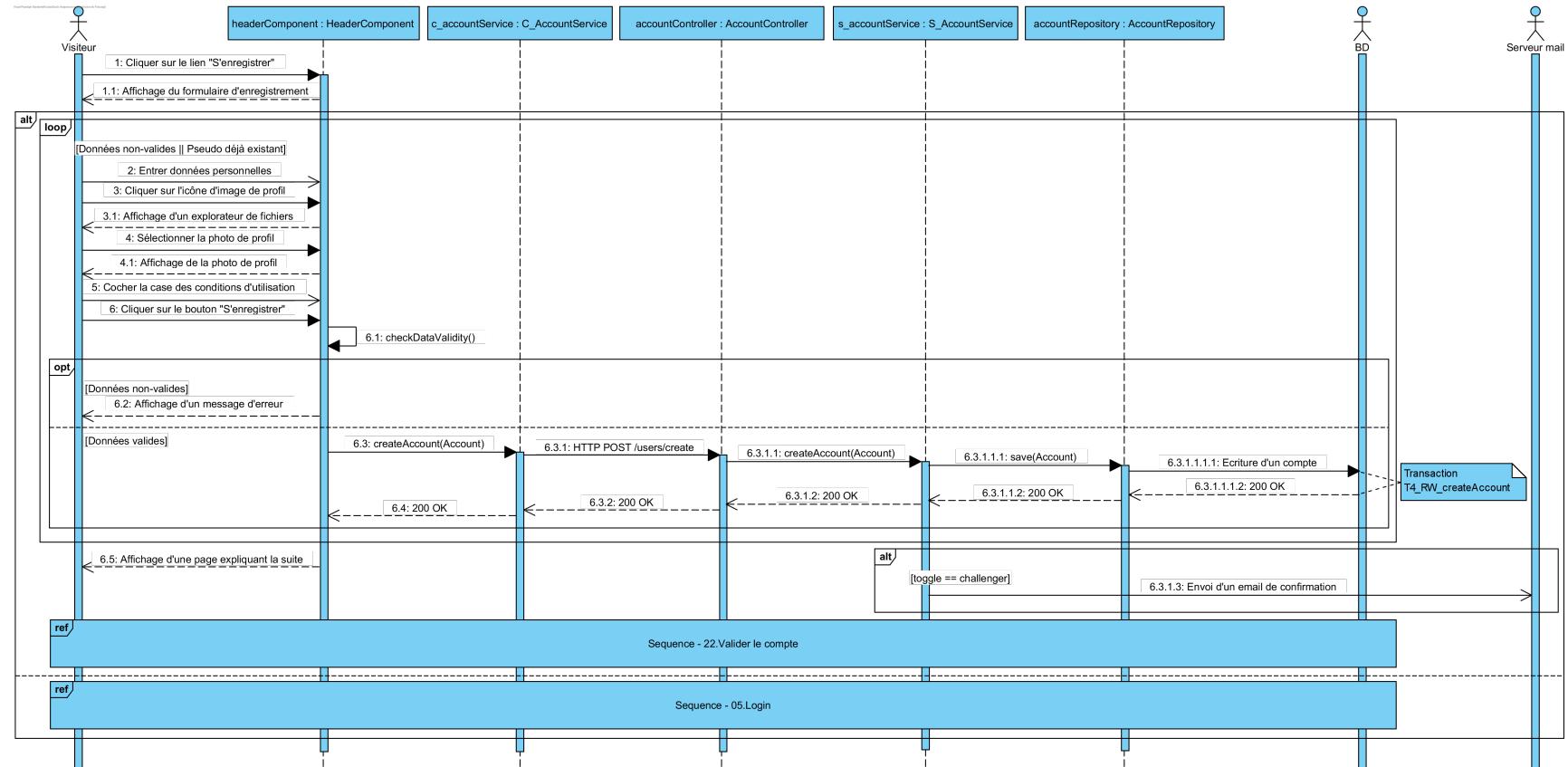
Fréquence : Elevé

Période : -

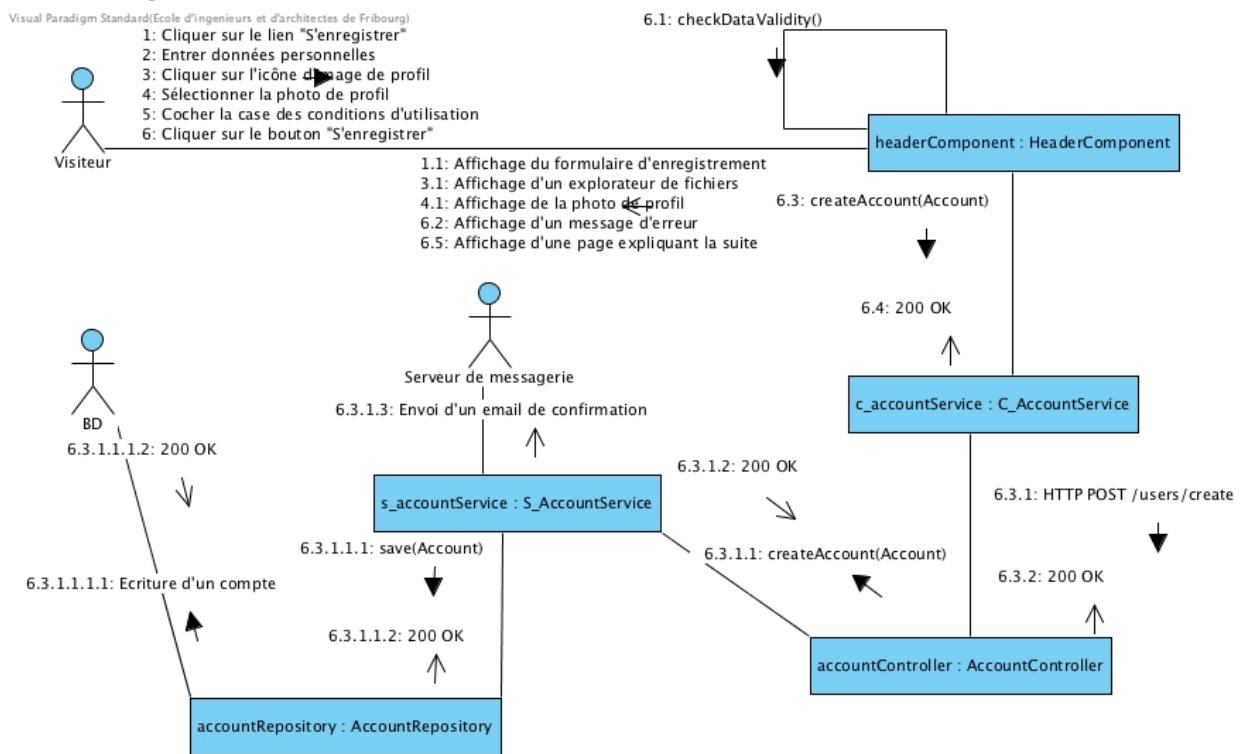
3.4 Cas 4 : S'enregistrer sur la plateforme

Une personne peut s'enregistrer sur la plateforme de concours.

3.4.1 Diagramme de séquence



3.4.2 Diagramme de communication



3.4.3 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

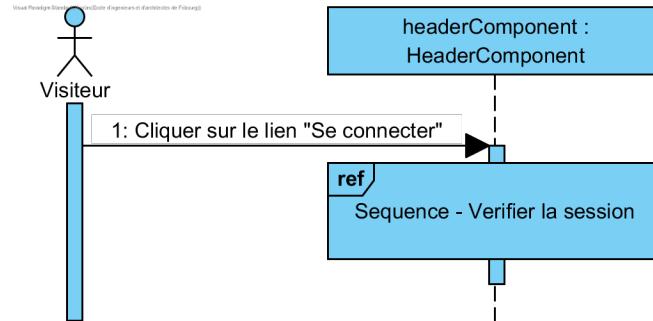
Fréquence : Moyenne

Période : -

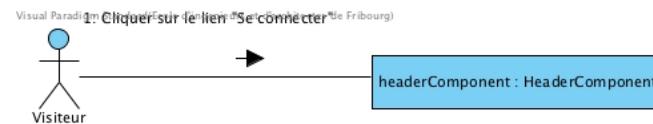
3.5 Cas 5 : Login

Une personne non loguée peut se loguer sur la plateforme de concours.

3.5.1 Diagramme de séquence



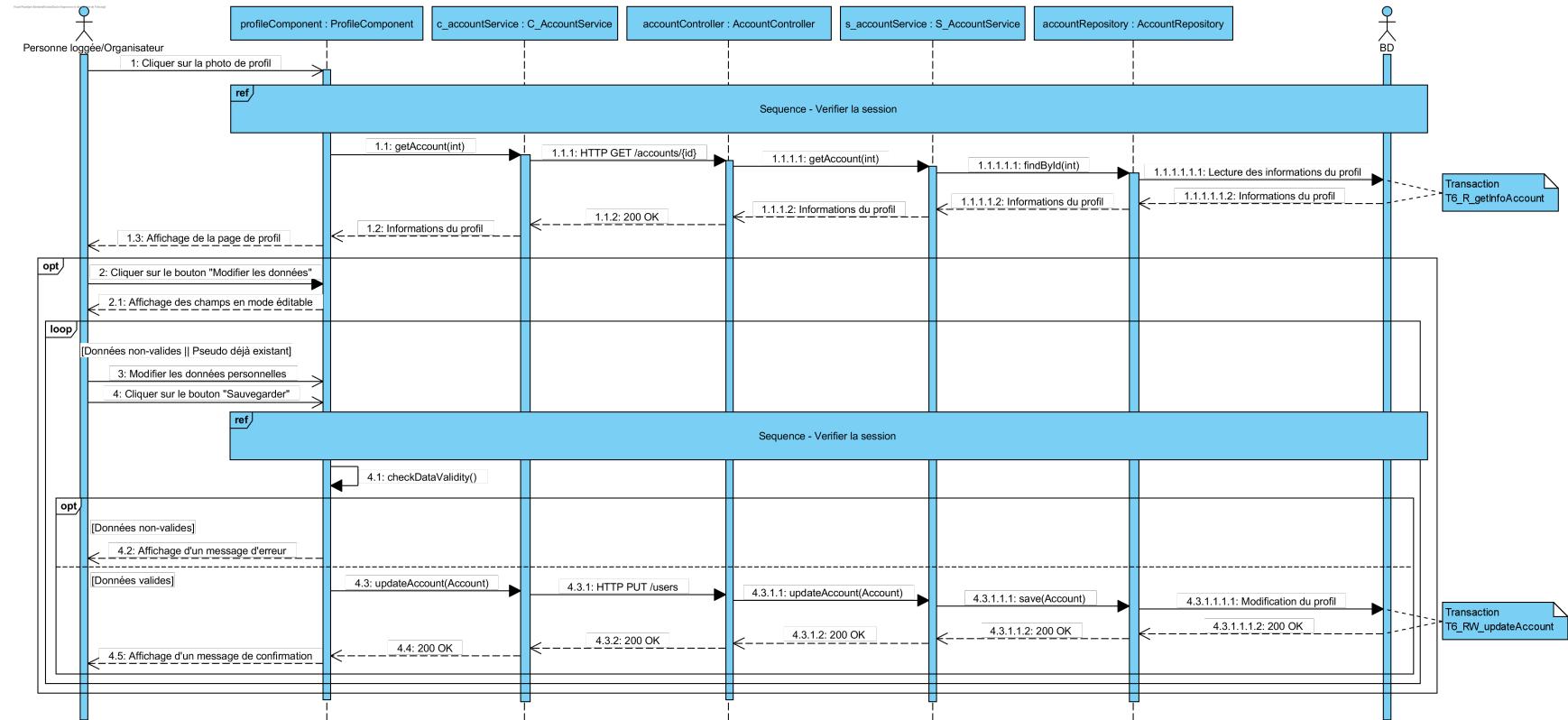
3.5.2 Diagramme de communication



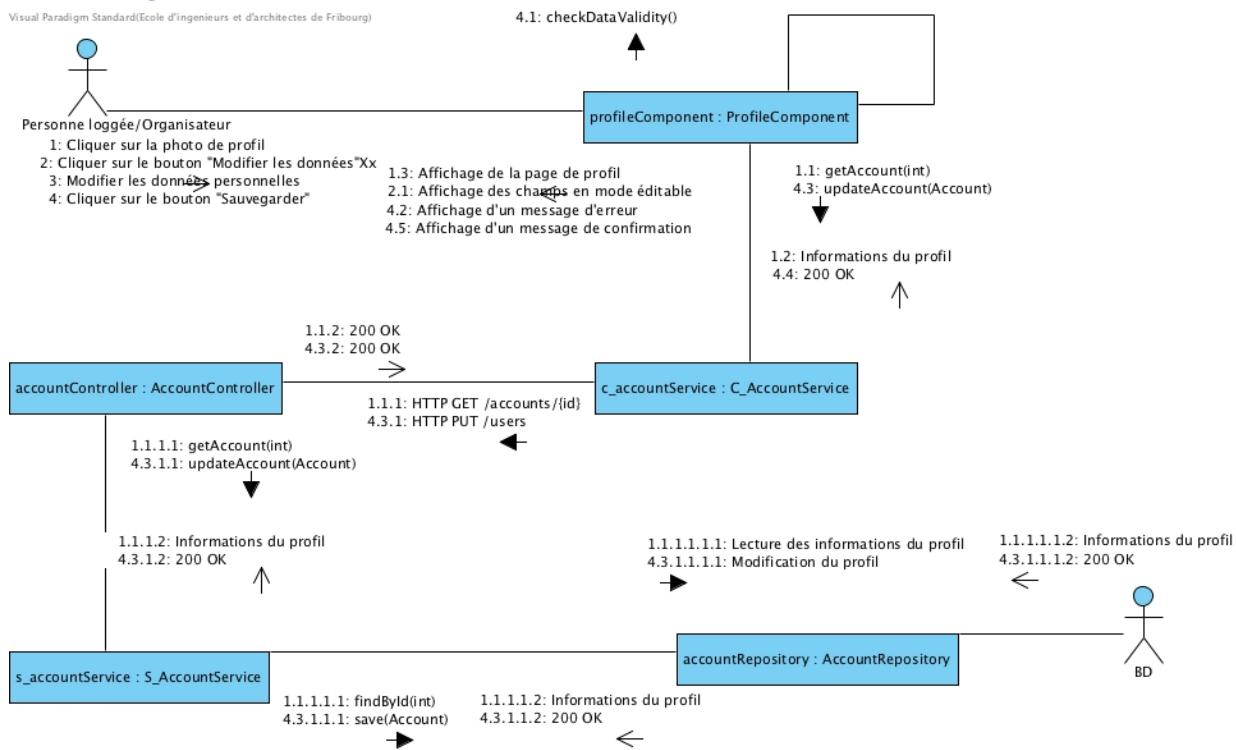
3.6 Cas 6 : Consulter / Modifier les données de son compte

Une personne loguée sur la plateforme peut paramétrer son profil.

3.6.1 Diagramme de séquence



3.6.2 Diagramme de communication



3.6.3 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

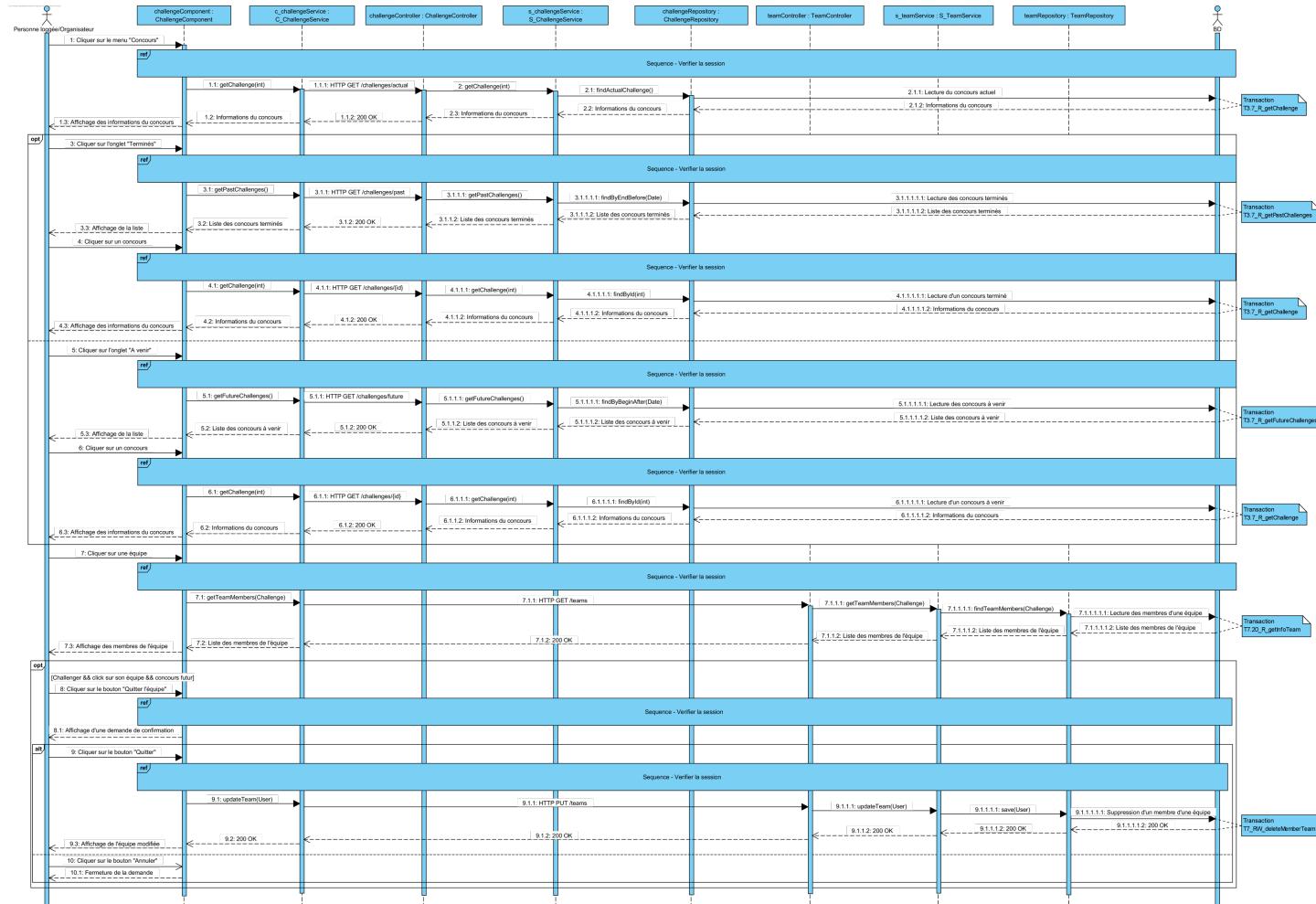
Fréquence : Faible

Période : -

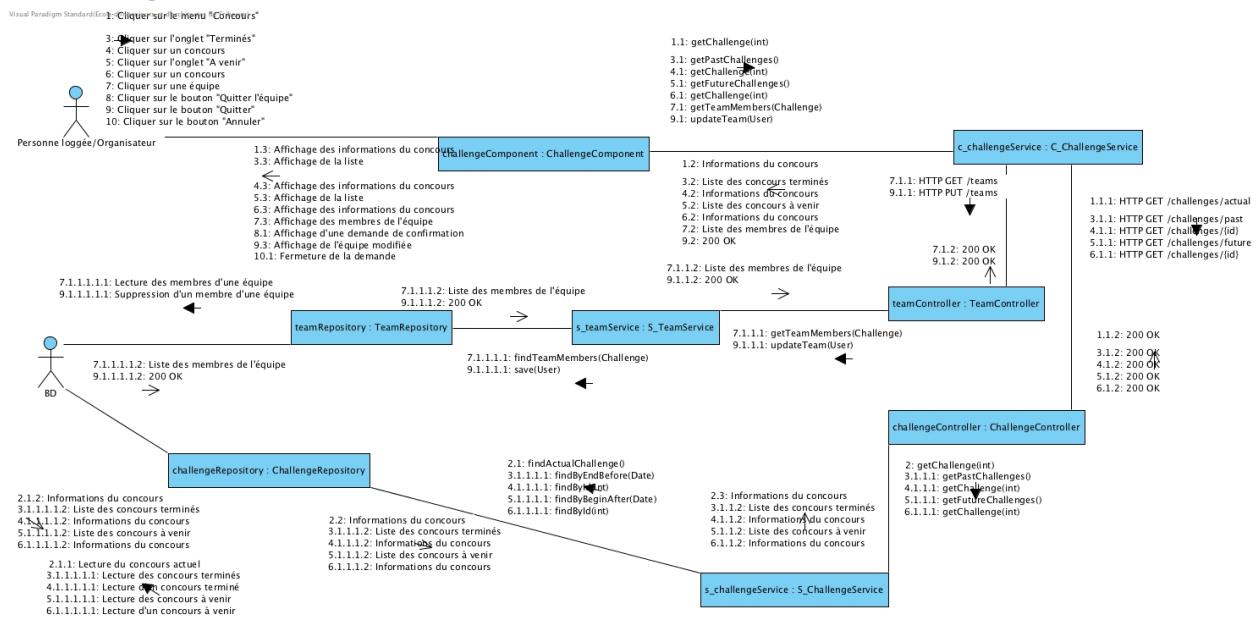
3.7 Cas 7 : Consulter les équipes

Une personne loguée peut consulter les membres des équipes inscrites à un concours.

3.7.1 Diagramme de séquence



3.7.2 Diagramme de communication



3.7.3 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 COMMITTED

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

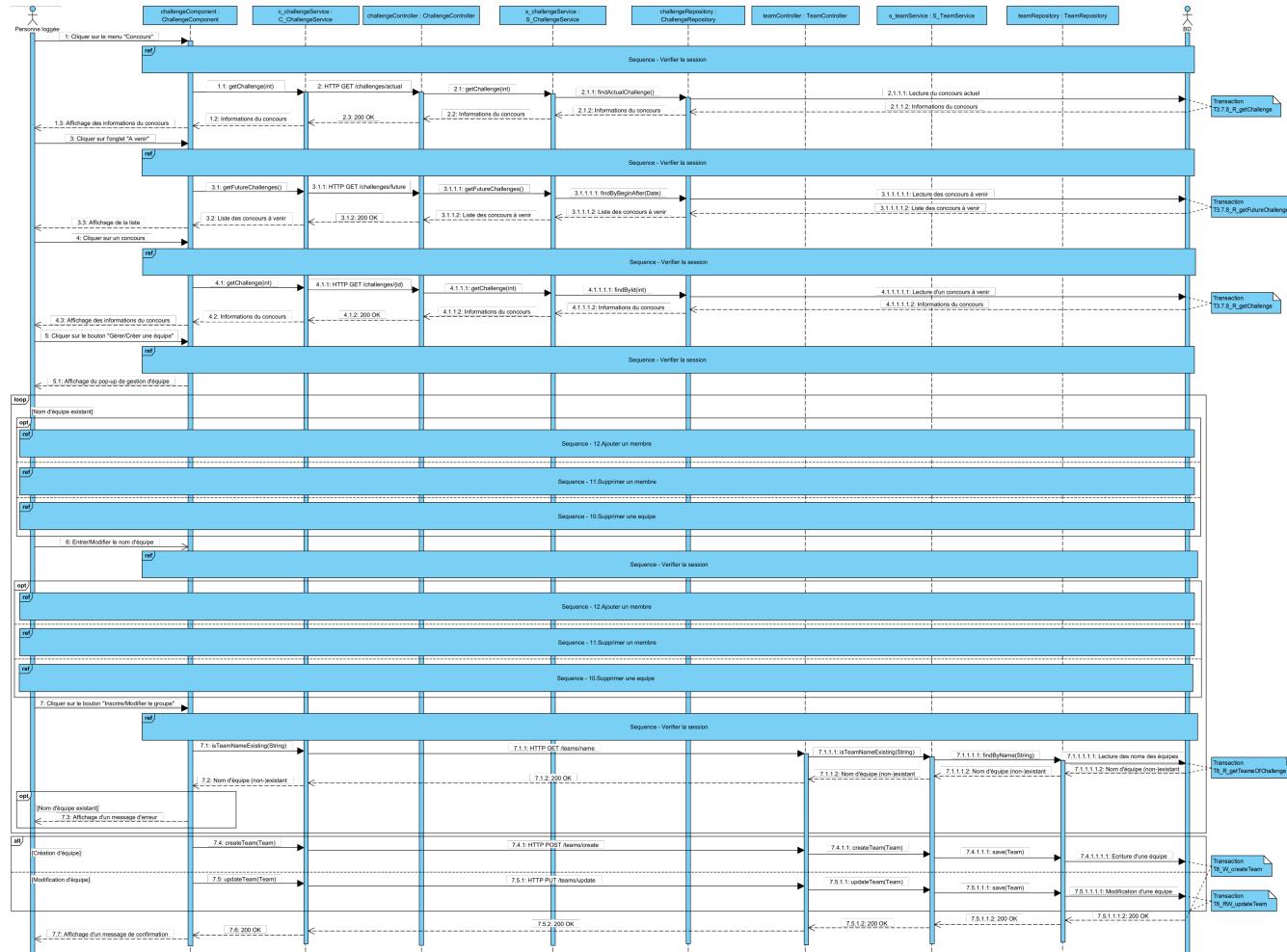
Fréquence : Faible

Période : -

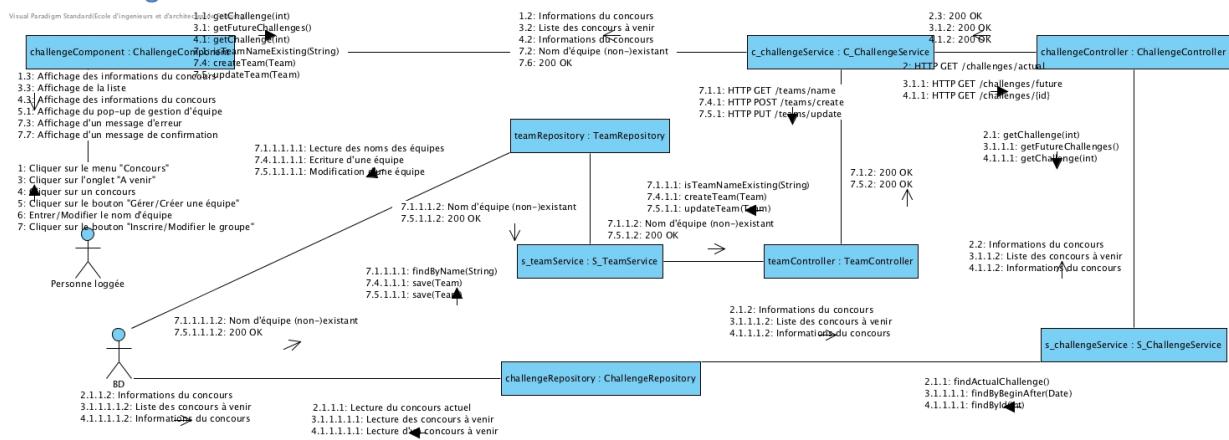
3.8 Cas 8 : Gérer une équipe

Une personne peut créer une équipe pour un concours et la modifier.

3.8.1 Diagramme de séquence



3.8.2 Diagramme de communication



3.8.3 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 COMMITTED

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.

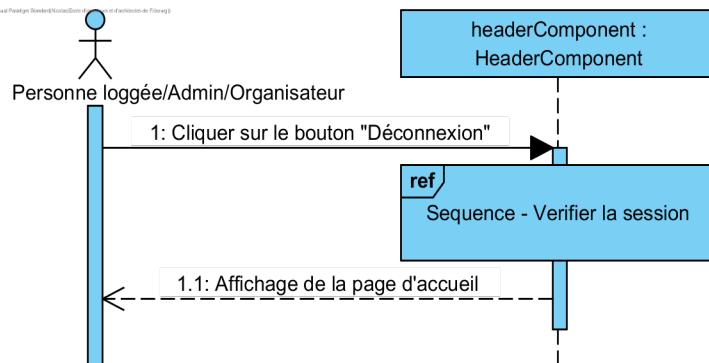
Tables touchées : Account,, Team, Challenge, Account-Team

Fréquence : Elevé

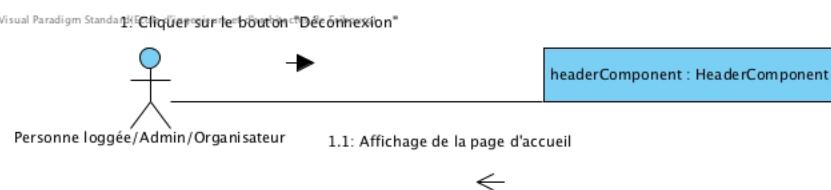
3.9 Cas 9 : Logout

Une personne connectée sur la plateforme peut se déconnecter.

3.9.1 Diagramme de séquence



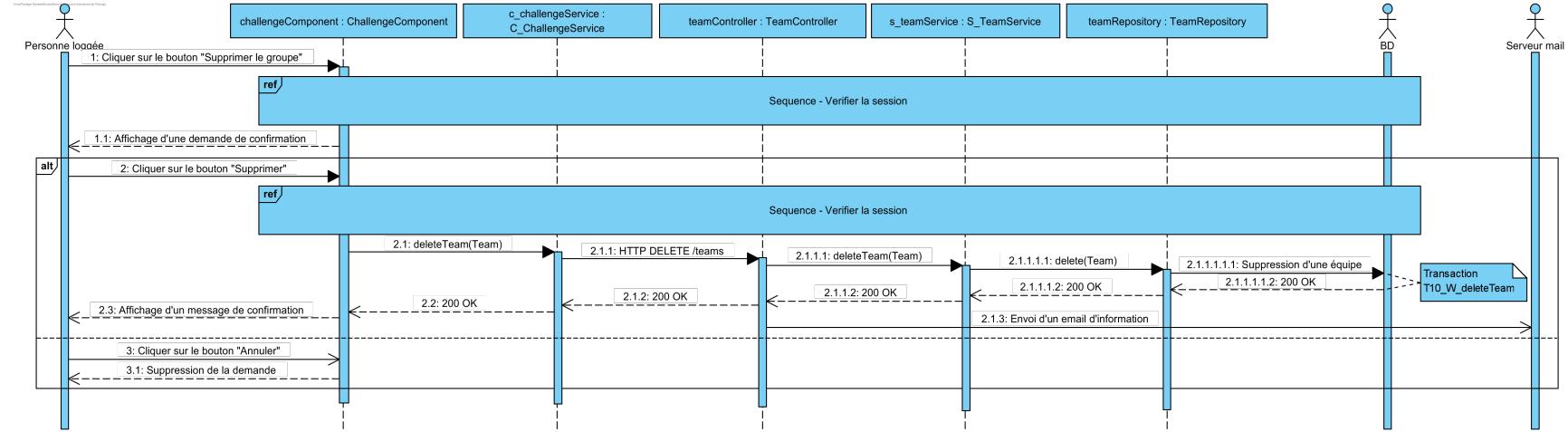
3.9.2 Diagramme de communication



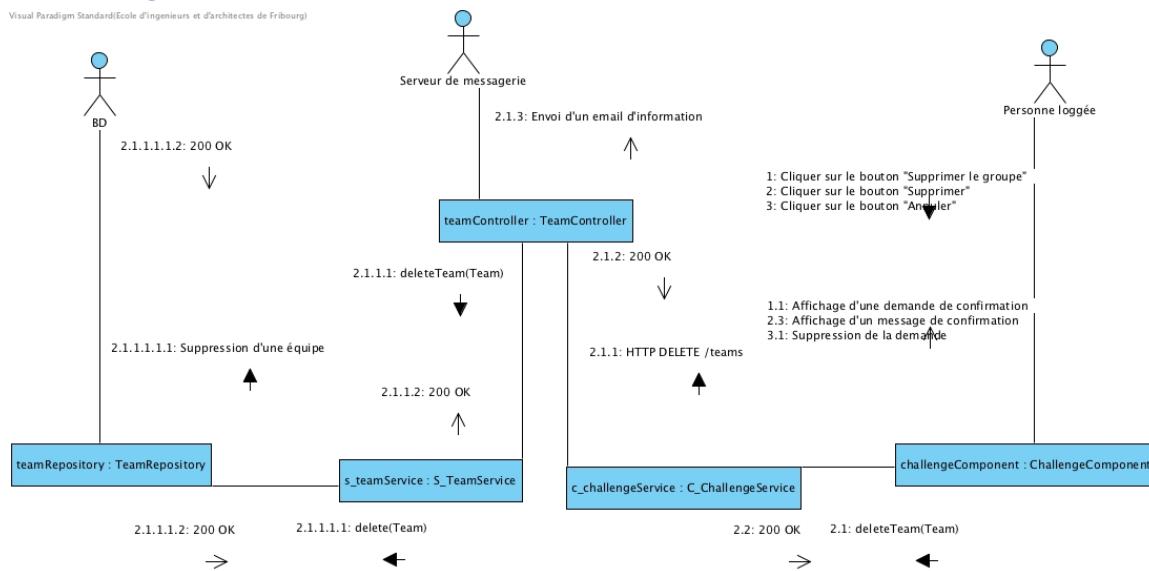
3.10 Cas 10 : Supprimer les équipes

Une personne loguée ayant créé une équipe au préalable peut la supprimer.

3.10.1 Diagramme de séquence



3.10.2 Diagramme de communication



3.10.3 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.

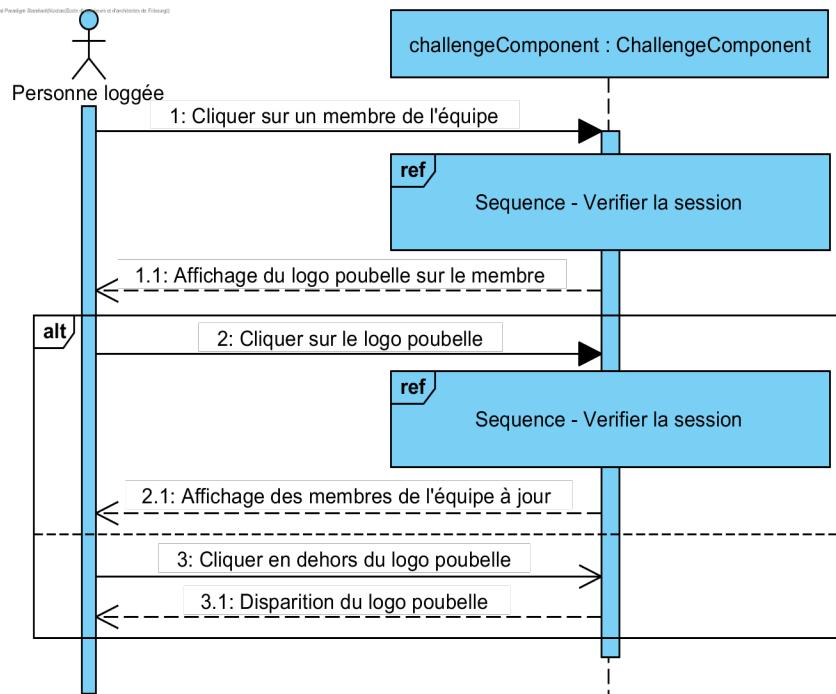
Tables touchées : Account,, Team, Challenge, Account-Team

Fréquence : Elevé

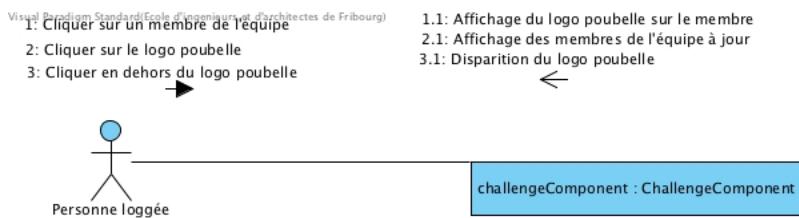
3.11 Cas 11 :: Supprimer un membre

Une personne loguée ayant créé une équipe au préalable peut en supprimer un membre.

3.11.1 Diagramme de séquence



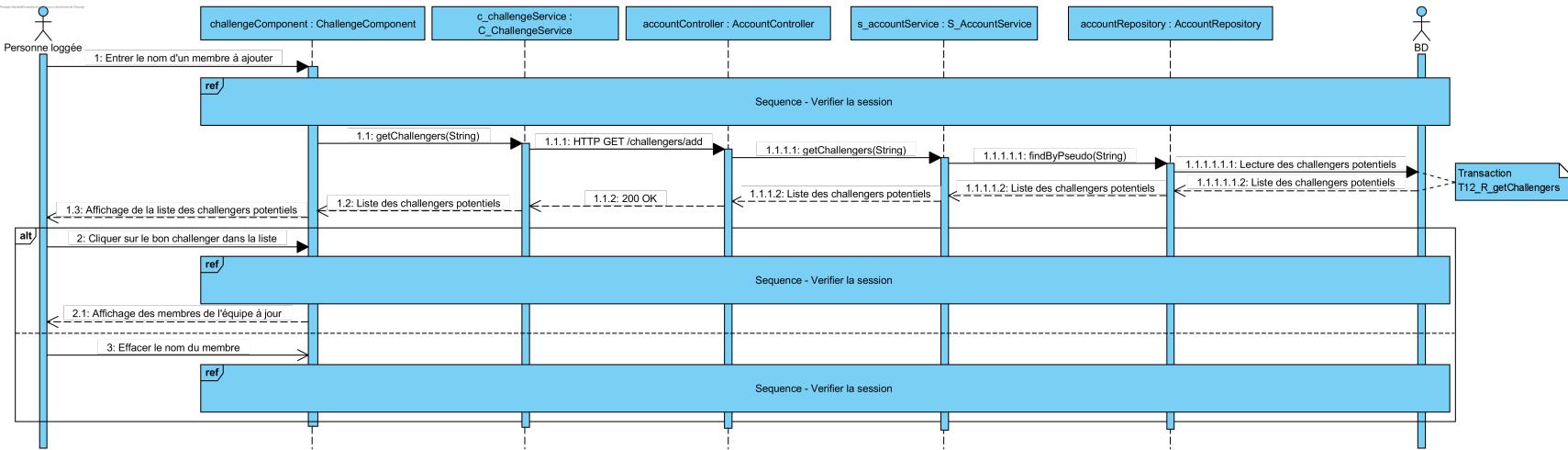
3.11.2 Diagramme de communication



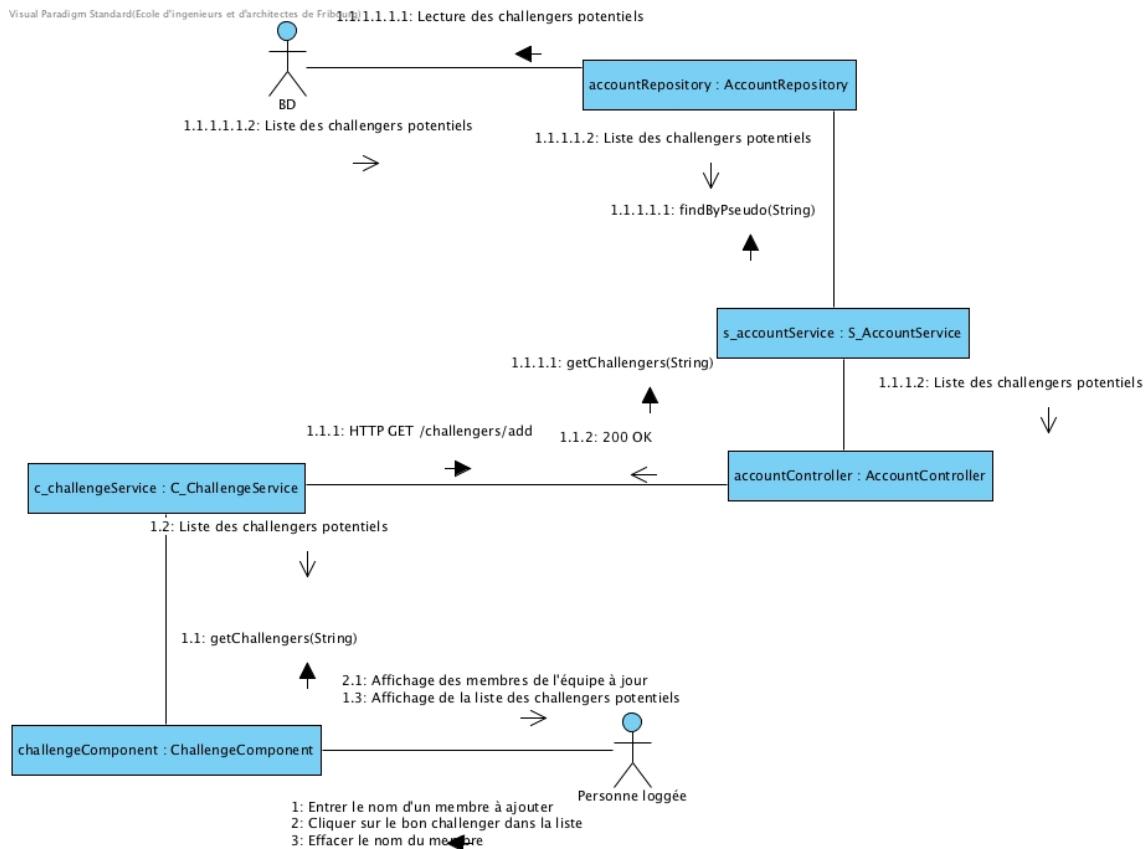
3.12 Cas 12 : Ajouter un membre

Une personne loguée ayant créé une équipe au préalable peut y ajouter un membre.

3.12.1 Diagramme de séquence



3.12.2 Diagramme de communication



3.12.3 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 personnes pouvant s'inscrire au challenge. Il n'est pas nécessaire d'avoir un niveau d'isolation supplémentaire.

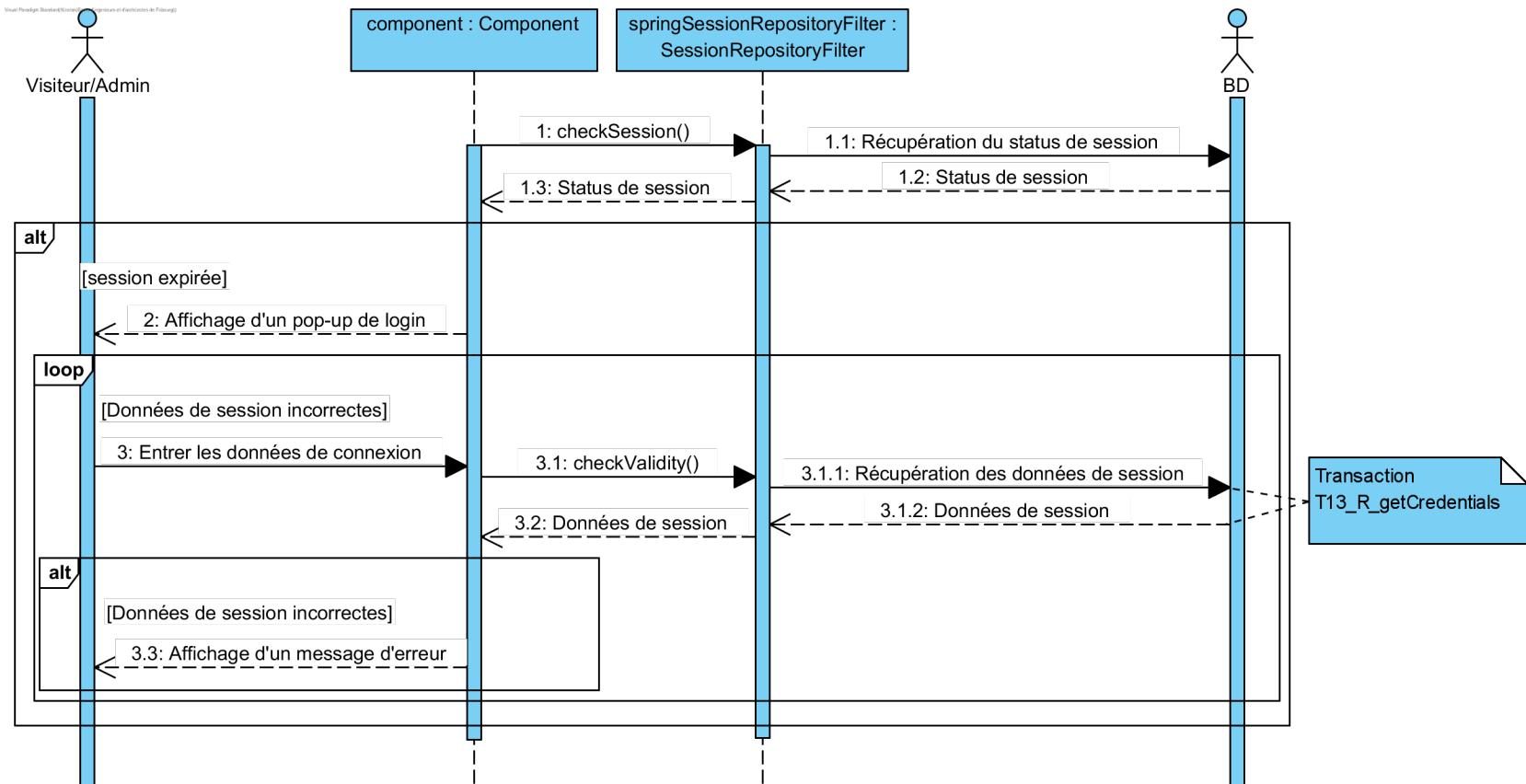
Tables touchées : Account, Role, Team, Challenge, Account-Team

Fréquence : Moyenne

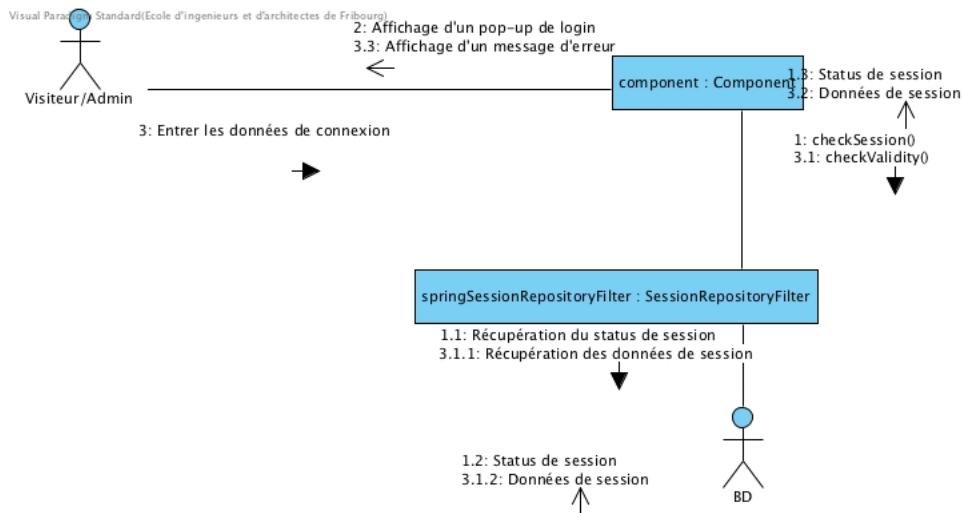
3.13 Cas 13 : Vérifier la session

Lorsqu'une personne fait une action sur la plateforme, le statut de la session doit être vérifié.

3.13.1 Diagramme de séquence



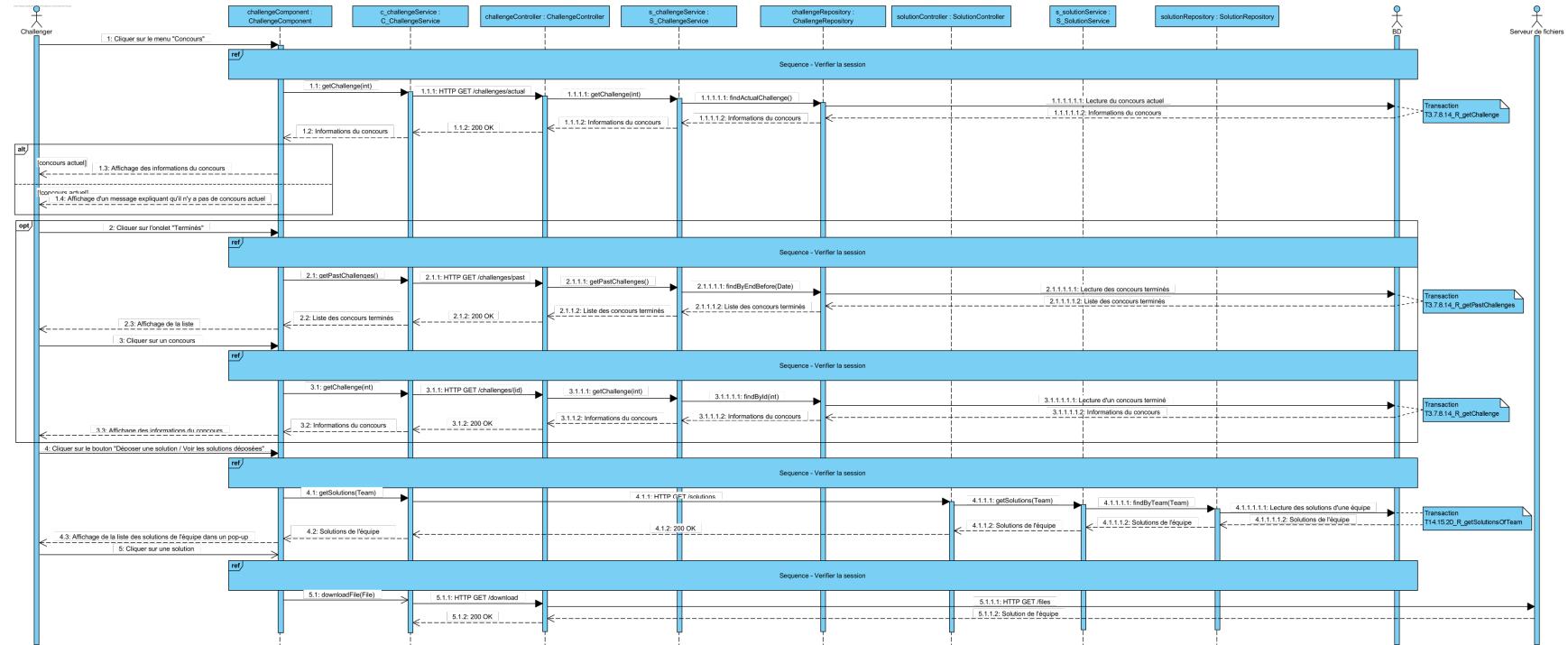
3.13.2 Diagramme de communication



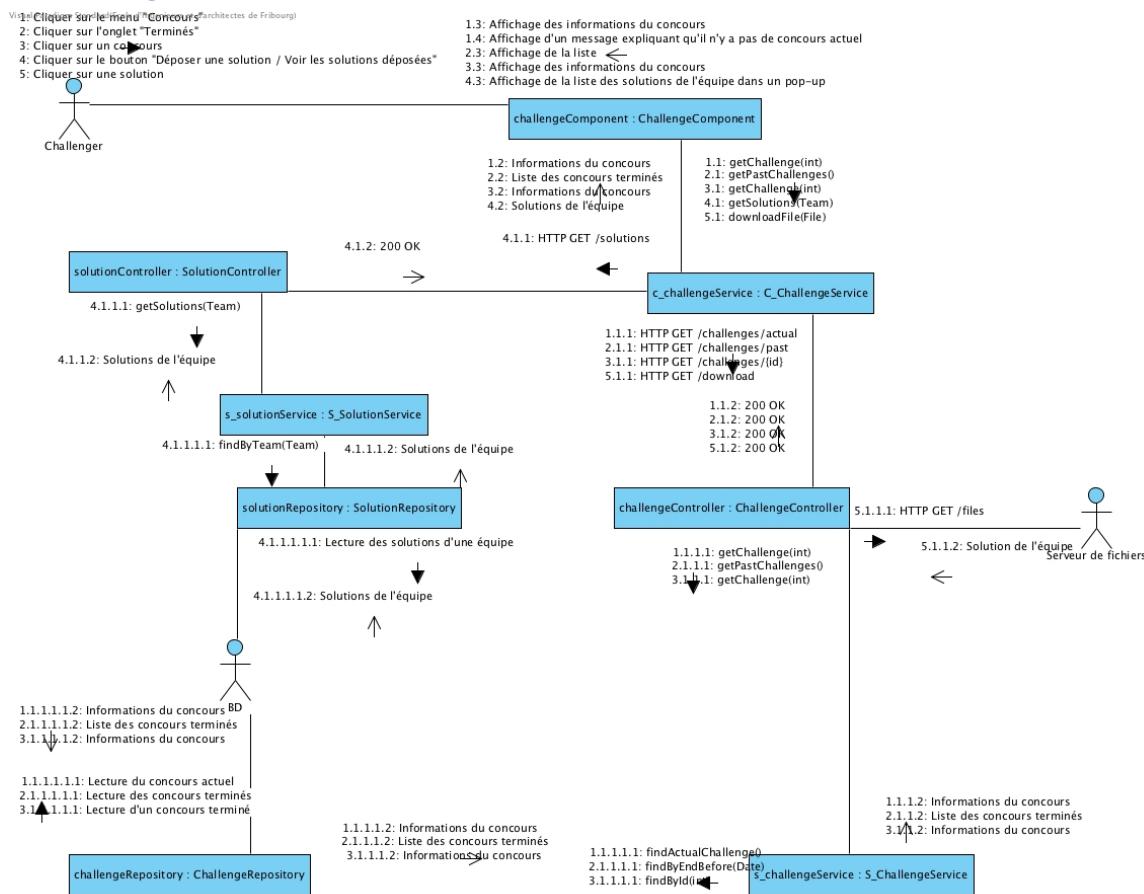
3.14 Cas 14 : Consulter les solutions déposées

Les solutions d'un concours peuvent être consultées par l'équipe qui les a déposées.

3.14.1 Diagramme de séquence



3.14.2 Diagramme de communication



3.14.3 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.

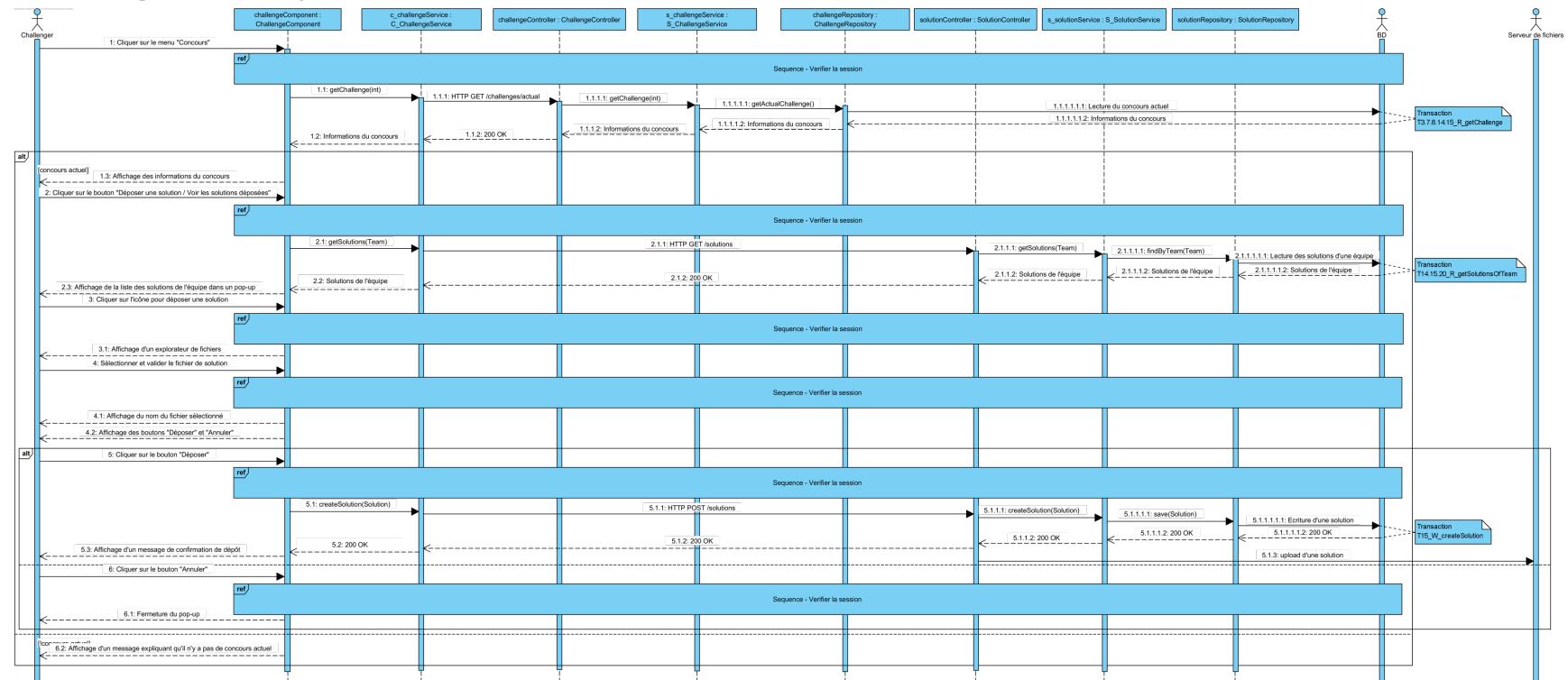
Tables touchées : Account,, Team, Challenge, Solution Account-Team

Fréquence : Elevé

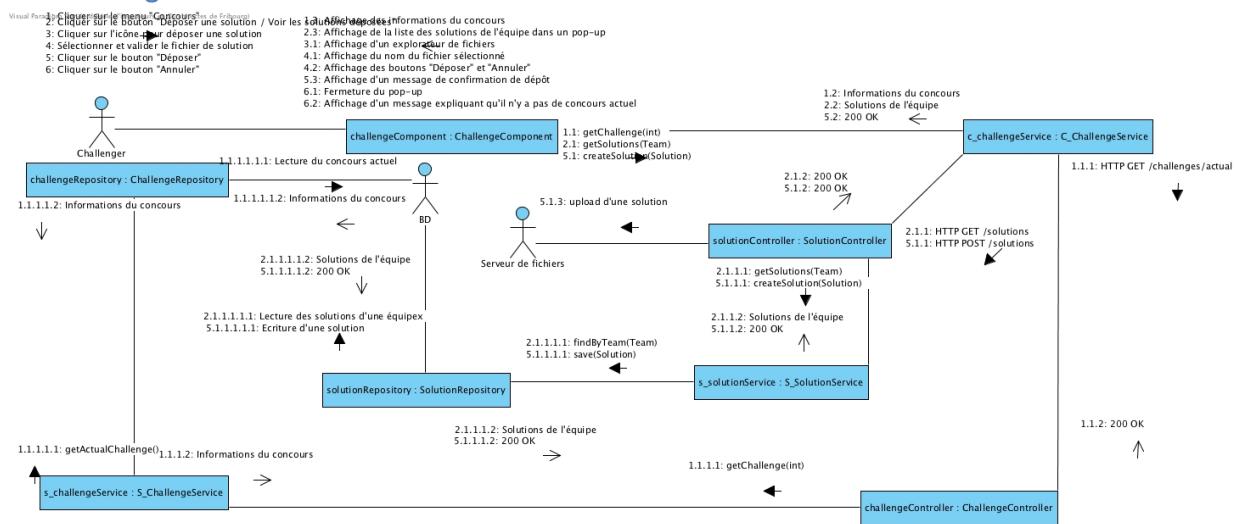
3.15 Cas 15 : Déposer une solution

Une personne participant à un concours peut déposer une solution.

3.15.1 Diagramme de séquence



3.15.2 Diagramme de communication



3.15.3 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

La transaction va rajouter une solution d'une équipe dans la base de données

Type de la transaction : Lecture et Ecriture

Isolation : SERIALIZABLE

La transaction va ajouter une solution, afin d'éviter que deux solutions soient ajoutées en même temps ou qu'il y aie des modifications durant la transaction, il faut le niveau d'isolation 3.

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

Fréquence : Elevé

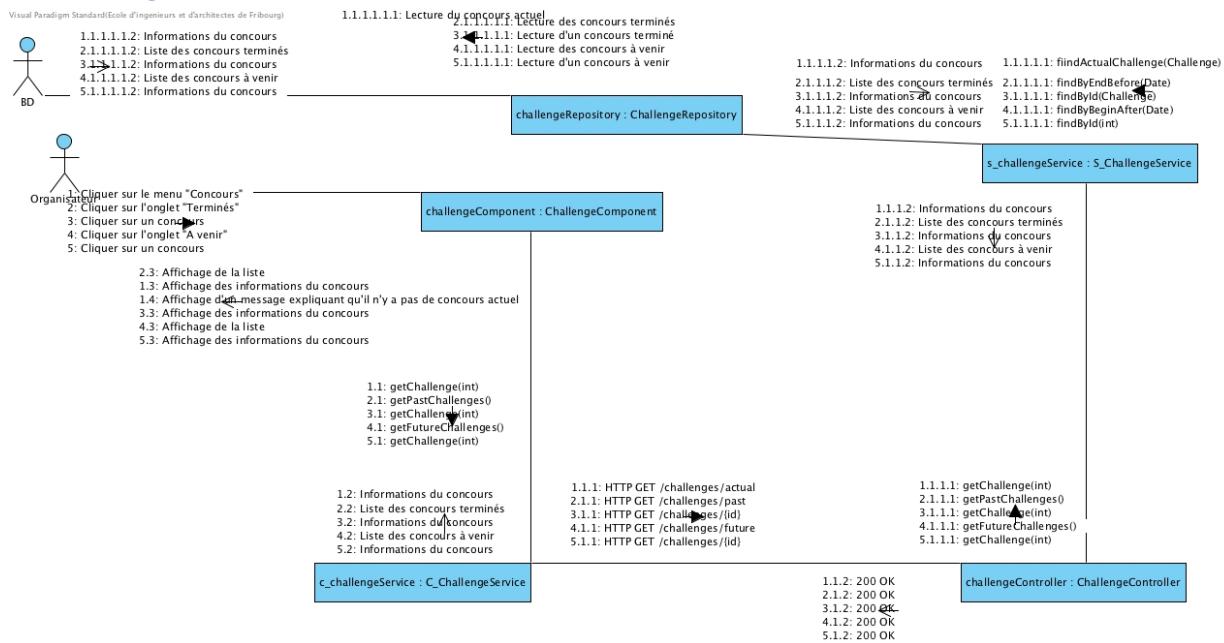
3.16 Cas 16 : Consulter les concours

Un organisateur peut consulter tous les concours et surtout ceux qu'il a créés.

3.16.1 Diagramme de séquence



3.16.2 Diagramme de communication



3.16.3 Transactions

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 »

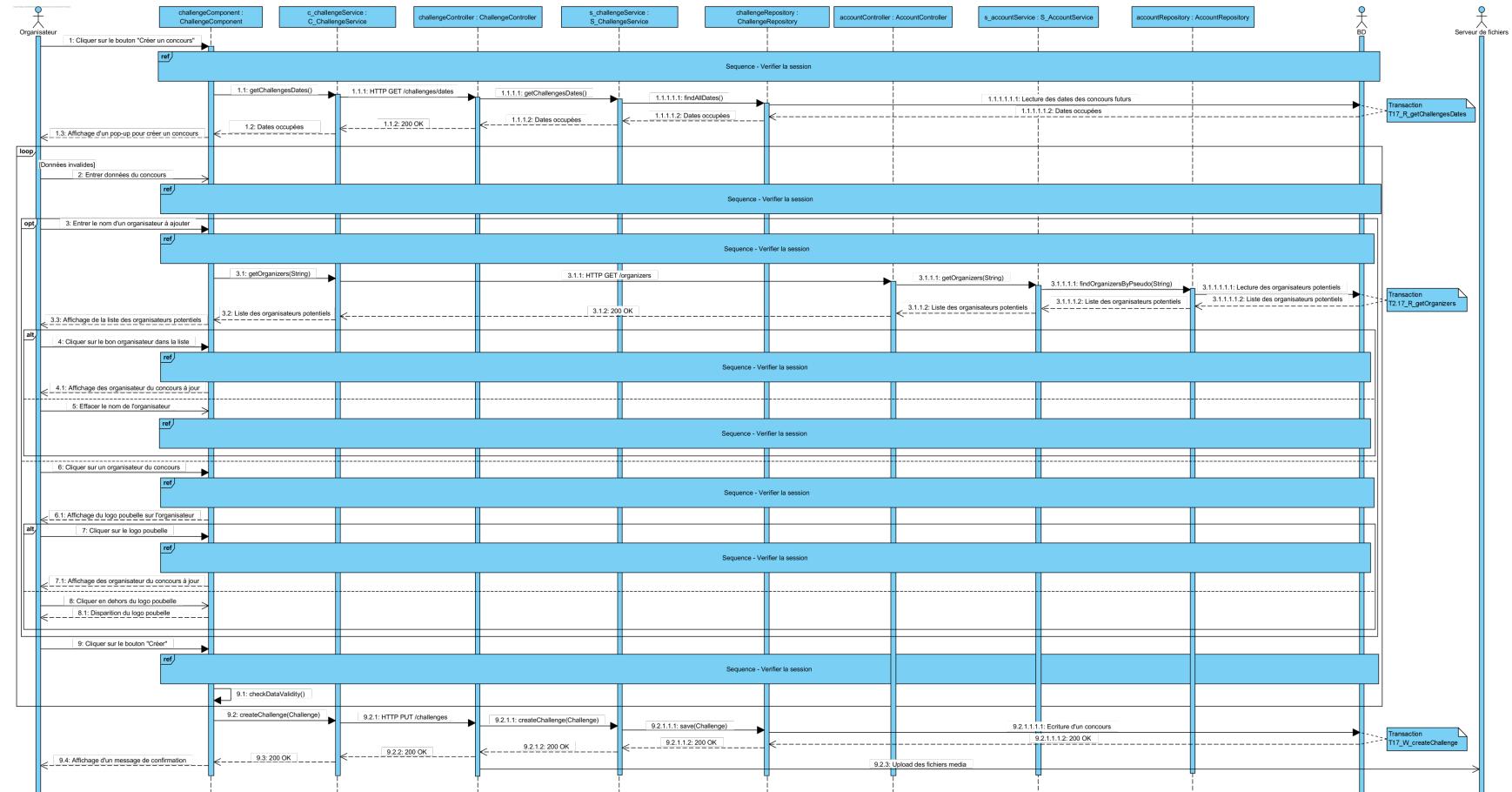
T3.7.8.14_R_getPastChallenges : Cette transaction est décrite dans le chapitre « 3.3.3 transactions » du cas d'utilisation « Consulter les informations d'un concours »

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 »

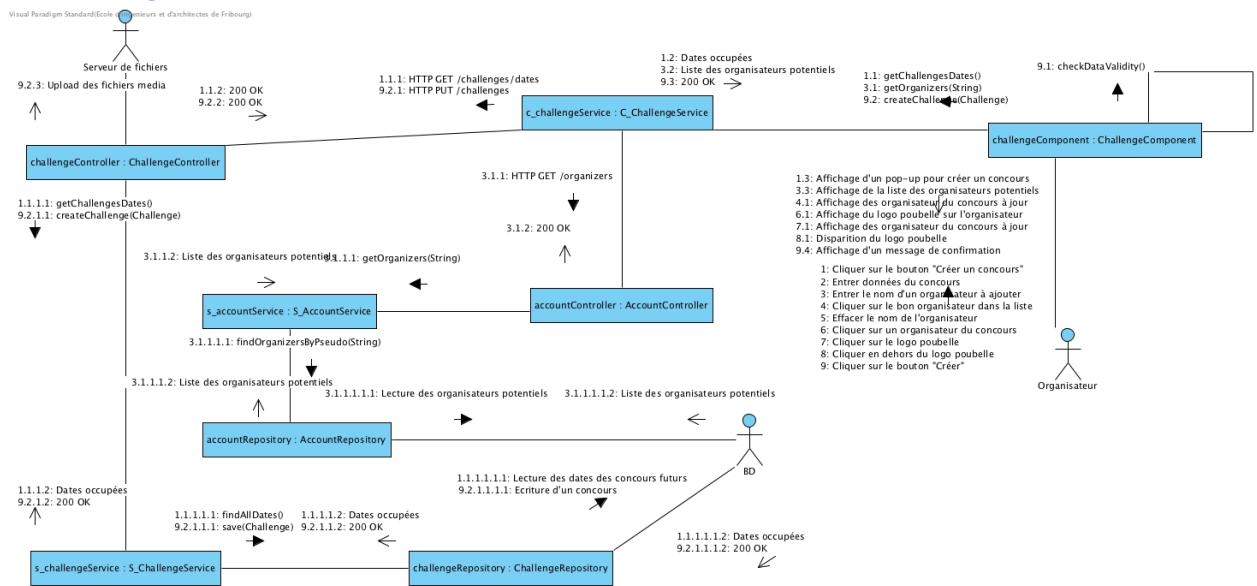
3.17 Cas 17 : Créer un concours

Un organisateur peut créer un concours seul ou en équipe.

3.17.1 Diagramme de séquence



3.17.2 Diagramme de communication



3.17.3 Transaction

T17_R_getFuturChallengesDates

La transaction va chercher les dates des prochains challenge

Type de la transaction : Lecture

Isolation : READ COMMITTED

La transaction va simplement rechercher les dates des prochains challenges, il n'est pas nécessaire d'avoir un niveau d'isolation supplémentaire

Tables touchées : Challenge

Fréquence : Rare

T2.17_R_getOrganizers : Cette transaction est décrite dans le chapitre « 3.2.3 transactions » du cas d'utilisation « Supprimer un compte »

T17_W_createChallenge

La transaction va créer un challenge dans la base de données

Type de la transaction : Lecture et Ecriture

Isolation : SERIALIZABLE

La transaction va ajouter un challenge, afin d'éviter que deux challenges soient ajoutées en même temps ou qu'il y ait des modifications durant la transaction, il faut le niveau d'isolation 3.

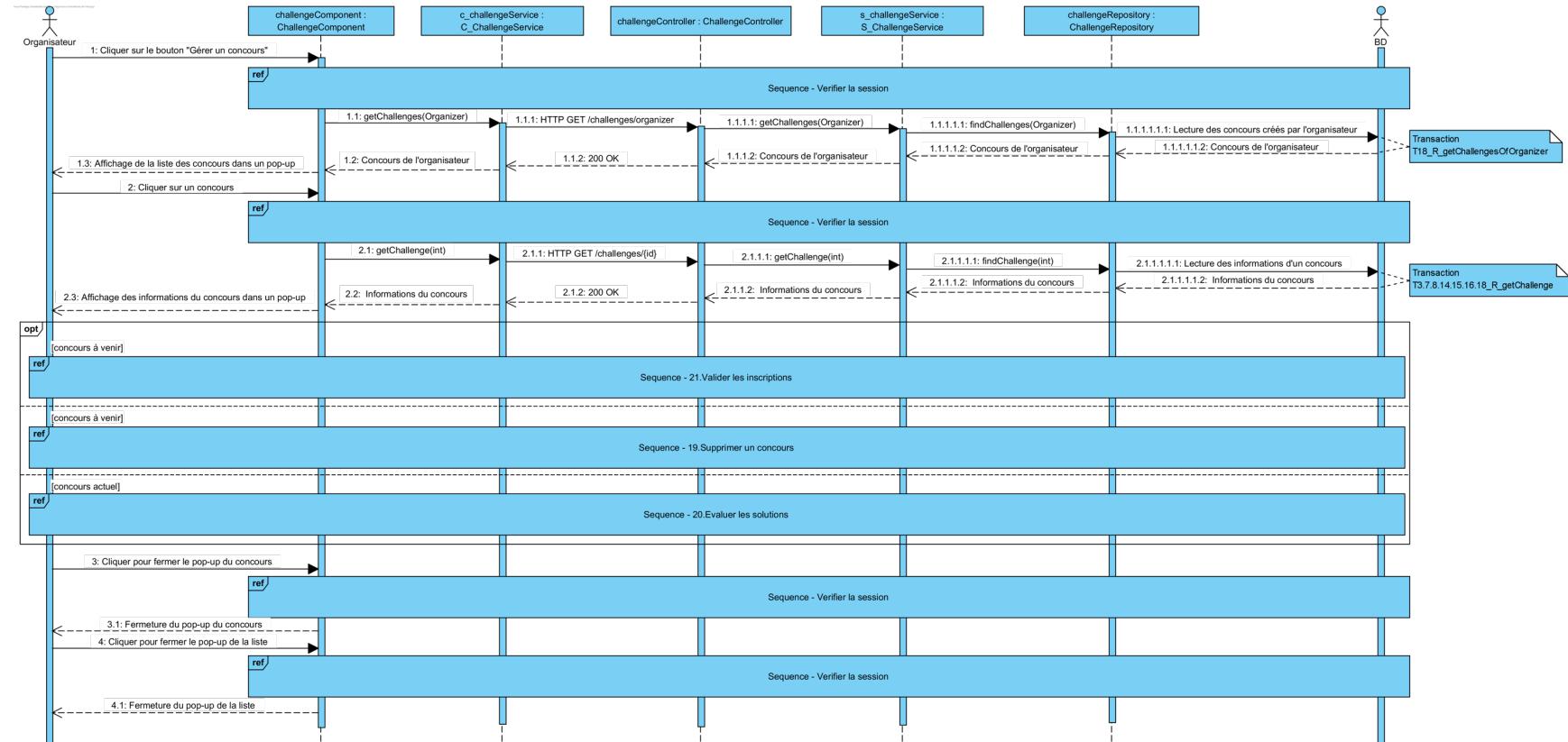
Tables touchées : Account,Role , Challenge, Data, Challenge-organizer

Fréquence : Rare

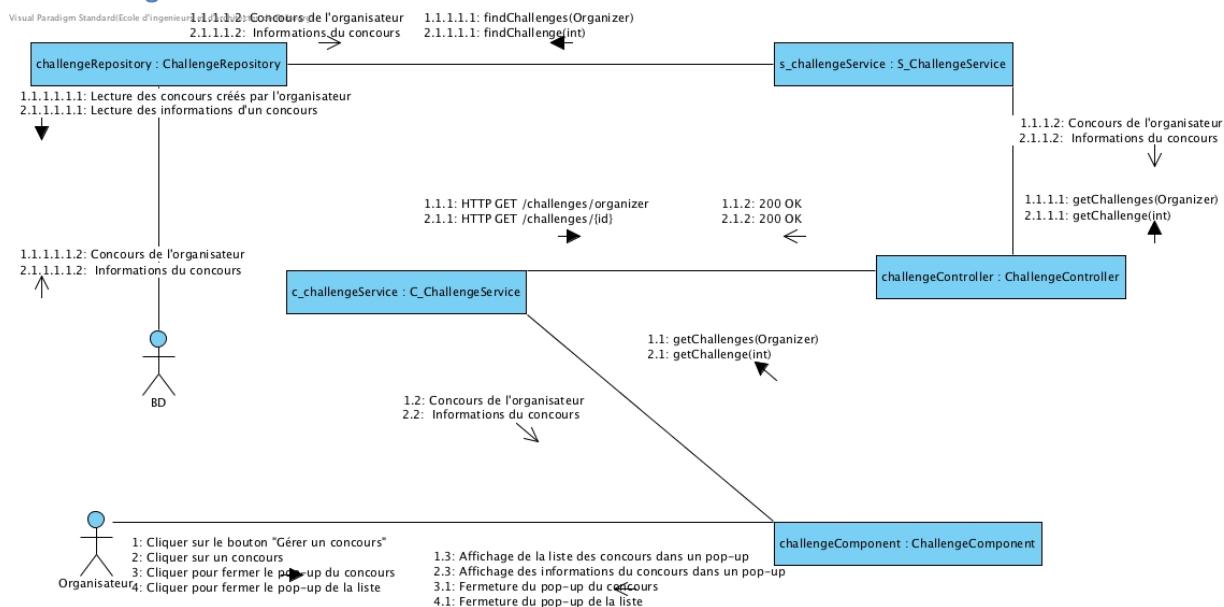
3.18 Cas 18 : Gérer un concours

Un organisateur peut modifier un concours créé au préalable et le supprimer seul ou en équipe.

3.18.1 Diagramme de séquence



3.18.2 Diagramme de communication



3.18.3 Transaction

T18_R_getChallengesOfOrganizer

La transaction va chercher toutes les informations des challenges liées à un organisateur.

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, Role , Challenge, Challenge-Organizer

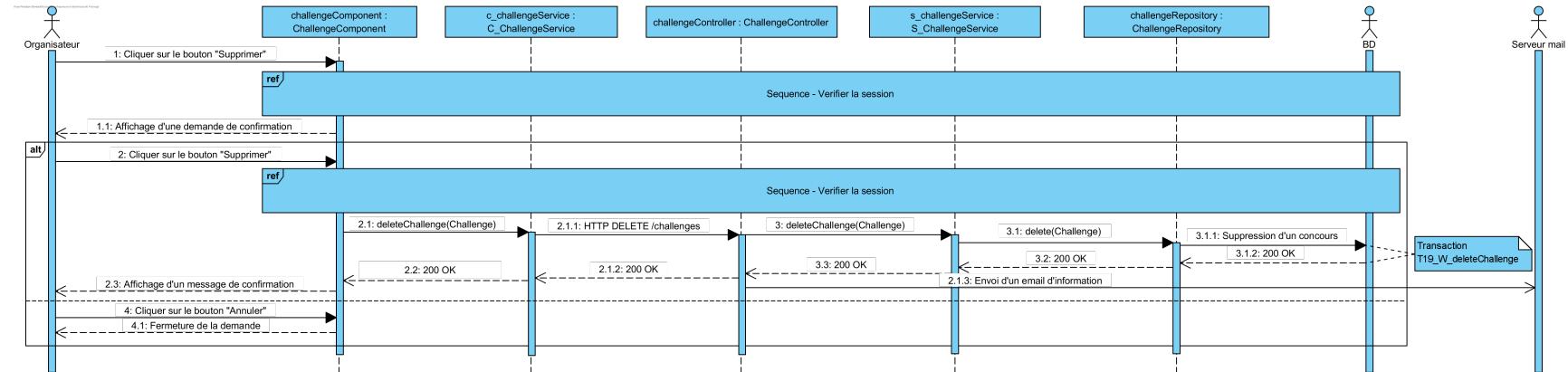
Fréquence : Rare

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 »

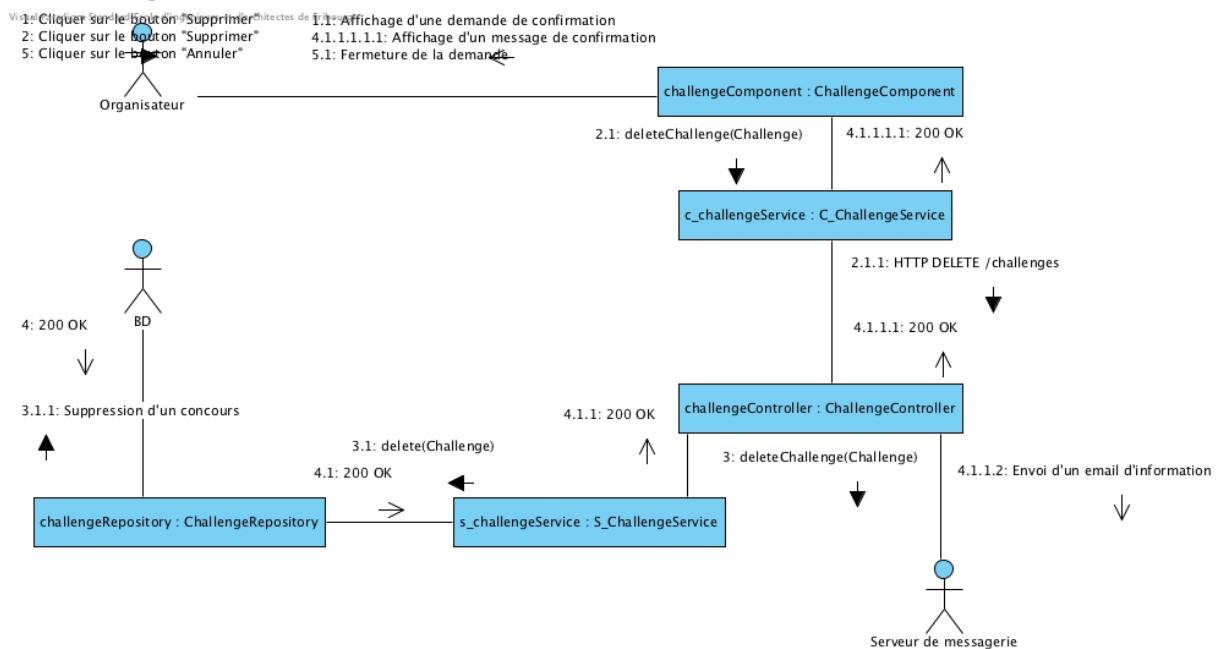
3.19 Cas 19 : Supprimer un concours

Un organisateur peut supprimer un concours n'ayant pas encore commencé.

3.19.1 Diagramme de séquence



3.19.2 Diagramme de communication



3.19.3 Transaction

T19_W_deleteChallenge

La transaction va supprimer un challenge dans la base de données

Type de la transaction : Lecture et Ecriture

Isolation : SERIALISABLE

La transaction va supprimer un challenge, cela inclus de supprimer toutes les équipes liées aux challenges. Il est important qu'une équipe liée à ce concours ne soit pas créé durant la transaction.

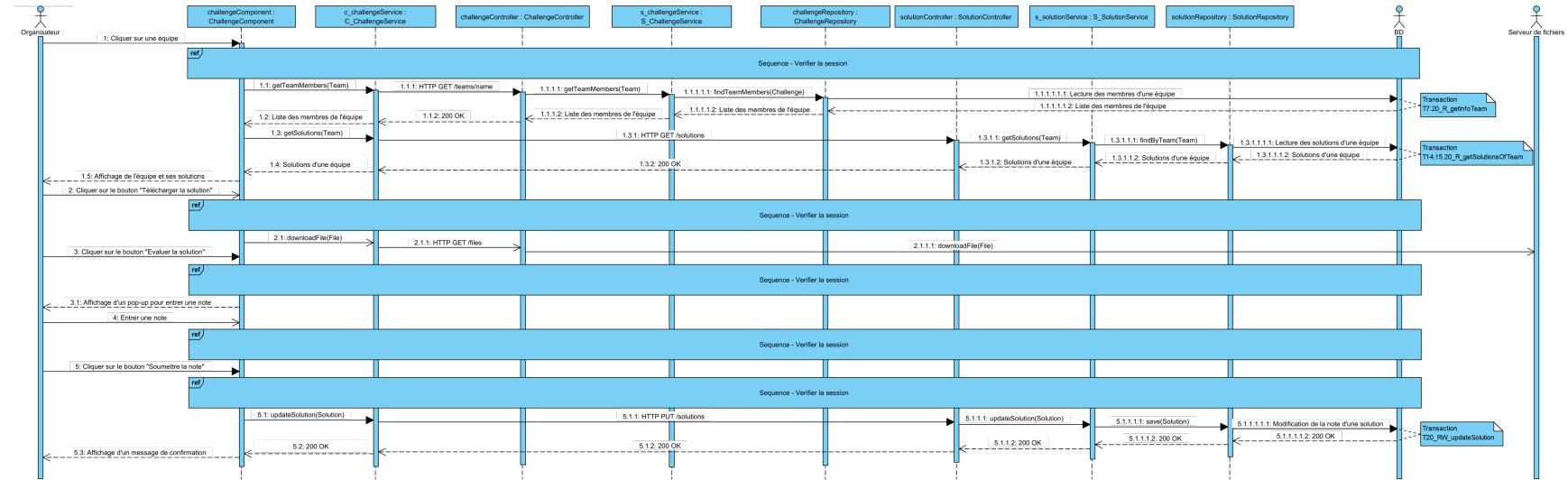
Tables touchées : Toutes les tables sauf solution

Fréquence : Rare

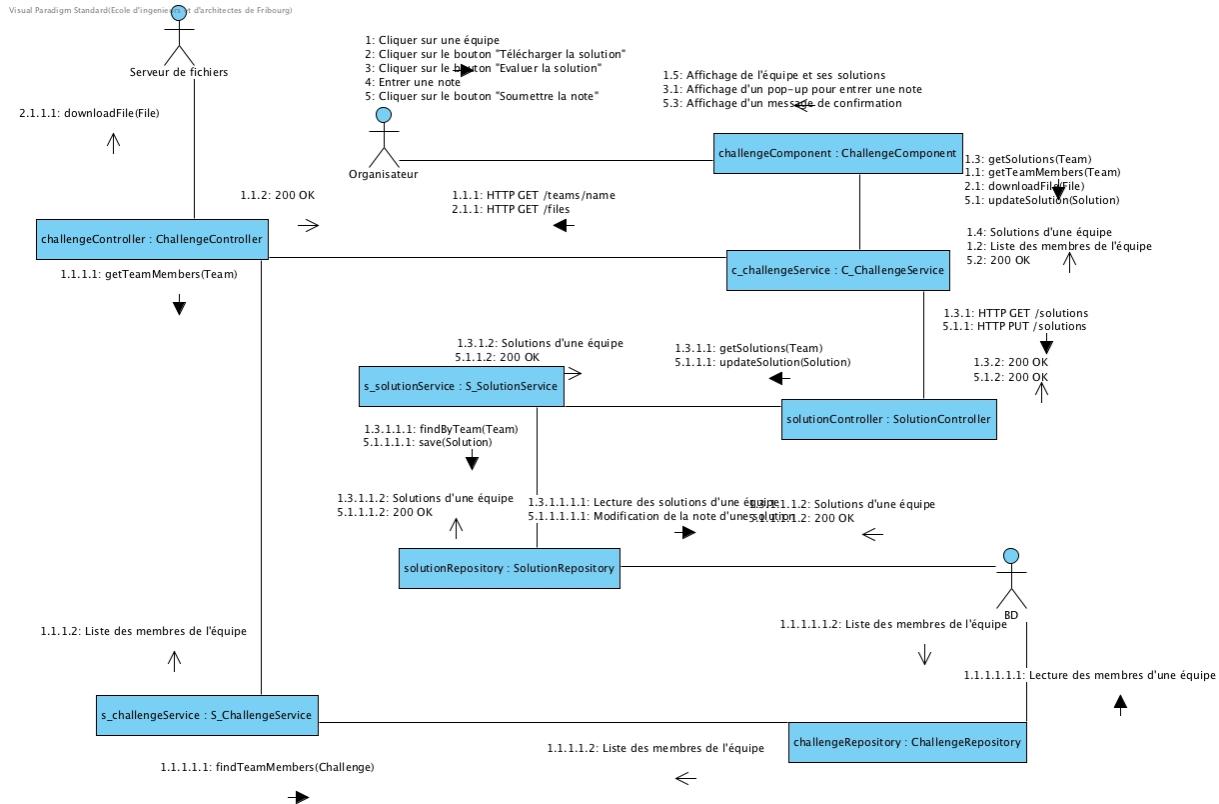
3.20 Cas 20 : Evaluer les solutions

Un organisateur évalue les solutions d'un concours en cours fournit par les équipes.

3.20.1 Diagramme de séquence



3.20.2 Diagramme de communication



3.20.3 Transaction

T7.20_R_getInfoTeam : Cette transaction est décrite dans le chapitre « 3.7.3 transactions » du cas d'utilisation « Consulter les équipes »

T14.15.20_R_getSolutionsOfTeam : Cette transaction est décrite dans le chapitre « 3.14.2 transactions » du cas d'utilisation « Consulter les équipes »

T20_RW_updateSolution

La transaction va mettre à jour la note d'une solution

Type de la transaction : Lecture et Ecriture

Isolation : REPEATABLE READ

Il est nécessaire que la solution ne soit pas être modifier ou supprimer, cependant d'autres solutions peuvent être ajouté dans la table.

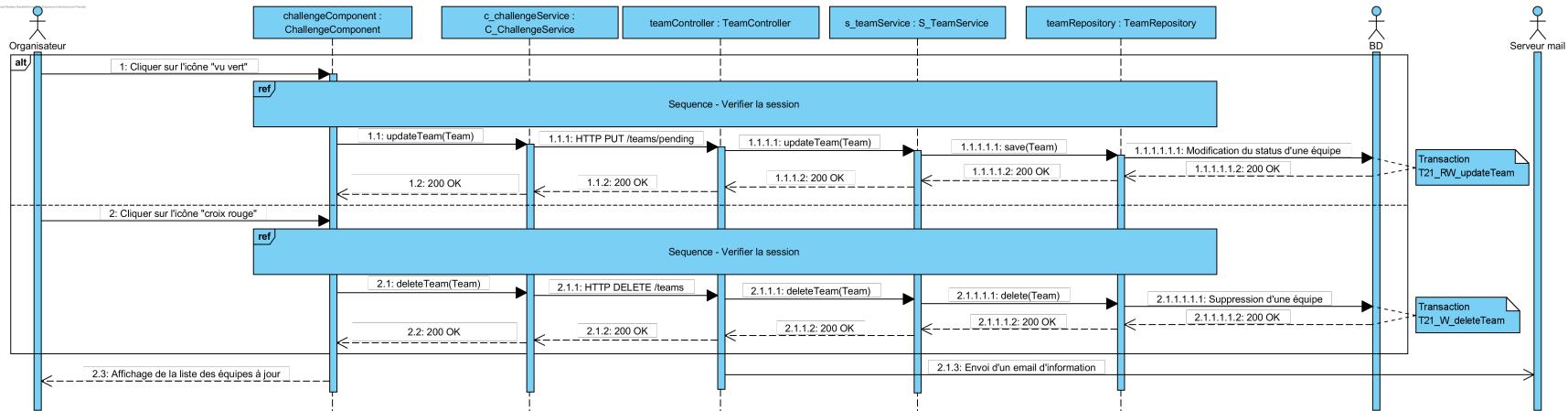
Tables touchées : Account, Role, Team, Solution Account-team, Account-organizer

Fréquence : Moyenne

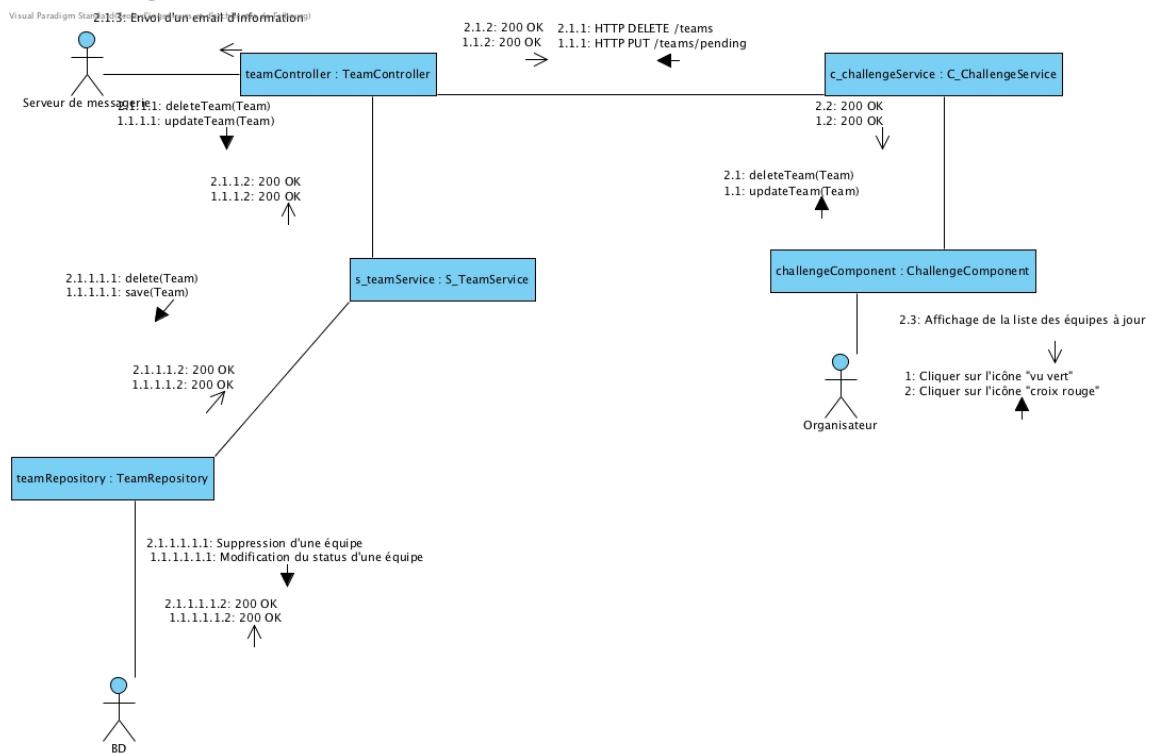
3.21 Cas 21 : Valider les inscriptions

Un organisateur peut valider les inscriptions d'équipes à son concours.

3.21.1 Diagramme de séquence



3.21.2 Diagramme de communication



3.21.3 Transaction

T21_RW_updateTeam

La transaction va modifier le statuts d'une équipe.

Type de la transaction : Lecture et Ecriture

Isolation : SERIALIZABLE

Une même équipe avec le même nom ne doit pas être ajouté ou supprimer lors de cette transaction.

Tables touchées : Account, Role, Team, Challenge, challenge-organizer

Fréquence : Moyenne

T21_W_deleteTeam

La transaction va supprimer une équipe dans un concours

Type de la transaction : Lecture et Ecriture

Isolation : SERIALIZABLE

Des membres ne doivent pas être ajoutés à l'équipes lors de cette transaction.

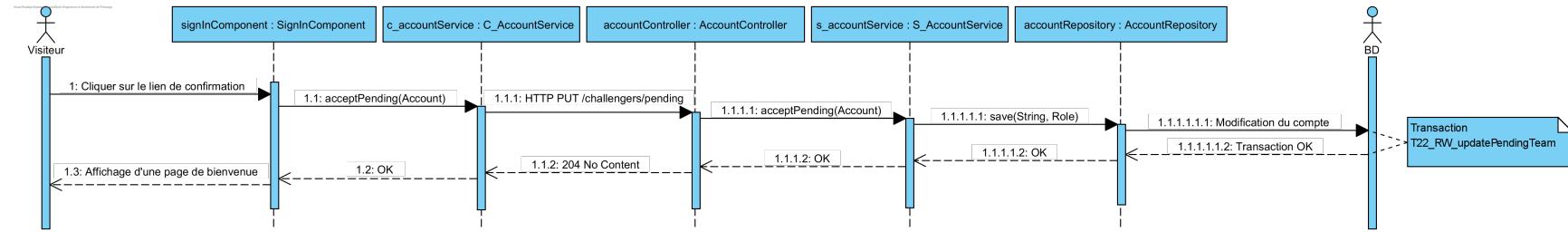
Tables touchées : Account, Role, Team, Challenge, account-team, challenge-organizer

Fréquence : Moyenne

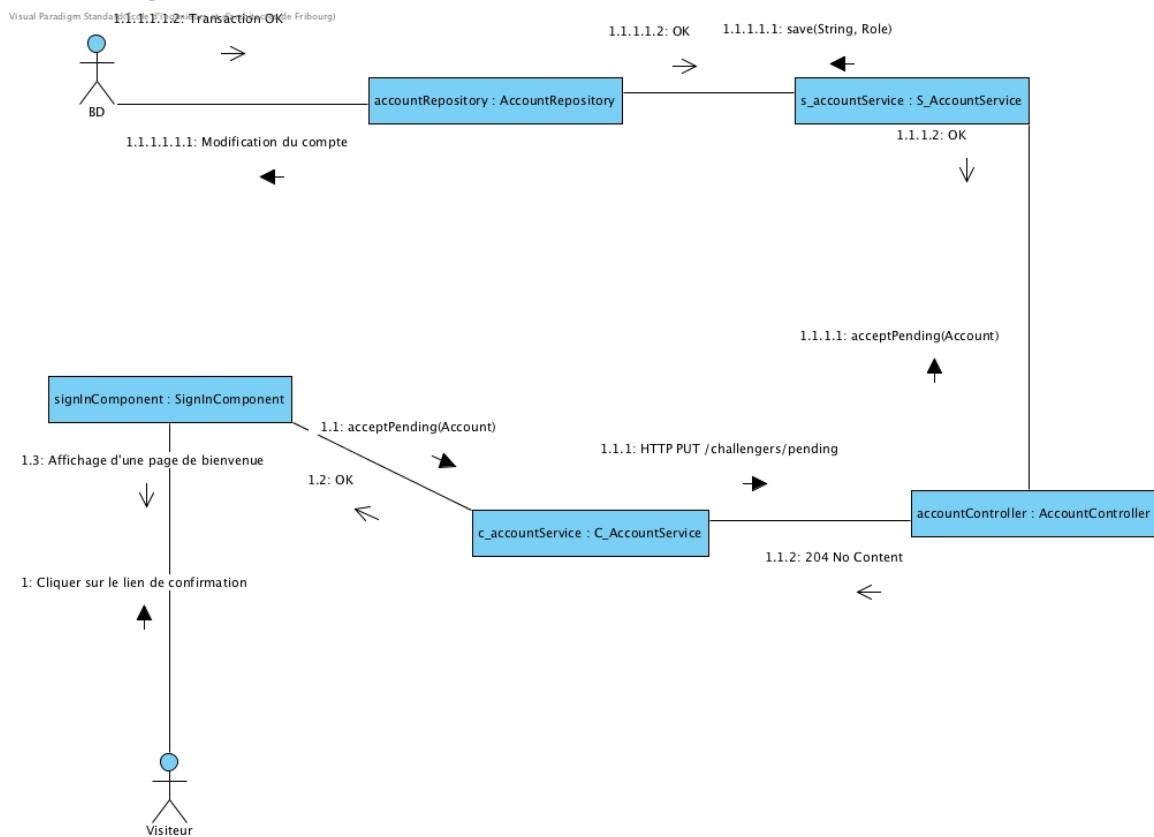
3.22 Cas 22 : Valider le compte

Un visiteur peut confirmer son adresse email.

3.22.1 Diagramme de séquence



3.22.2 Diagramme de communication



3.22.3 Transaction

T22_RW_updatePendingAccount :

La transaction va changer le rôle d'un compte souhaitant devenir utilisateur loggée.

Type de la transaction : Lecture et Ecriture

Isolation : SERIALIZABLE

Il ne faut pas qu'une ligne soit créée en même temps dans la table

Table(s) touchée(s) : Account, Role

Fréquence : Moyenne

Période(s) : -

4 Liste des objets et composants

Voici la liste des objets et composants de l'application :

No	Nom de l'objet ou du composant	Nome de la classe	No. Des Use Case
1	organizerComponent	OrganizerComponent	1
2	c_accountService	C_AccountService	1, 2, 4, 6, 22
3	accountController	AccountController	1, 2, 4, 6, 12, 17, 22
4	s_accountService	S_AccountService	1, 2, 4, 6, 12, 17, 22
5	accountRepository	AccountRepository	1, 2, 4, 6, 12, 17, 22
6	adminComponent	AdminComponent	2
7	challengeComponent	ChallengeComponent	3, 7, 8, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21
8	c_challengeService	C_ChallengeService	3, 7, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 21
9	challengeController	ChallengeController	3, 7, 8, 14, 15, 16, 17, 18, 19, 20
10	s_challengeService	S_ChallengeService	3, 7, 8, 14, 15, 16, 17, 18, 19, 20
11	challengeRepository	ChallengeRepository	3, 7, 8, 14, 15, 16, 17, 18, 19, 20
12	headerComponent	HeaderComponent	4, 5, 9
13	profileComponent	ProfileComponent	6
14	teamController	TeamController	7, 8, 10, 21
15	s_teamService	S_TeamService	7, 8, 10, 21
16	teamRepository	TeamRepository	7, 8, 10, 21
17	component	Component	13
18	springSessionRepositoryFilter	SessionRepositoryFilter	13
19	solutionController	SolutionController	14, 15, 20

20	s_solutionService	S_SolutionService	14, 15, 20
21	solutionRepository	SolutionRepository	14, 15, 20
22	signInComponent	SignInComponent	22

5 Diagramme de classes

Diagramme de classes du backend :

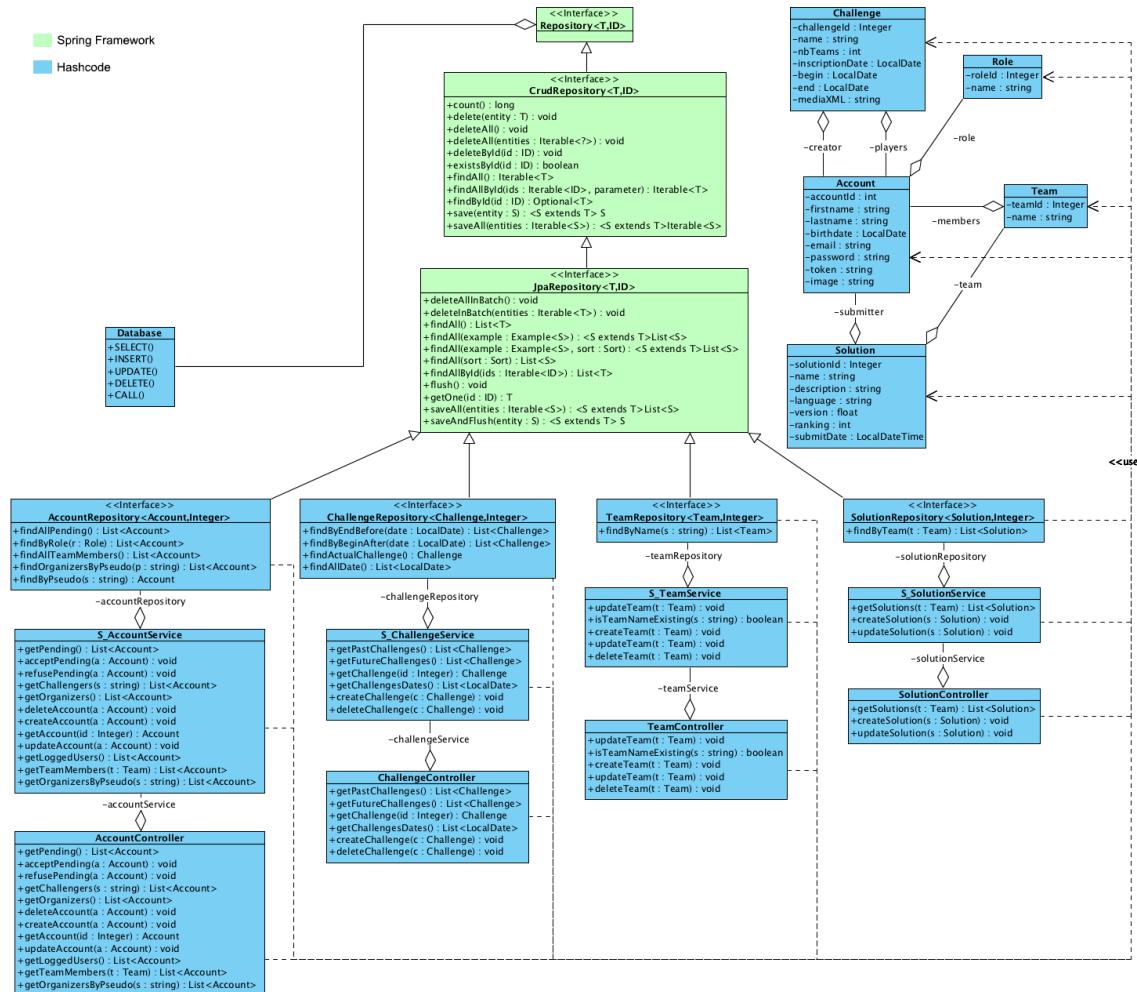
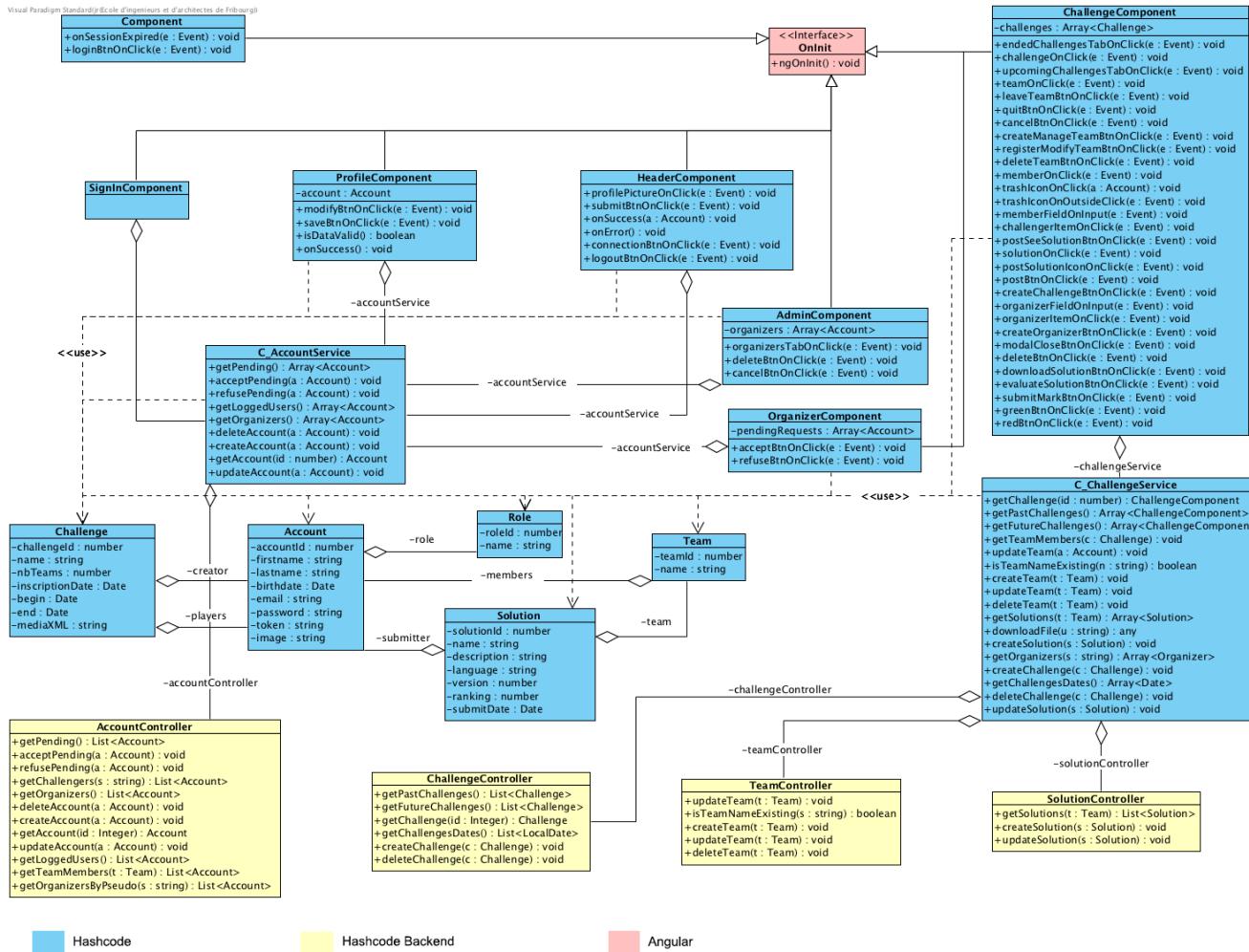


Diagramme de classes du frontend :



6 Concurrence

6.1 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).

Nom Transactions	Account	Role	Team	Challenge	Solution	Data	account-team	challenge-organizer	Niveau Isolation
T1_R_getAllPendingOrganizers	R	R							1
T1_RW_updatePendingOrganizers	R/W	R							3
T2_R_getLoggedUsers	R	R							1
T2.17_R_getOrganizers	R	R							1
T2_RW_deleteAccount	R/W	R							3
T3.7.8.14.15.16.18_R_getChallenge	R	R	R	R		R	R	R	1
T3.7.8.14.16_R_getPastChallenge				R					1
T3.7.8.16_R_getFutureChallenge				R					1
T4_RW_createAccount	W	R							3
T6_R_getInfoAccount		R							1
T6_RW_updateAccount		R/W							3
T7.20_R_getInfoTeam	R		R				R		1
T7_RW_deleteMemberTeam	R		R				R/W		2
T8_R_getTeamsOfChallenge			R	R				R	1

T8_W_createTeam	R		R/W	R			R/W		3
T8_RW_updateTeam	R		R/W	R			R/W		3
T10_W_deleteTeam	R		R/W	R			R/W		3
T12_R_getChallengers	R	R	R	R			R		1
T14.15.20_R_getSolutionsOfTeam	R		R	R	R		R		1
T14_R_getSolutionOfTeam	R		R	R	R		R		1
T15_W_createSolution	R		R	R	R/W		R		3
T17_R_getFutureChallengesDates				R					1
T17_W_createChallenge	R	R		R/W		R/W		R/W	3
T18_R_getChallengesOfOrganizer	R	R		R				R	1
T19_W_deleteChallenge	R	R	R/W	R/W		R/W	R/W	R/W	3
T20_RW_updateSolution	R	R	R		R/W		R	R	2
T21_RW_updateTeam	R	R	R/W	R				R	3
T21_W_deleteTeam	R	R	R/W	R			R/W	R	3
T22_RW_updatePendingAccount	R/W	R							3

6.2 Présentation d'un scénario concurrentiel

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

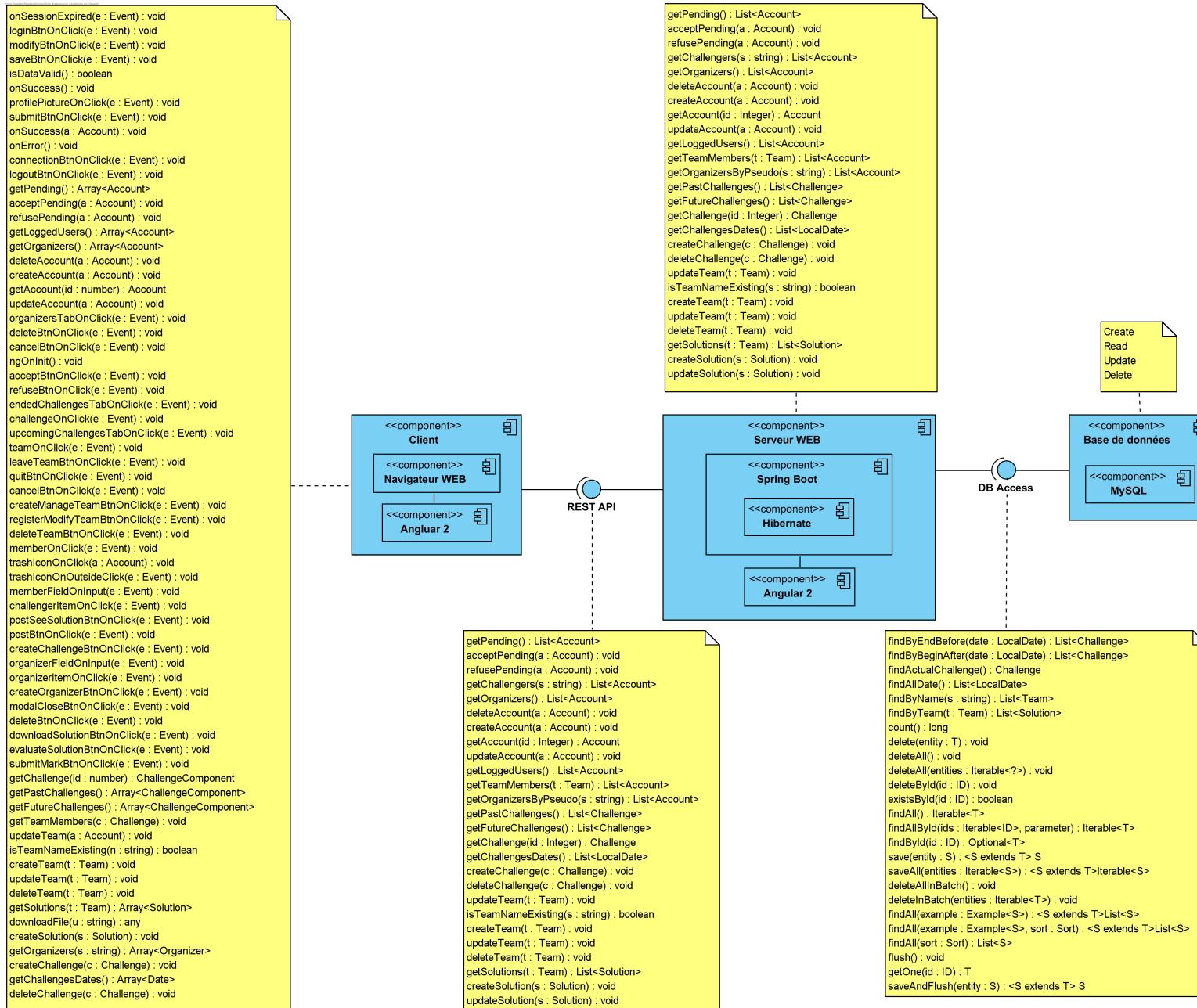
1. Les noms des transactions en jeu
2. Les problèmes qui risquent d'arriver
3. Expliquez alors clairement comment s'applique la philosophie de la gestion de concurrence que vous avez choisie sur ce cas et avec laquelle vous évitez les problèmes.

Le scénario utilise les transactions du cas d'utilisation 1 : « **T1_R_getAllPendingOrganizers** » et « **T1_RW_updatePendingOrganizers** ».

Dans ce cas, imaginons que deux administrateurs aient chargé la liste des personnes souhaitant devenir organisateur et que le premier ait refusé la demande et que l'utilisateur disparait de sa liste. Cependant la liste du deuxième administrateur ne sera pas mise à jour et si celui-ci souhaite valider la demande. Dans ce cas, la transaction de modification a prévu de comparer la version de la liste envoyée avec celle actuelle sur la base de données. Si il s'avère que les données lues ont été modifiées entre temps, un message d'erreur est affiché à l'utilisateur. La transaction étant SERIALIZABLE, il est impossible qu'il y ait des modifications durant la transaction.

7 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.



8 Signatures

Fribourg, le 20 avril 2018

Joé Butty

Nicolas Fuchs

Jonathan Rial