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

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.

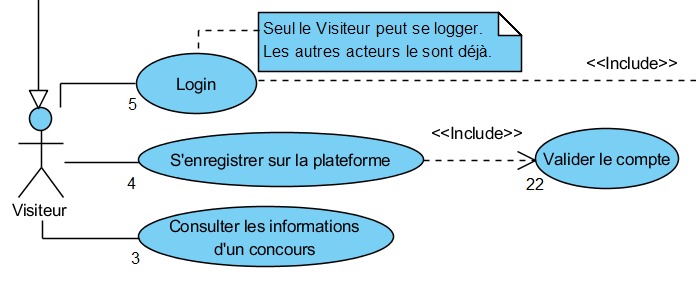
### Contraintes relationnelles

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.

### 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

# 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 données
  + Au niveau des accès à la base
  + Au niveau du transactionnel

# Diagrammes de séquence et de communication

## Cas 1 : Définir les privilèges

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 2 : Supprimer un compte

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

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

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 4 : S’enregistrer sur la plateforme

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 5 : Login

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

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

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 7 : Consulter les équipes

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 8 : Gérer une équipe

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 9 : Logout

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 10 : Supprimer les équipes

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## 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

## Cas 13 : Vérifier la session

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 14 : Consulter les solutions déposées

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 15 : Déposer une solution

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 16 : Consulter les concours

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 17 : Créer un concours

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 18 : Gérer un concours

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 19 : Supprimer un concours

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 20 : Evaluer les solutions

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 21 : Valider les inscriptions

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

### Transaction

## Cas 22 : Valider le compte

Résumé du cas.

### Diagramme de séquence

### Diagramme de communication

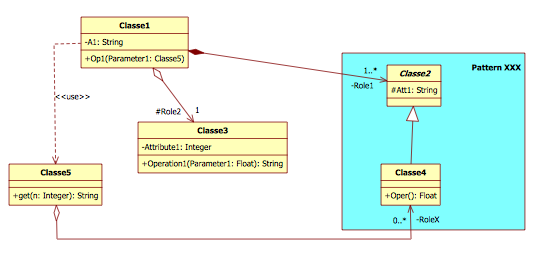
### Transaction

# 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