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 3

2 Présentation des composants utilisés 3

3 Diagrammes de séquence et de communication 3

3.1 Cas 1 : Nom du cas d'utilisation 3

3.1.1 Diagramme(s) de séquence 3

3.1.2 Diagramme(s) de communication 4

3.1.3 Transaction(s) 4

3.2 Cas 2: Nom du cas d'utilisation 6

3.2.1 Diagramme(s) de séquence 6

3.2.2 Diagramme(s) de communication 6

3.2.3 Transaction(s) 6

4 Liste des objets et composants 6

5 Diagramme de classes 6

6 Concurrence 7

6.1 Récapitulatif sur les transactions 7

6.2 Présentation d’un scénario concurrentiel 8

7 Diagramme de composants 8

8 Signatures 9

# Changements par rapport à la phase de création

Changement dans le cas d’utilisation qui modifie le compte (Rajout d’un bouton éditable).

# Présentation des composants utilisés

Pour chacun des composants que vous pensez utiliser, vous devez en faire :

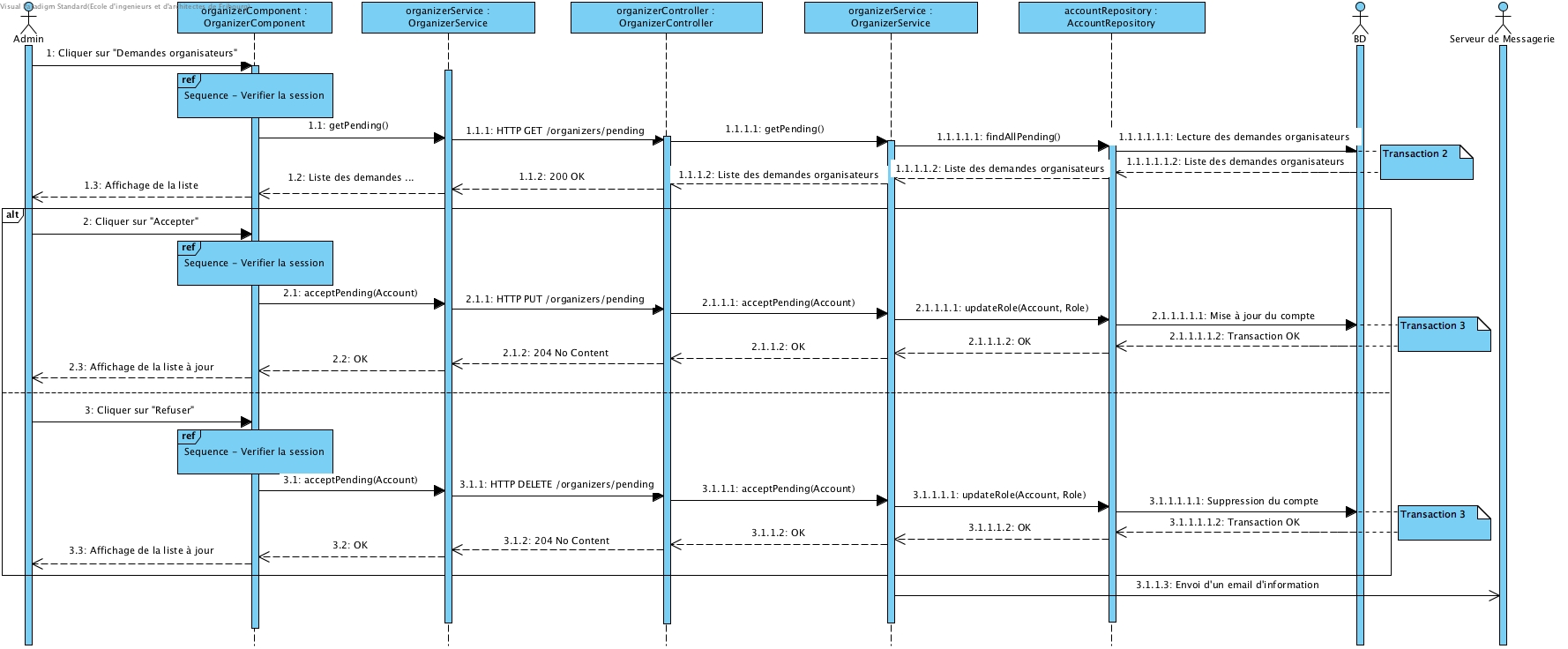
* Une présentation globale (enlever la virgule)
* Son utilité par rapport à votre projet
* Une présentation détaillée des contraintes que cela vous impose :
  + Au niveau de la modélisation en générale de votre application (à rajouter)
  + Au niveau de la modélisation de votre base de donnéesec
  + Au niveau des accès à la base
  + Au niveau du transactionnel

# Diagrammes de séquence et de communication

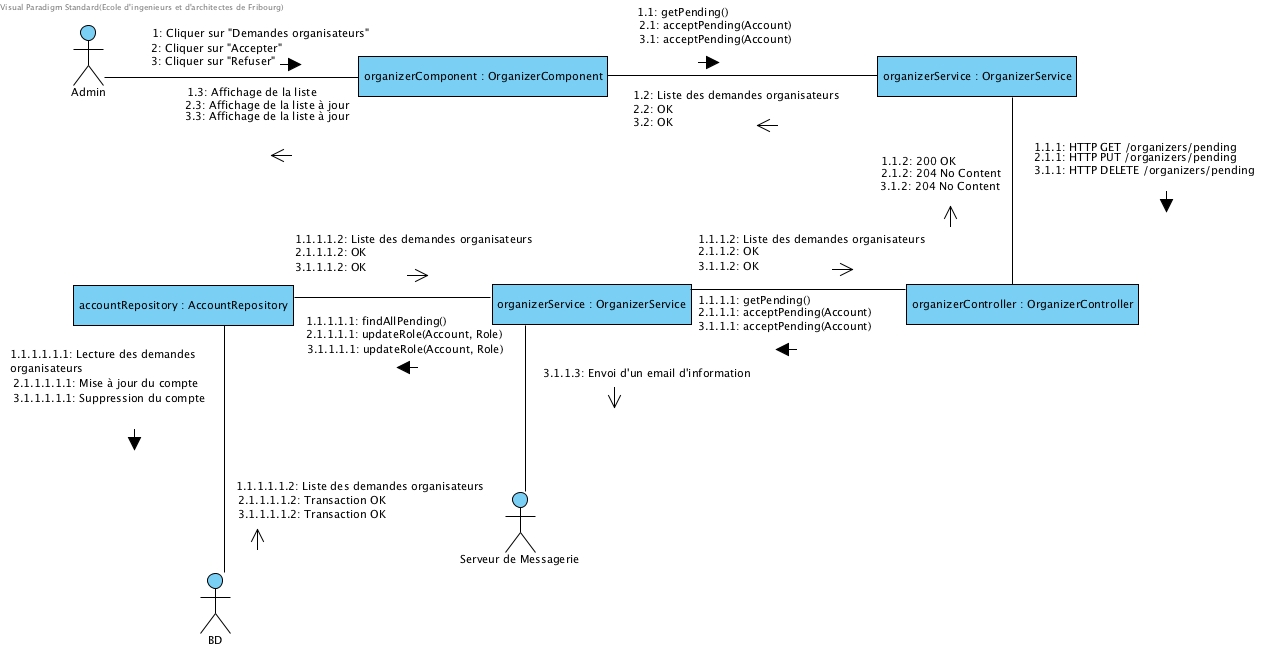
## Cas 1 : Définir les privilèges

Un administrateur peut accepter la demande d’une personne qui souhaite devenir organisateur.

### Diagramme(s) de séquence



### Diagramme(s) 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 chercher la liste des organisateurs. Pour éviter les lectures sales et récupérer des organisateurs qui ont déjà été modifiés, il est nécessaire d’imposer un verrous d’écriture lors de la transaction.

**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.

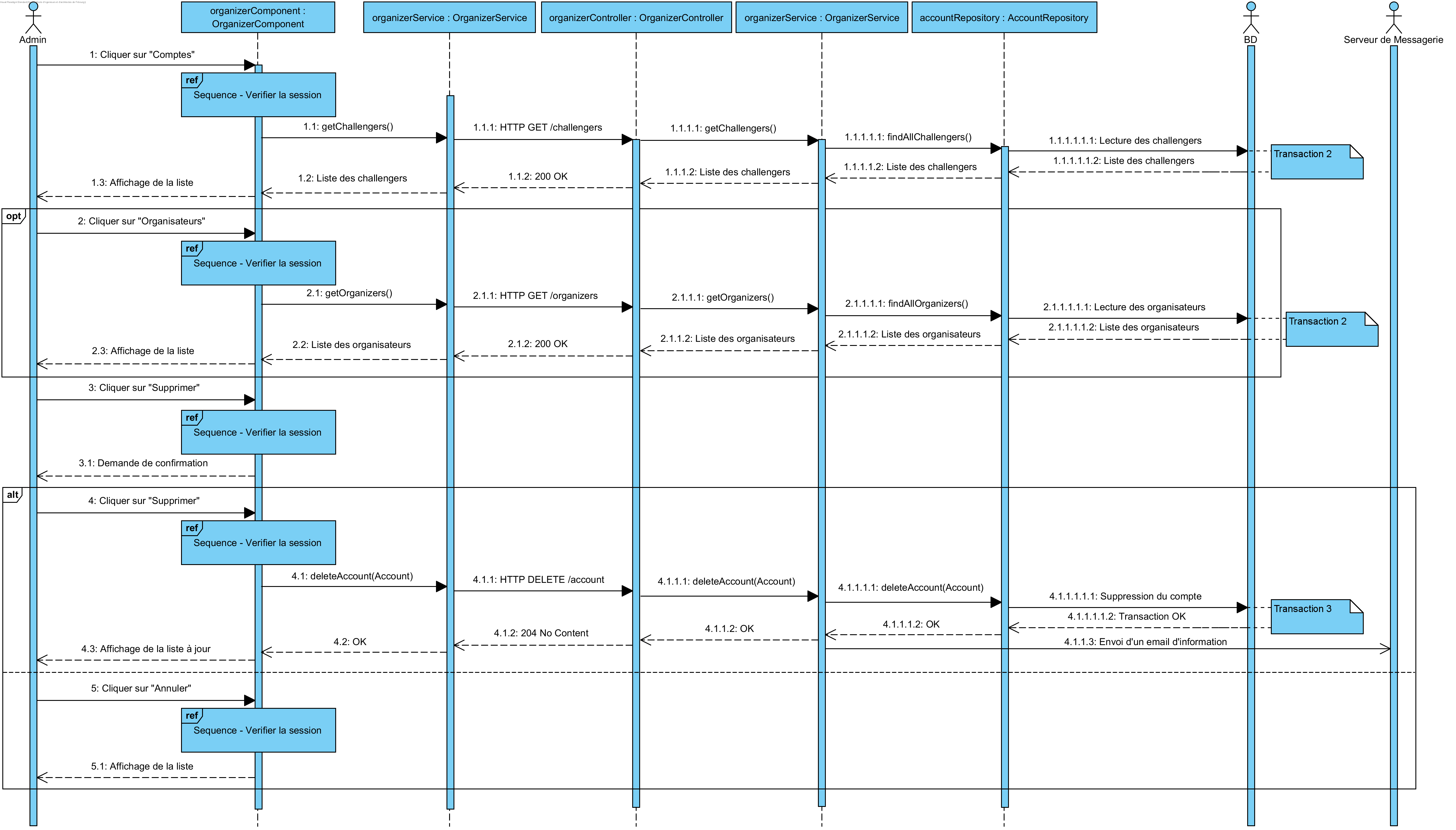
**Table(s) touchée(s) :** Account, Role

**Fréquence :** Rare

**Période(s) :** -

## Cas 2 : Supprimer un compte

### Diagramme(s) de séquence



### Diagramme(s) 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 chercher la liste des personnes loggées. Pour éviter les lectures sales et récupérer des personnes loggées qui ont déjà été supprimées, il est nécessaire d’imposer un verrous d’écriture lors de la transaction.

**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 chercher la liste des organisateurs. Pour éviter les lectures sales et récupérer des organisateurs qui ont déjà été supprimées, il est nécessaire d’imposer un verrous d’écriture lors de la transaction.

**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

### Diagramme(s) de séquence

### Diagramme(s) de communication

### Transactions

**T3\_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 les informations d’un challenge, afin que les informations soient à jours, il est nécessaire de mettre un verrou sur la transaction.

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

**Fréquence :** Elevé

**Période :** -

**T3\_R\_getPastChallenges**

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

**Type de la transaction :** Lecture

**Isolation :** READ UNCOMMITED

Les anciens challenges ne seront pas modifiés, il n’est donc pas nécessaire de mettre des verrous lorsqu’on récupère les informations

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

**Fréquence :** Elevé

**Période :** -

**T3\_R\_getFuturChallenges**

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

**Type de la transaction :** Lecture

**Isolation :** SERIALIZABLE

Dans le cas où deux organisateurs (O1, O2) sont connectés, les deux ont chargé la liste des prochains concours, si O1 a supprimé un futur concours après que O2 ait récupéré la liste, dans la vue de O2, il y aura un concours n’existant plus dans la base de données.

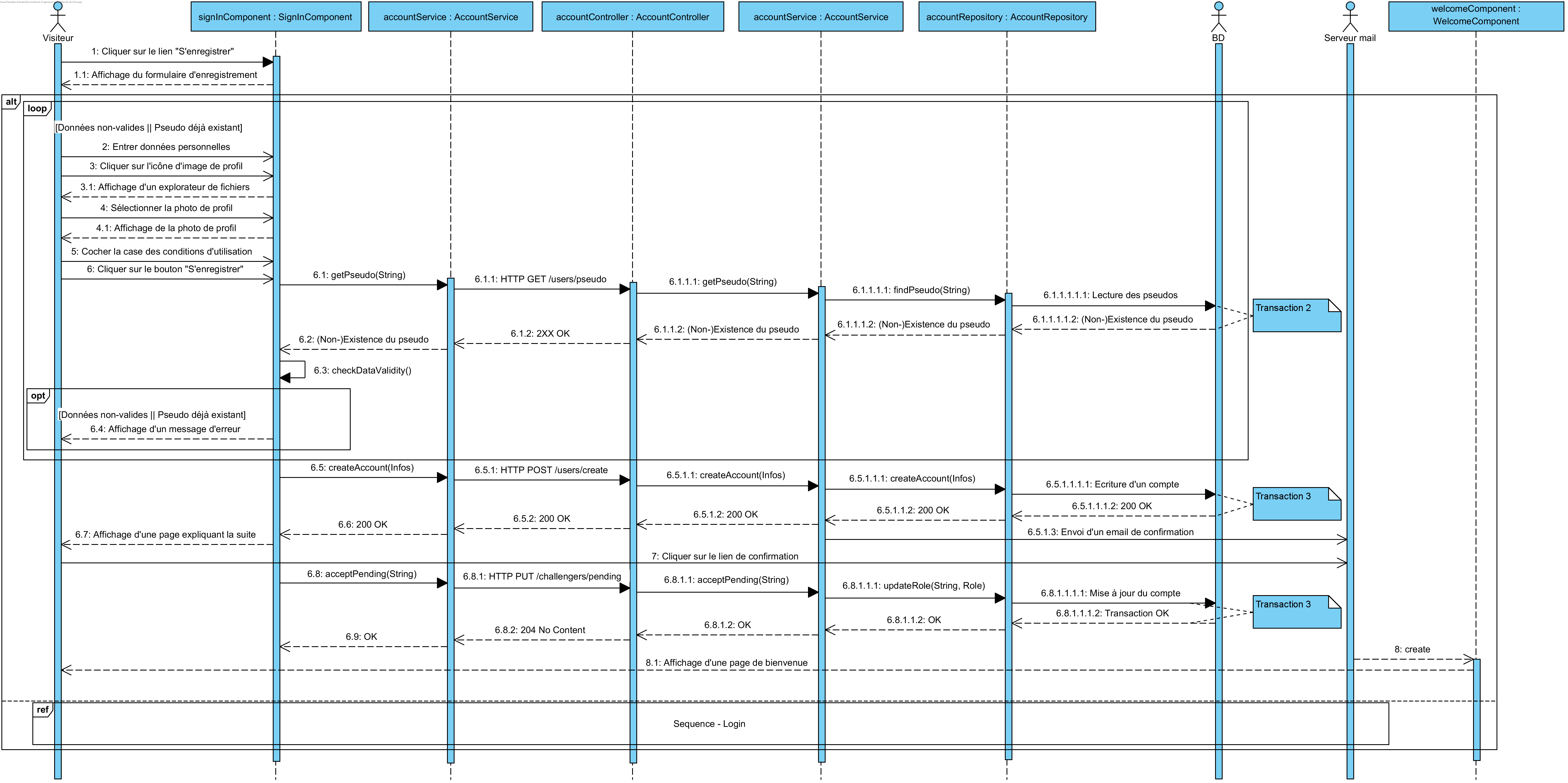
**Tables touchées :** Challenge, Account-Challenge

**Fréquence :** Elevé

**Période :** -

## Cas 4 : S'enregistrer sur la plateforme

### Diagramme(s) de séquence



### Diagramme(s) de communication

### Transactions

**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.

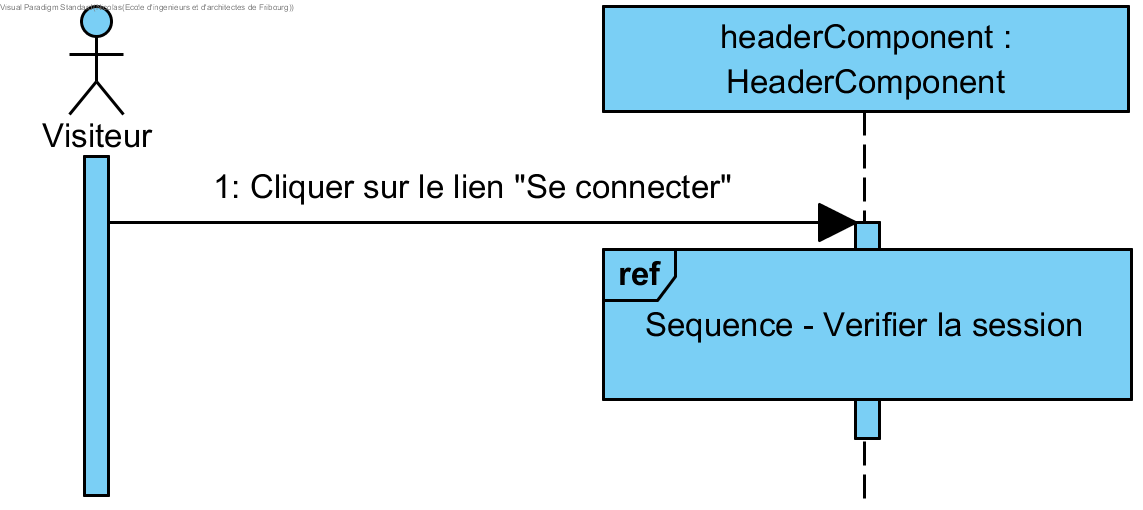
**Tables touchées :** Account

**Fréquence :** Moyenne

**Période :** -

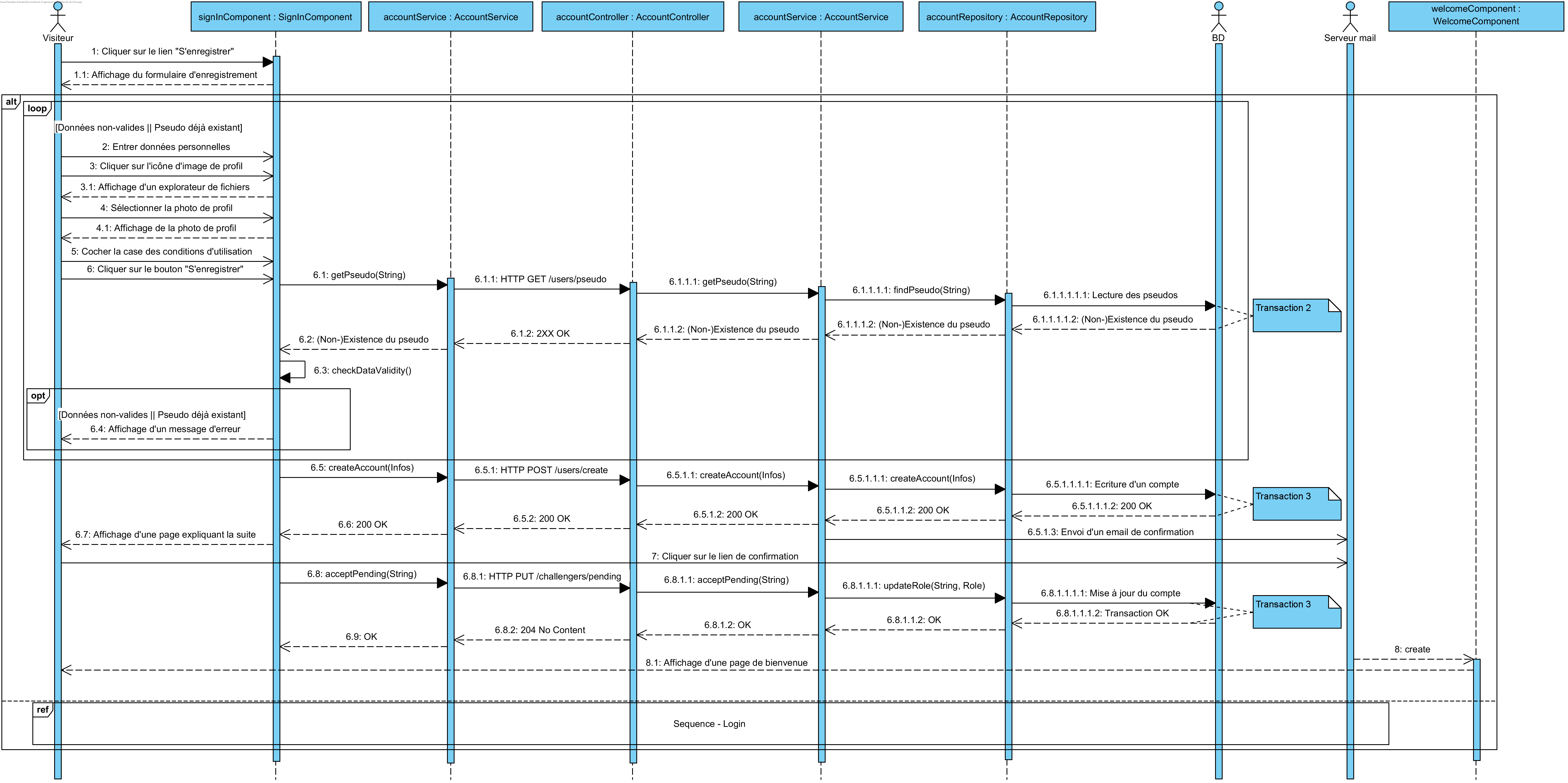
## Cas 5 : Login

### Diagramme(s) de séquence



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

### Diagramme(s) de séquence



### Diagramme(s) de communication

### Transactions

**T6\_R\_getInfoAccount**

La transaction va chercher toutes les informations d’un account.

**Type de la transaction :** Lecture

**Isolation :** READ UNCOMMITTED

Il ne peut pas y avoir de lecture sale car les données ne sont modifiées uniquement dans ce cas d’utilisation.

**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 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.

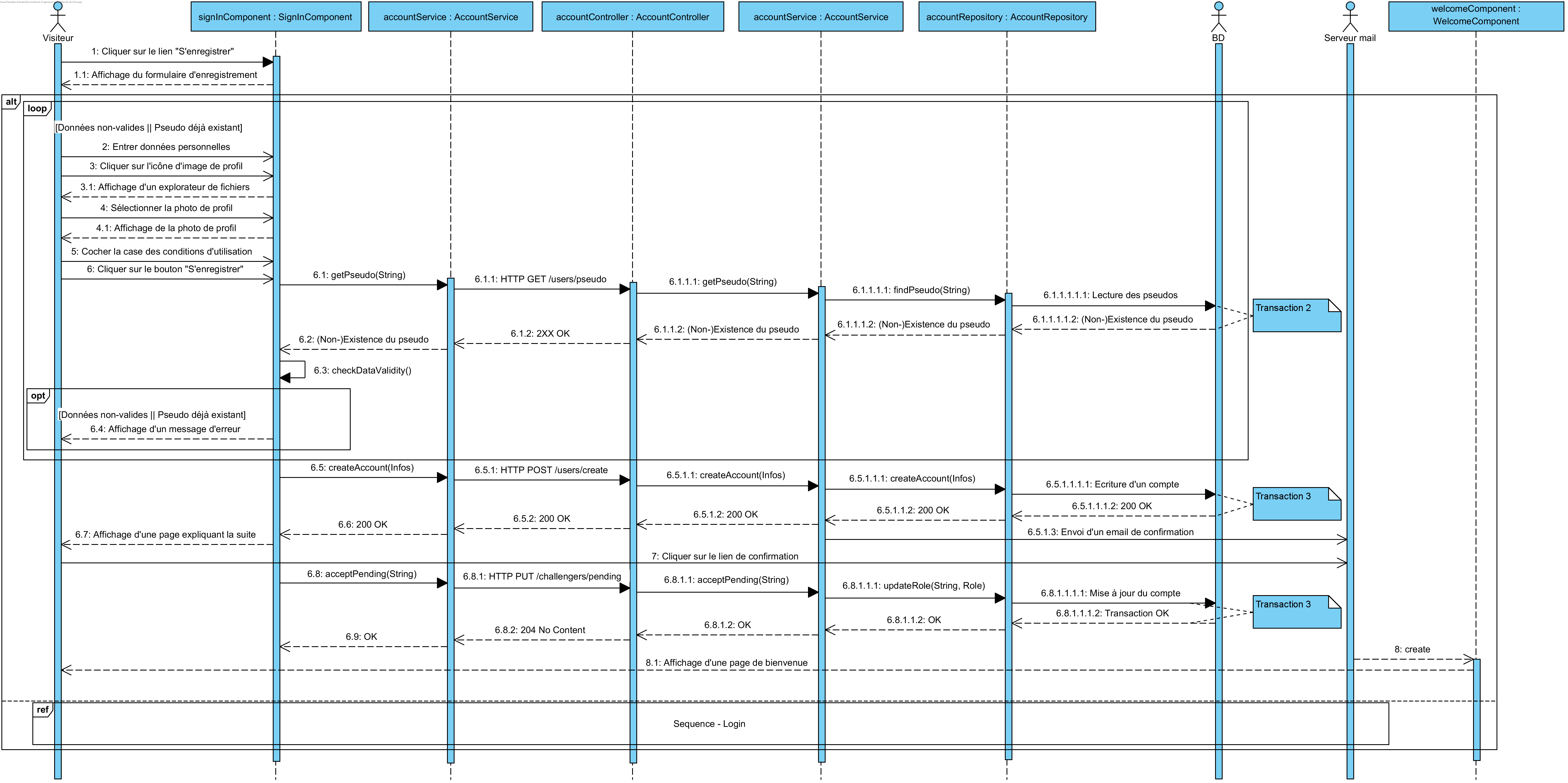
**Tables touchées :** Account

**Fréquence :** Moyenne

**Période :** -

## Cas 7: Consulter les équipes

### Diagramme(s) de séquence



### Diagramme(s) de communication

### Transactions

**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

Les informations des équipes doivent être valides, il faut donc éviter une lecture sale.

**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 une lecture sale au milieu de la transaction en cas d’annulation de l’opération.

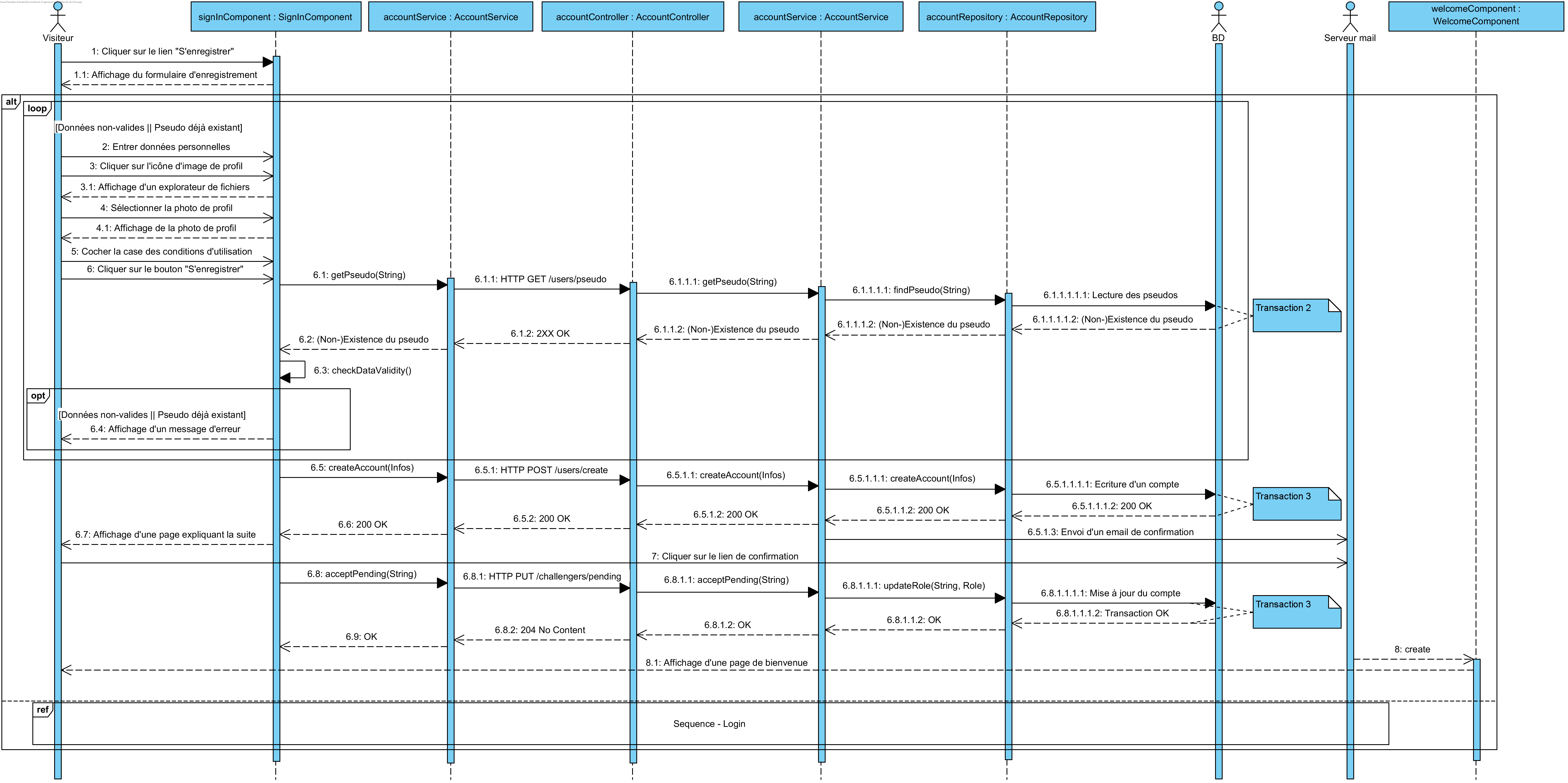
**Tables touchées :** Team, Account,, Account-Team

**Fréquence :** Faible

**Période :** -

## Cas 8 : Gérer une équipe

### Diagramme(s) de séquence



### Diagramme(s) de communication

### Transactions

**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.

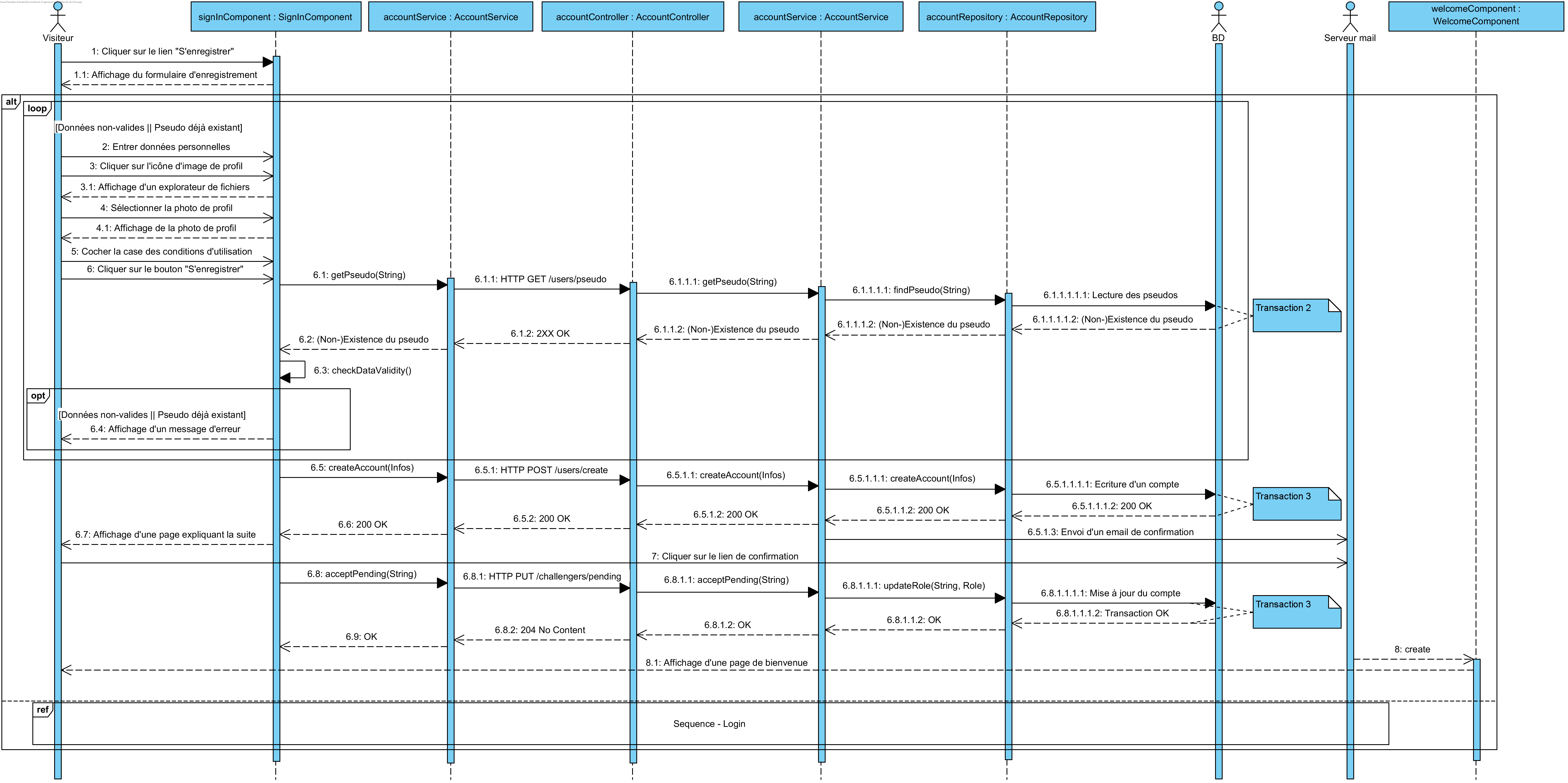
**Tables touchées :** Account

**Fréquence :** Moyenne

**Période :** -

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

### Diagramme(s) de séquence



### Diagramme(s) de communication

### Transactions

**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.

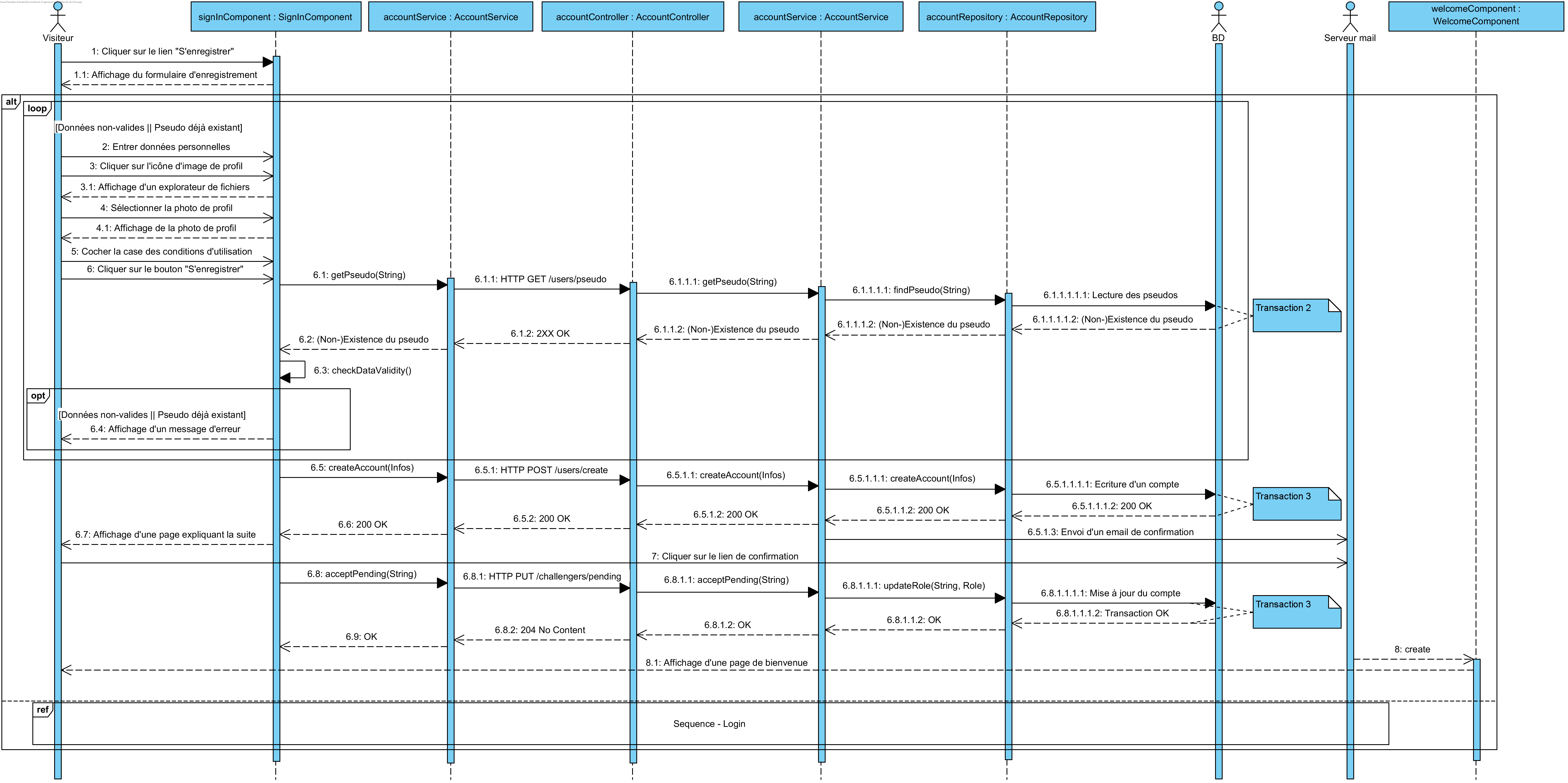
**Tables touchées :** Account

**Fréquence :** Moyenne

**Période :** -

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

### Diagramme(s) de séquence



### Diagramme(s) de communication

### Transactions

**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.

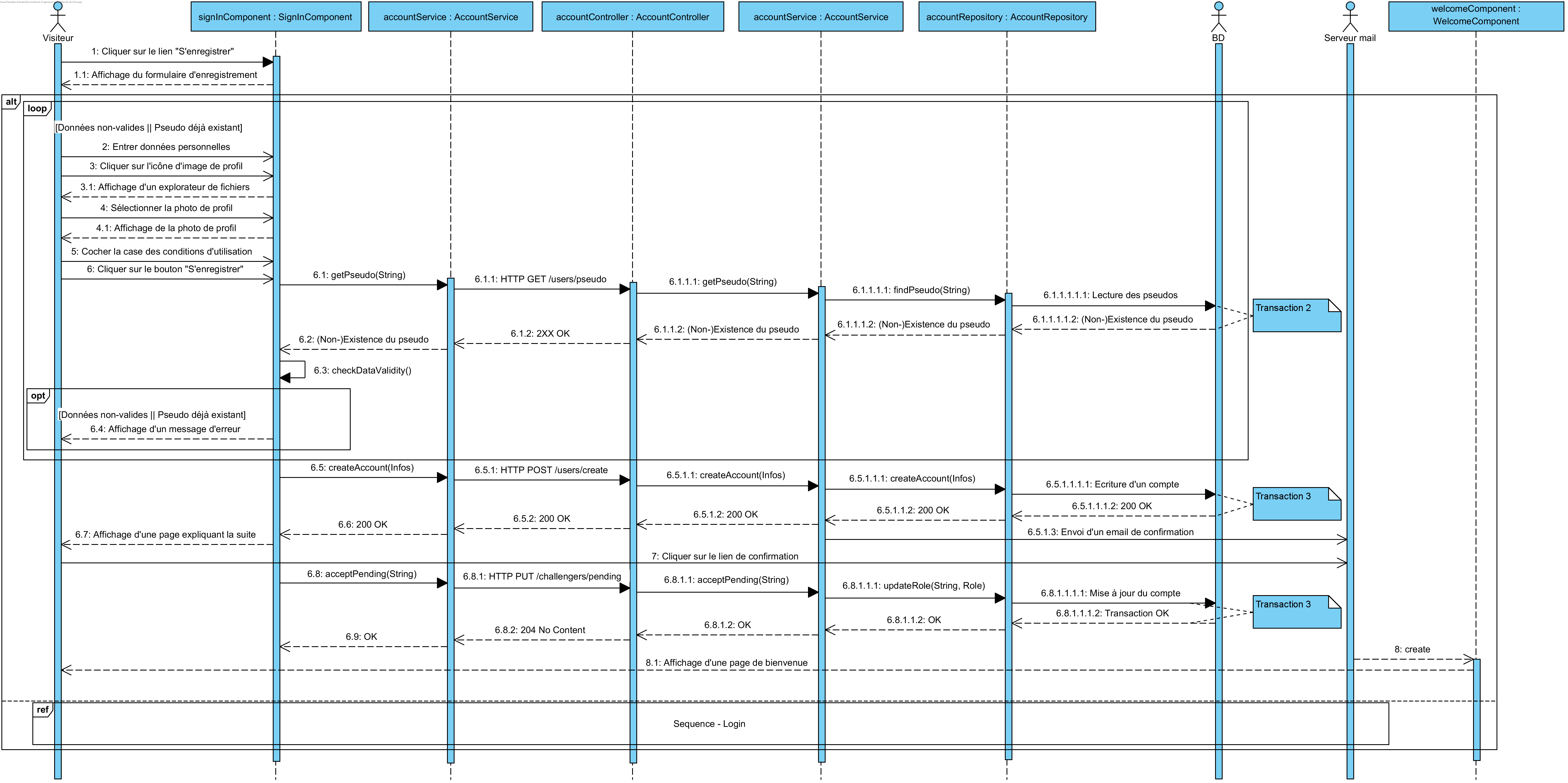
**Tables touchées :** Account

**Fréquence :** Moyenne

**Période :** -

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

### Diagramme(s) de séquence



### Diagramme(s) de communication

### Transactions

**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

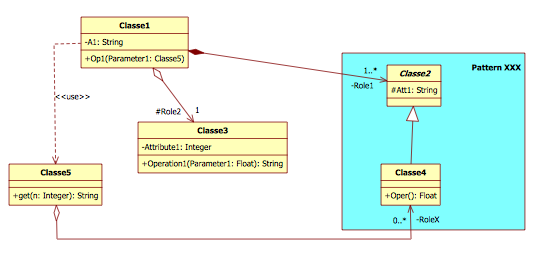
Il est nécessaire de verrouiller la table car dans le cas où deux visiteurs (V1, V2) souhaitent s’inscrire en même temps avec le même pseudo, il ne faut pas que V1 et V2 vérifie l’existence du pseudo en même temps et écrive en même temps dans la base de donnée.

**Tables touchées :** Account

**Fréquence :** Moyenne

**Période :** -

# 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