**Projet Grails**

**Informations :**

* Serveur Web Apache pour stocker les images
* H2 pour la base de données

**Comptes:**

|  |  |  |
| --- | --- | --- |
| ------------------ | Username | Password |
| *Administrateur* | admin | password |
| *Joueur 1* | ines | ines |
| *Joueur 2* | hawaou | hawaou |
| *Joueur 3* | playerUser | playerUser |
| *Joueur 4* | playerTwoUser | playerTwoUser |

Pour la réalisation de notre projet nous avons implémenté deux types de rôle :

* ***ROLE\_ADMIN***: pour les administrateurs
* ***ROLE\_USER***: pour les autres utilisateurs

**Fonctionnalités implémentées :**

* Gestions des Utilisateurs :
  + Les administrateurs peuvent effectuer toutes les opérations de CRUD
  + Les joueurs qui ne sont pas administrateur peuvent se connecter et effectuer des opérations les concernant (voir leur match, message, modifier leur profile ,etc)
* Stockage des images des utilisateurs via les formulaires dans le serveur Apache
* Affichage des images des utilisateurs dans les différentes pages
* Flag sur les messages pour savoir s’ils ont été lu (couleur bleu si un message n’a pas été lu et noir s’il a été lu)

**Bonus**

Cron fonctionnel pour purger les messages, toutes les nuits à 4h du matin. (class JobMessageService,MessageDataService)

A screenshot of a cell phone

Description generated with very high confidence

Figure 1: Modèle de données en fonction des contraintes et besoins du TP

**Documentation et spécification de l’API**

Dans les 3 tableaux ci-dessous nous présentons les différents services de notre API REST et les possibles réponses http.

L’objectif de ce projet est de concevoir et implémenter une API REST. Pour l'implémentation de l’API nous avons utilisé le Framework Grails.

Le modèle de donnée de l’API est composée de trois entités(User,Message,Match).

1. **Ressource « User »**

L’entité User est composée des attributs suivants

* Id
* username
* Password
* Image
* enabled
* accountExpired
* accountLocked
* passwordExpired

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ressource = User | | | | |
| Méthode | **URI** | **Description** | **Réponses** | **Accessible**  **pour** |
| POST | **http://localhost:8081/api/login** | Authentification des utilisateurs. | http 200 OK : liste des rôles + access token | **Tous** |
| http 401 : erreur d’authentification |
| POST | **http://localhost:8081/api/user** | Ajouter un utilisateur. | http 201 OK : utilisateur bien ajouté. | **ADMIN** |
| http 400: erreur dans l’ajout de l’utilisateur. |
| http 401 : Authentification requise. |
| http 403: accès non autorisé pour USER. |
| GET | **http://localhost:8081/api/user/{id}** | Chercher un utilisateur à travers son id. | http 200 : représentation JSON de l’utilisateur. | **ADMIN** |
| http 404 : utilisateur non trouvé. |
| http 401 : Authentification requise. |
| http 403: accès non autorisé pour USER. |
| GET | **http://localhost:8081/api/user** | Chercher tous les utilisateurs. | http 200 : représentation JSON de la liste des utilisateurs. | **ADMIN** |
| http 404 : aucun utilisateur trouvé. |
| http 401 : Authentification requise. |
| http 403: accès non autorisé pour USER. |
| PUT | **http://localhost:8081/api/user** | Modifier un utilisateur. | http 200 : utilisateur bien modifié. | **ADMIN** |
| http 404 : utilisateur non trouvé. |
| http 401 : Authentification requise. |
| http 403: accès non autorisé pour USER. |
| DELETE | **http://localhost:8081/api/user/{id}** | Supprimer un utilisateur à travers son id. | http 200 : utilisateur bien supprimé. | **ADMIN** |
| http 404 : utilisateur non trouvé. |
| http 401 : Authentification requise. |
| http 403: accès non autorisé pour USER. |

Figure 2: Spécifications des Services liés au User

1. **Ressource « Match »**

L'entité Match est composée des attributs suivants

* winner
* winnerScore
* looser
* looserScore

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ressource = Message | | | | |
| Méthode | **URI** | **Description** | **Réponses** | **Accessible**  **Pour** |
| POST | **http://localhost:8081/api/message** | Ajouter un message. | http 201 OK : message bien ajouté. | **ADMIN**  **&**  **USER** |
| http 400: erreur dans l’ajout du message. |
| http 401 : Authentification requise. |
| GET | **http://localhost:8081/api/ message /{id}** | Chercher un message à travers son id. | http 200 : représentation JSON du message. | **ADMIN**  **&**  **USER** |
| http 404 : message non trouvé. |
| http 401 : Authentification requise. |
| GET | **http://localhost:8081/api/ message** | Chercher tous les messages. | http 200 : représentation JSON de la liste des messages. | **ADMIN**  **&**  **USER** |
| http 404 : aucun message trouvé. |
| http 401 : Authentification requise. |
| PUT | **http://localhost:8081/api/ message** | Modifier un message. | http 200 : message bien modifié. | **ADMIN** |
| http 404 : message non trouvé. |
| http 401 : Authentification requise. |
| DELETE | **http://localhost:8081/api/ message /{id}** | Supprimer un message à travers son id. | http 200 : message bien supprimé. | **ADMIN** |
| http 404 : message non trouvé. |
| http 401 : Authentification requise. |

Figure 3: Spécifications des Services liés au Message

1. **Ressource « Message »**

L’entité Message est composée des attributs suivants

* author
* target
* content
* isRead

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ressource = Match | | | | |
| Méthode | **URI** | **Description** | **Réponses** | **Accessible**  **pour** |
| POST | **http://localhost:8081/api/match** | Ajouter un match. | http 201 OK : match bien ajouté. | **ADMIN** |
| http 400: erreur dans l’ajout du match. |
| http 401 : Authentification requise. |
| http 403: accès non autorisé pour USER. |
| GET | **http://localhost:8081/api/ match /{id}** | Chercher un match à travers son id. | http 200 : représentation JSON du match. | **ADMIN** |
| http 404 : match non trouvé. |
| http 401 : Authentification requise. |
| http 403: accès non autorisé pour USER. |
| GET | **http://localhost:8081/api/ match** | Chercher tous les matchs. | http 200 : représentation JSON de la liste des matchs. | **ADMIN** |
| http 404 : aucun match trouvé. |
| http 401: accès non autorisé. |
| http 403: accès non autorisé pour USER. |
| PUT | **http://localhost:8081/api/ match** | Modifier un match. | http 200 : match bien modifié. | **ADMIN** |
| http 404 : match non trouvé. |
| http 401 : Authentification requise. |
| http 403: accès non autorisé pour USER. |
| DELETE | **http://localhost:8081/api/ match /{id}** | Supprimer un match à travers son id. | http 200 : match bien supprimé. | **ADMIN** |
| http 404 : match non trouvé. |
| http 401 : Authentification requise. |
| http 403: accès non autorisé pour USER. |

Figure 4: Spécifications des Services liés au Match

**Procédure de Test**

Nous avons utilisé une collection Postman. Pour chaque ressource, nous avons créé plusieurs scénarios pour les différentes réponses http possibles.

**Bonus**

Utilisation de Spring Security et son extension pour la mise en place dans le cadre d’une API REST : nous avons importé les deux plugins **spring-security-core** et **spring-security-rest** dans le fichier **build.gradle**. Après nous avons configuré à la main les accès aux différents services des ressources exposées dans le fichier **application.groovy**

compile 'org.grails.plugins:spring-security-core:3.2.3'  
compile "org.grails.plugins:spring-security-rest:2.0.0.RC1"