133 – Réaliser des applications Web en Session-Handling

Documentation de projet

Version 1 du 31 mars 2025

Crée le 24.03.25

|  |  |
| --- | --- |
|  | Module du 17.03.25 au xx.04.25 |

Table des matières

[1 Introduction et contexte du projet 4](#_Toc194309794)

[2 Analyse à faire complètement avec EA 5](#_Toc194309795)

[2.1 Use case client et use case Rest 5](#_Toc194309796)

[2.1.1 Use case client 1 5](#_Toc194309797)

[2.1.2 Use case client 2 6](#_Toc194309798)

[2.1.3 Use case Rest 1 7](#_Toc194309799)

[2.1.4 Use case Rest 2 8](#_Toc194309800)

[2.1.5 Use case GW 9](#_Toc194309801)

[2.2 Activity Diagram d'un cas complet 10](#_Toc194309802)

[2.2.1 API Rest 1 (ouvrir compétition) 10](#_Toc194309803)

[2.2.1 API Rest 2 (login) 10](#_Toc194309804)

[2.3 Maquettes du projet 11](#_Toc194309805)

[2.3.1 Login 11](#_Toc194309806)

[2.3.2 Admin 11](#_Toc194309807)

[2.3.3 Utilisateur 12](#_Toc194309808)

[2.4 Séquence System global entre les applications 13](#_Toc194309809)

[2.4.1 API Rest 1 (ouvrir compétition) 13](#_Toc194309810)

[2.4.2 API Rest 2 (login) 14](#_Toc194309811)

[3 Conception 16](#_Toc194309812)

[3.1 Diagramme de classes de chaque application 16](#_Toc194309813)

[3.1.1 Application 1 16](#_Toc194309814)

[3.1.2 Application 2 16](#_Toc194309815)

[3.1.3 API Gateway 18](#_Toc194309816)

[3.1.4 API Rest 1 19](#_Toc194309817)

[3.1.5 API Rest 2 20](#_Toc194309818)

[3.2 Navigation Diagram complète des applications 21](#_Toc194309819)

[3.2.1 Application 1 21](#_Toc194309820)

[3.2.2 Application 2 22](#_Toc194309821)

[4 Bases de données 23](#_Toc194309822)

[4.1 Modèles ER 23](#_Toc194309823)

[4.2 Modèles Workbench MySQL 24](#_Toc194309824)

[4.3 DB 1 24](#_Toc194309825)

[4.4 DB 2 24](#_Toc194309826)

[5 Implémentation des applications <Le client Ap1> et <Le client Ap2> 26](#_Toc194309827)

[5.1 Une descente de code client 26](#_Toc194309828)

[6 Implémentation de l'application <API Gateway> 27](#_Toc194309829)

[6.1 Une descente de code APIGateway 27](#_Toc194309830)

[7 Implémentation des applications <API élève1> et <API élève2> 28](#_Toc194309831)

[7.1 Une descente de code de l'API REST 28](#_Toc194309832)

[8 Hébergement 29](#_Toc194309833)

[9 Installation du projet complet avec les 5 applications 30](#_Toc194309834)

[10 Outils, langages (versions, définitions, installations) 31](#_Toc194309835)

[11 Tests de fonctionnement du projet 32](#_Toc194309836)

[12 Auto-évaluations et conclusions 33](#_Toc194309837)

[12.1 Auto-évaluation et conclusion de client 1 33](#_Toc194309838)

[12.2 Auto-évaluation et conclusion de client 2 33](#_Toc194309839)

# Introduction et contexte du projet

# Analyse à faire complètement avec EA

## Use case client et use case Rest

### Use case client 1

Voici le diagramme use case du client 1

Une image contenant texte, capture d’écran, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

|  |  |
| --- | --- |
| Acteurs | Description |
| Visiteur | **Login :** Le visiteur peut se connecter à un compte administrateur. |
| Admin | **getCompetitions :** Il pourra récupérer toute la liste des compétitions |
| **Ouvrir competition :** Il pourra ouvrir une nouvelle compétition |
| **Supprimer compétition :** Il pourra supprimer une compétition |
| **Désinscrire un participant :** Il pourra désinscrire un participant d’une compétition |
| **Changer état compétition : il pourra modifier l’état d’une compétition (inscriptions, votes, terminé)** |
| **Logout :** Il pourra se déconnecter |

### Use case client 2

Voici le diagramme de use case du client 2 :

Une image contenant texte, capture d’écran, diagramme, cercle

Le contenu généré par l’IA peut être incorrect.

Voici les actions que peuvent faire un visiteur et un utilisateur :

|  |  |
| --- | --- |
| Acteurs | Actions |
| Visiteur | Login : le visiteur peut se connecter à son compte s’il en possède déjà un. |
| S’enregistrer : le visiteur peut s’enregistrer pour créer un nouveau compte. |
| Utilisateur | Se déconnecter : l’utilisateur peut se déconnecter de son compte. |
| Get compétitions : l’utilisateur obtient la liste des compétitions en cours dès sa connexion/inscription. |
| Participer à une compétition : l’utilisateur peut choisir une ou plusieurs compétitions dans laquelle il veut participer. Il n’est pas obligé de participer à une compétition. |
| Voter : l’utilisateur peut voter pour un autre participant pour chaque compétition |

### Use case Rest 1

Voici le diagramme de use case de l’API Rest 1 :

Une image contenant texte, capture d’écran, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

Voici un tableau récapitulatif des acteurs et de leurs actions :

|  |  |
| --- | --- |
| Acteurs | Actions |
| API Gateway | Désinscrire un participant : cette action contactera la db car pour chaque compétitions, leurs participants sont enregistrés. |
| Ouvrir une compétition : cette action ajoutera une nouvelle compétition dans la db. |
| Supprimer une compétition : cette action supprimera la compétition de la db. |
| Changer l’état d’une compétition : cette action met à jour une compétition dans la db en changeant la valeur de la colonne « état ». |

### Use case Rest 2

Une image contenant texte, capture d’écran, diagramme, cercle

Le contenu généré par l’IA peut être incorrect.

|  |  |
| --- | --- |
| Acteurs | Actions |
| ApiGateway | Login : il peut vérifier les informations de login indiquées. |
| S’enregistrer : il peut ajouter un nouvel utilisateur avec les informations de login indiquées. |
| Utilisateur | Get compétitions : il peut récupérer la liste des compétitions dans la DB |
| Voter : il peut ajouter le vote d’un utilisateur pour le participant de la compétition indiqué. |
| Participer à compétition : il peut inscrire l’utilisateur indiqué à une compétition. |

### Use case GW

Une image contenant diagramme, capture d’écran, texte, ligne

Le contenu généré par l’IA peut être incorrect.

Ce diagramme reprend les éléments des use-case précédent. Il fait simplement le pont entre le client et l’API adéquate.

## Activity Diagram d'un cas complet

### API Rest 1 (ouvrir compétition)

Une image contenant texte, capture d’écran, diagramme, Parallèle

Le contenu généré par l’IA peut être incorrect.

Ce diagramme représente l’ouverture d’une compétition par un administrateur.

### API Rest 2 (login)

Ce diagramme d’activités représente le login d’un visiteur :

Une image contenant texte, diagramme, capture d’écran, Parallèle

Le contenu généré par l’IA peut être incorrect.

Les actions effectuées dans ce diagramme sont les mêmes que celles dans le diagramme de séquences.

## Maquettes du projet

### Login

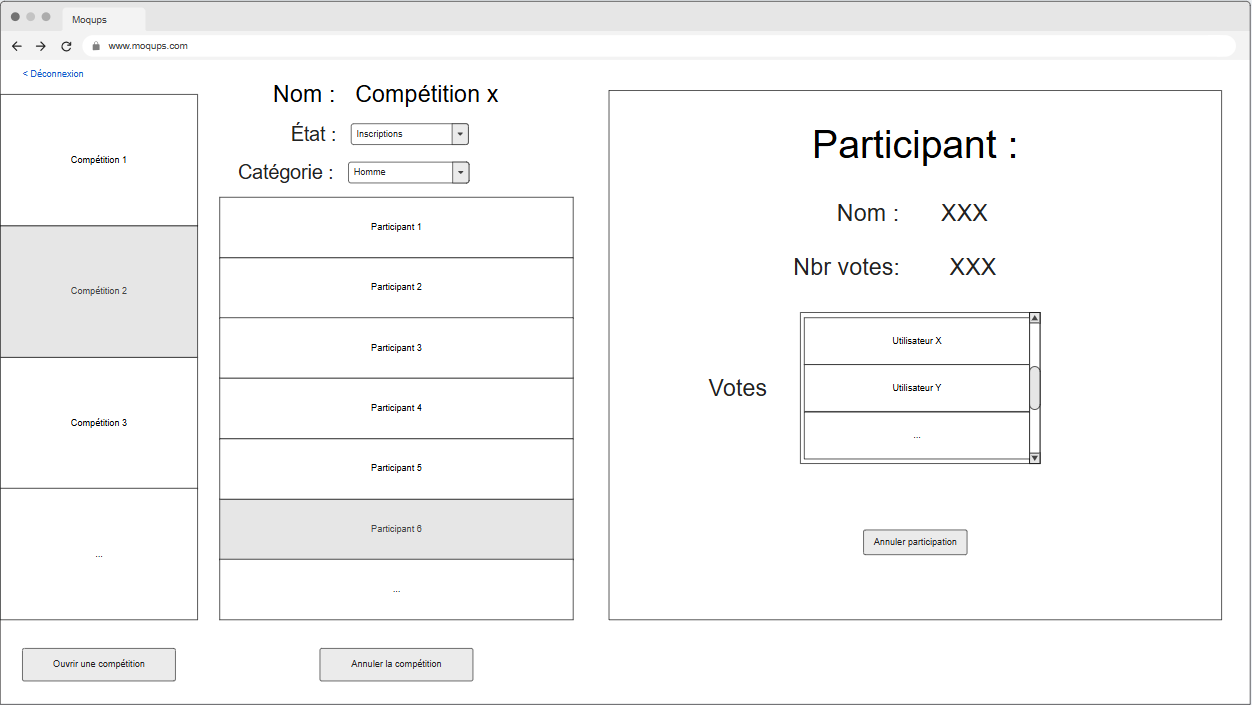
Voici la page de login pour tous les visiteurs :

Une image contenant texte, logiciel, Icône d’ordinateur, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Le visiteur peut se connecter ou s’enregistrer s’il n’a pas de compte.

### Admin



L’administrateur pourra ajouter de nouvelles compétitions, supprimer les compétitions et modifier leur état ou leur catégorie. L’administrateur peut aussi annuler la participation d’un utilisateur.

### Utilisateur

Voici la maquette pour la vue utilisateur :

Une image contenant texte, capture d’écran, nombre, Police

Le contenu généré par l’IA peut être incorrect.

L’utilisateur peut naviguer entre les compétitions pour voir leur catégorie, leur état et leurs participants.

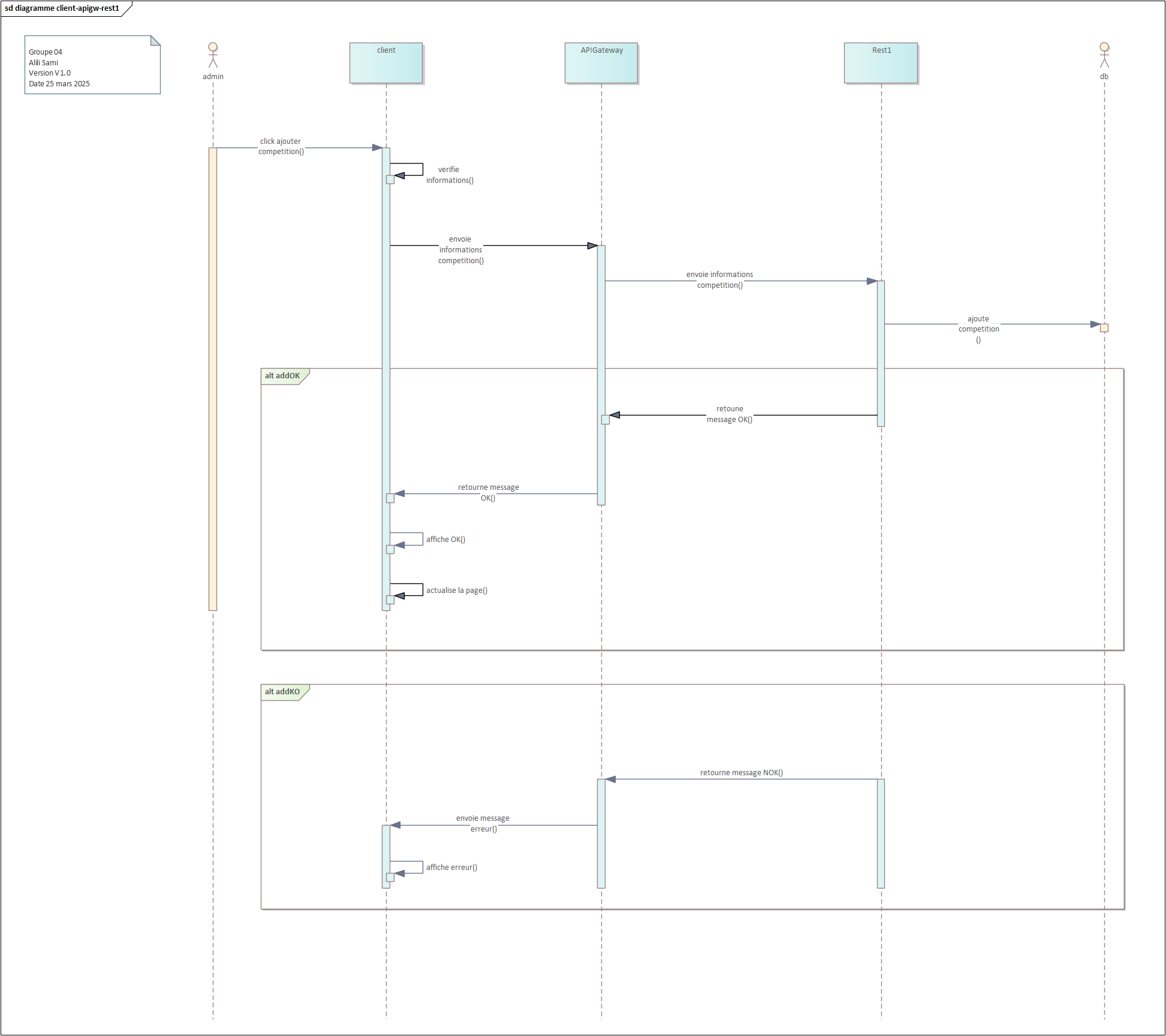
Il peut participer à une compétition si l’état de cette dernière le lui permet.

Il peut également naviguer entre les participants d’une compétition pour voir leur nom, nombres de votes et les utilisateurs ayant voté pour ce dernier.

Il peut voter pour un participant par compétition et annuler son vote s’il veut voter un autre participant.

## Séquence System global entre les applications

### API Rest 1 (ouvrir compétition)



Ce diagramme représente l’ouverture d’une compétition par un administrateur. Ça se déroulera de la manière suivante :

* L’administrateur appuie sur le bouton pour ajouter une compétition.
* L’application client vérifie si les données sont valides.
* L’application client envoie ensuite les informations de la nouvelle compétition à l’ApiGateway.
* L’ApiGateway envoie ensuite les informations au Rest 1.
* Le Rest 1 ajoute la compétition à la DB.
* L’ajout est OK :
  + Retourne un message de succès à l’ApiGateway.
  + L’ApiGateway renvoie le message au client.
  + L’application client affiche le message de succès.
  + L’application rafraichi la liste des compétitions.
* L’ajout est NOK :
  + Le Rest 1 envoie un message d’erreur à l’ApiGateway.
  + L’ApiGateway renvoie le message d’erreur au client.
  + Le client affiche un message d’erreur.

### API Rest 2 (login)

Voici le diagramme de séquences de l’API Rest 2 qui reprend l’action « login ».

Une image contenant texte, diagramme, Parallèle, Plan

Le contenu généré par l’IA peut être incorrect.

* Le visiteur clique sur le bouton du login.
* L’application cliente vérifie les informations entrées dans les champs nécessaires au login.
* Si les champs ne sont pas vides, l’application cliente envoie les informations à l’ApiGateway qui les envoie au Rest 2.
* Le Rest 2 récupère l’utilisateur en question auprès de la DB.
* Le Rest 2 teste si le mot de passe est correct.
* Le login est OK :
  + Retourne un message de succès à l’ApiGateway.
  + L’ApiGateway ouvre une nouvelle session.
  + L’ApiGateway renvoie le message au client.
  + L’application cliente ouvre la page de l’utilisateur.
* L’ajout est NOK :
  + Le Rest 2 envoie un message d’erreur à l’ApiGateway.
  + L’ApiGateway renvoie le message d’erreur au cliente.
  + Le client affiche un message d’erreur.

# Conception

## Diagramme de classes de chaque application

### Application 1

Voici le diagramme de classe de l’application client 1 :

Une image contenant texte, capture d’écran, diagramme, Police

Le contenu généré par l’IA peut être incorrect.

Dans ce diagramme il y a un fichier index, une connexion et un administrateur. Ces fichiers correspondent aux différentes pages de notre application.

Il y a, pour chacun de ces fichiers, un controller.js, et un worker.js commun à tous les controller.

Il y a un fichier douanier (le pont entre les fichiers javascript et java).

### Application 2

Voici le diagramme de classe de l’application client 2 :

Une image contenant texte, capture d’écran, diagramme, Police

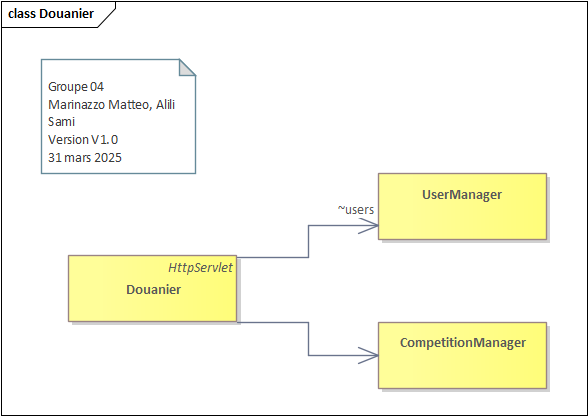
Le contenu généré par l’IA peut être incorrect.

Dans ce diagramme il y a un fichier index, une connexion et un utilisateur. Ces fichiers correspondent aux différentes pages de notre application.

Il y a, pour chacun de ces fichiers, un controller.js, et un worker.js commun à tous les controller.

Il y a un fichier douanier (le pont entre les fichiers javascript et java).

### API Gateway



Ceci représente le diagramme de classe ApiGateway. Elle contiendra la classe Douanier qui appellera les manager adéquat et qui appelleront ensuite les bon Rest.

### API Rest 1

Une image contenant texte, capture d’écran, Rectangle, diagramme

Le contenu généré par l’IA peut être incorrect.

Ce diagramme représente le diagramme de classe Rest1. Il contient le point d’entré CtrlRest1 qui utilisera les 2 services afin d’effectuer certaines actions sur les model à l’aide des repository. Ces services retourneront les DTO et non les model afin de gérer les informations retournées.

### API Rest 2

Une image contenant texte, capture d’écran, diagramme, Post-it

Le contenu généré par l’IA peut être incorrect.

Le diagramme de classes ci-dessus est un diagramme de classes de l’API Rest 2 contenant 11 fichiers.

Un dto utilisateur, un service utilisateurService, voteService, participationService.

Il y a aussi un repository pour le model Utilisateur, Vote, Participation et un controller qui contrôle tout ça.

## Navigation Diagram complète des applications

### Application 1

Une image contenant texte, capture d’écran, diagramme, Parallèle

Le contenu généré par l’IA peut être incorrect.

### Application 2

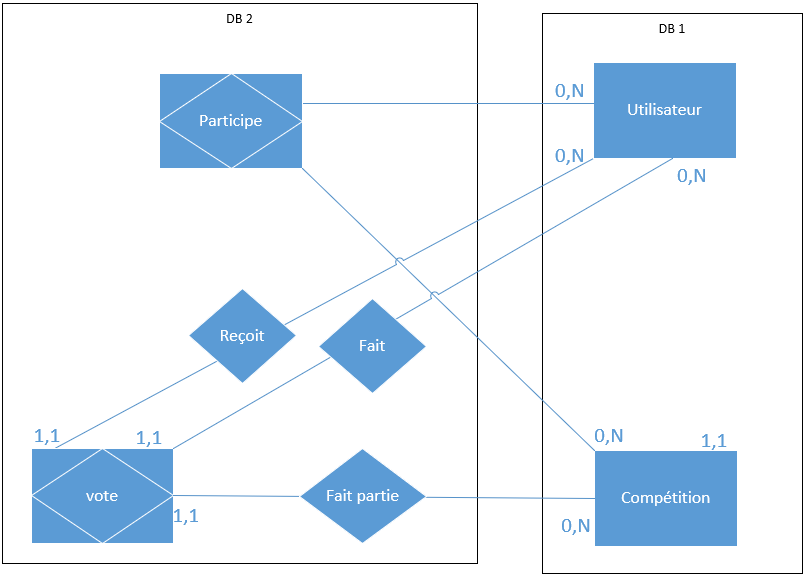
Une image contenant texte, capture d’écran, diagramme, Parallèle

Le contenu généré par l’IA peut être incorrect.

# Bases de données

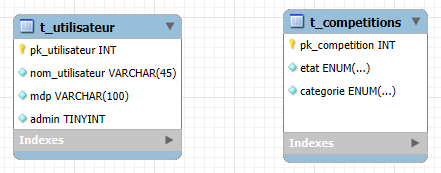
## Modèles ER

Voici le modèle ER de nos deux bases de données :

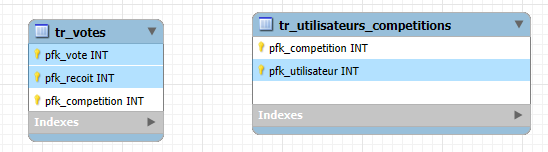


## Modèles Workbench MySQL

## DB 1



## DB 2



# Implémentation des applications <Le client Ap1> et <Le client Ap2>

## Une descente de code client

# Implémentation de l'application <API Gateway>

## Une descente de code APIGateway

# Implémentation des applications <API élève1> et <API élève2>

## Une descente de code de l'API REST

# Hébergement

# Installation du projet complet avec les 5 applications

# Outils, langages (versions, définitions, installations)

# Tests de fonctionnement du projet

# Auto-évaluations et conclusions

## Auto-évaluation et conclusion de client 1

## Auto-évaluation et conclusion de client 2