Rapport – Projet M1S2

API Émargement

I. Présentation métier :

A. Présentation du contexte :

Notre application a pour but la création d'une application d'émargement destiné aux élèves, enseignants et personnels administratifs.

Actuellement, une ou plusieurs listes d'émargement, au format papier, sont fournies chaque jour ou semaine aux étudiants. Selon le format, ces listes peuvent concerner tout ou partie des étudiants d'une promotion, et concerner un ou plusieurs enseignements.

Ce fonctionnement pose plusieurs difficultés :

- Son remplissage est fastidieux, et réalisé à la main par les personnes concernées : étudiants, enseignants, membres du personnel administratif.
- Il n'apporte aucune garantie quant la validité et l'exhaustivité des informations saisies :
    - Un étudiant présent peut ne pas émarger, par oubli ou non accès à la feuille.
    - Un étudiant non présent peut émarger pour une séance passée ou future.
    - Un enseignant peut ne pas compléter et signer la feuille, par oubli ou non accès à la feuille.
- Il n'y a aucun historique disponible, ni aucun moyen simple d'accéder aux listes d'émargement en cours de circulation.

Trois acteurs :

- Un étudiant : Émarge pour signifier sa présence lors d'une séance de cours
- Un enseignant : Valide les émargements
- Un membre du personnel administratif : Crée liste d'émargements et consulte la liste d'émargement

B. Présentation de la solution :

Cette application permettra un émargement correct et précis des étudiants pendant les séances d'enseignement. De plus il remplacera les listes d'émargements au format papier, tout en facilitant et sécurisant leur gestion à l'aide de QR-Codes.

Nous avons conçu différentes sortes de fonctionnalités pour chaque type d'utilisateur.

Notre projet Spring Boot est composé de plusieurs couches :

- **Couche Modèle :**
  projet-s2-groupe16/src/java/fr.orleans.m1.wsi.projet2emargement/Modele/

  Couche qui contient l'ensemble de nos modèle implémenté, c'est à dire Modèle, c'est à dire l'ensemble des entités nécessaires au bon fonctionnement de notre projet.
  Exemple : Emargement.java / Etudiant.java / Enseignant.java …

- **Couche Contrôleur :**
  projet-s2-groupe16/src/java/fr.orleans.m1.wsi.projet2emargement/Controller/

  Couche qui contient l'ensemble des contrôleurs permettant la gestion des interactions entre l'utilisateur et l'application, autrement dit l'ensemble des requêtes que l'utilisateur peut exécuter.
  Exemple : EmargementController.java / EtudiantController.java …

- **Couche Repository :**
  projet-s2-groupe16/src/java/fr.orleans.m1.wsi.projet2emargement/Repository/

  Couche permettant l'interaction avec la source de données externes, la base de données.
  Exemple : FacadeEmargement.java / FacadeEtudiant.java …

- **Couche Service :**
  Couche permettant l'implémentation des traitements métiers spécifiques à l'application, sauf que dans notre cas la gestion de ces implémentations a été faite automatiquement.
  La couche service est donc absente du projet, n'empêchant pas le bon fonctionnement de l'application.

  Tout ceci accompagné de quelques tests.

L'API client se situe entre le client et le service web. Sa fonction principale est de transmettre les informations des émargements aux personnels administratifs, de telle sorte qu'ils puissent les modifier à distance.

Cet API est privé et uniquement accessible par un client authentifié.

II. Gestion de projet :

A. Tâches à réaliser :

Pour accéder au fichier regroupant et détaillant, voir le fichier .md
projet-s2-groupe-16/Gestion-Projet/Taches.md

- Liste de tous les étudiants par groupe
- Liste des étudiants présents
- Liste des étudiants absents
- Un semestre est composé d'un ou plusieurs modules
- Un module est composé de plusieurs types de cours (CM/TD/TP)
- Un type de cours (TD/TP) est composé de plusieurs groupes
- Un étudiant est inscrit dans un groupe selon le type de cours
- Chaque étudiant possède un état "Absent-Présent"
- Un enseignant est rattaché à un ou plusieurs groupes
- Un émargement est composé d'une heure, et d'une date et d'un groupe et d'étudiants
- Scan de QR code fait appel à la fonction de changement d'état (absent/présent) selon le groupe
- Un enseignant scan le QR Code (à la fin de l'heure) qui va clôturer l'URL (les émargements)
- Un membre administratif génère la liste d'émargements
- Un membre administratif peut consulter l'historique de toutes les listes d'émargements (closes et ouvertes et en attente)
- L'URL (une fois qu'on scan de QR code) nous envoie vers une page d'authentification :
    - Si utilisateur = enseignant alors avoir la possibilité de clôturer le QR Code
    - Si utilisateur = étudiant alors avoir la possibilité de valider sa présence
- Un membre administratif peut
    - Consulter la liste des identifiants (QR code) qui sont en cours de circulation
    - Consulter la liste des identifiants (QR code) qui sont clôturés

    - Créer/générer un nouvel identifiant (QR code)
    - Consulter les listes d'émargements en accédant à l'identifiant d'un QR code clôturé
- Un QR code possède un temps de validité afin de s'auto-clôturer dans le cas où le prof n'a pas clôturé (entre autres pour "le travail en autonomie")

B. Optionnel :

- Recherche de l'historique des présences d'un seul étudiant
- Recherche de l'historique d'absences d'un seul étudiant

C. Étapes à suivre :

1. Création des entités (Étudiant, Utilisateur, Enseignant, PersonnelAdministratif, Semestre, Module, Groupe, Émargement)
2. Création de la base de données (Émargements)
3. Création de la table des URI et des contrôleurs (Mapping GET & POST)
4. Génération des API
5. Génération config (Spring Security)
6. Requêtes CRUD
7. Création / Génération des tests
8. Documentation : Spring REST Docs (Au fur et à mesure)

III. Documentation Technique :

A. Sources du projet :

Cf partie I B

B. Déploiement du projet :

Pour exécuter notre projet :

Étape 1 :
On lance les services Docker en accédant à "projet-s2-groupe-16/docker-compose.yml" et en démarrant les services.
(Attention le docker doit être déjà lancé sur l'ordinateur de l'utilisateur).
Cette étape nous permet de lancer les services ainsi que de préparer docker à l'accueil de notre base de données.

Étape 2 :
On exécute notre Application main grâce au fichier "projet-s2-groupe-16/src/java/fr.orleans.m1.wsi.projets2emargement/ProjetS2EmargementApplication.java
Cette étape nous permet de lancer notre application.

Étape 3 :
L'application est lancée, l'utilisateur n'a plus qu'à exécuter les requêtes selon ces besoins.

IV. Résumé de l'ensemble

A. Table des URI avec les codes d'erreurs:

Pour accéder au fichier regroupant et détaillant l'ensemble de la table des URI, voir le fichier .md
projet-s2-groupe-16/Gestion-Projet/WeeklyMeeting/TableURI.md

Voici une liste des actions possibles, avec leur URL les verbes http utilisables et les erreurs possibles.

Connexion :

Rôle : (Étudiant, Enseignant, PersonnelAdmin)

POST www.localhost:8080/emargement

- Requête ne nécessitant aucune authentification
- Contient dans le body deux paramètres le login et le mot de passe

- 406 : quand les informations dans les paramètres sont incorrectes

Émargement d'un étudiant :

PUT www.localhost:8080/emargement/{idEmargement}

- Requête authentifiée uniquement disponible pour le rôle (Étudiant)
- Contient dans le body de la requête un paramètre idEmargement qui permettra à un étudiant de valider sa présence pour cet émargement

- 200 : l'émargement est validé
- 409 : l'étudiant a déjà validé sont émargement c'est à dire que son État==Présent
- 404 : l'identifiant idEmargement ne correspond à aucun émargement existant
- 403 : si la personne authentifiée n'a pas accès à cette URI

Clôture d'un émargement par un enseignant :

PUT www.localhost:8080/emargement/{idEmargement}

- Requête authentifiée uniquement disponible pour le rôle (Enseignant)
- Contient dans le body de la requête un paramètre idEmargement qui permettra à un enseignant de clôturer l'émargement

- 200 : l'émargement est clos
- 409 : l'enseignant a déjà clôturé l'émargement
- 404 : l'identifiant idEmargement ne correspond à aucun émargement existant
- 403 : si la personne authentifiée n'a pas accès à cette URI

Liste de tous les émargements clos par un personnel administratif :

GET www.localhost:8080/emargement/clos

- Requête authentifiée uniquement disponible pour le rôle (PersonnelAdm)
- Retourne dans le body la liste des objets Émargement ayant comme etatEmargement==Clos

- 200 : si la liste a bien été récupérée vide ou non vide
- 403 : si la personne authentifiée n'a pas accès à cette URI

Liste de tous les émargements ouverts par un personnel administratif :

GET www.localhost:8080/emargement/ouverts

- Requête authentifiée uniquement disponible pour le rôle (PersonnelAdm)
- Retourne dans le body la liste des objets Émargement ayant comme etatEmargement==Ouvert

- 200 : si la liste a bien été récupérée vide ou non vide
- 403 : si la personne authentifiée n'a pas accès à cette URI

Consultation de l'émargement par un personnel administratif :

GET www.localhost:8080/emargement/{idEmargement}

- Requête authentifiée uniquement disponible pour le rôle (PersonnelAdm)
- Retourne dans le body l'objet Émargement correspondant à l'identifiant idEmargement

- 200 : si l'Émargement existe et est bien récupéré
- 404 : si aucun Émargement ne correspond à cet identifiant
- 403 : si la personne authentifiée n'a pas accès à cette URI

Création d'un nouvel émargement par un personnel administratif :

POST www.localhost:8080/emargement/creation

- Requête authentifiée uniquement disponible pour le rôle (PersonnelAdm)
- Requiert dans le body de la requête une structure JSON de la classe Émargement contenant au moins les champs **A DEFINIR UNE FOIS QU'ON A TRAITE LA FACADE**

- 201 : si l'Émargement a pu être créé sans erreur + Location de la ressource créée
- 406 : si les attributs de l'objet envoyés ne sont pas conformes aux attentes
- 403 : si la personne authentifiée n'a pas accès à cette URI

Liste de tous les étudiants présents par un personnel administratif :

GET www.localhost:8080/emargement/{idEmargement}/present

- Requête authentifiée uniquement disponible pour le rôle (PersonnelAdm)
- Retourne dans le body uniquement la liste des étudiants présents de l'objet Émargement correspondant à l'identifiant idEmargement

- 201 : si l'Émargement existe et la liste des étudiants présents est bien récupérée
- 404 : si aucun Émargement ne correspond à cet identifiant

- 403 : si la personne authentifiée n'a pas accès à cette URI

Liste de tous les étudiants absents par un personnel administratif :

GET www.localhost:8080/emargement/{idEmargement}/absent

- Requête authentifiée uniquement disponible pour le rôle (PersonnelAdm)
- Retourne dans le body uniquement la liste des étudiants absents de l'objet Émargement correspondant à l'identifiant idEmargement

- 201 : si l'Émargement existe et la liste des étudiants absents est bien récupérée
- 404 : si aucun Émargement ne correspond à cet identifiant
- 403 : si la personne authentifiée n'a pas accès à cette URI

Fonctionnalités obligatoires:

Avancement	Fonctionnalités	URI	Méthode	Rôle
	Connexion	/emargement	POST	All
Fait	Emargement d'un etudiant	/emargement/{idEmargement}	PUT	Etudiant
Fait	Cloture d'un emargement par un enseignant	/emargement/{idEmargement}	PUT	Enseignant
Fait	Liste de tous les emargements clos par un personnel administratif	/emargement/clos	GET	PersonnelAdm
Fait	Liste de tous les emargements ouverts par un personnel administratif	/emargement/ouverts	GET	PersonnelAdm
Fait	Consultation de l'emargement par un personnel administratif	/emargement/{idEmargement}	GET	PersonnelAdm
Fait	Creation d'un nouvel emargement par un personnel administratif	/emargement/	POST	PersonnelAdm
Fait	Liste de tous les etudiants presents par un personnel administratif	/emargement/{idEmargement}/present	GET	PersonnelAdm
Fait	Liste de tous les etudiants absents par un personnel administratif	/emargement/{idEmargement}/absent	GET	PersonnelAdm
Fait	Liste de tous les emargements par un personnel administratif	/emargement/	GET	PersonnelAdm

Fonctionnalités optionnelles:
Bonus gestion des entités (CRUD)
Role: PersonnelAdmin

Fonctionnalités	URI	Méthode	Rôle
Liste de tous les enseignants	/enseignant	GET	PersonnelAdm
Ajout d'un enseignant	/enseignant	POST	PersonnelAdm
Consultation des informations d'un enseignant	/enseignant/{idEns}	GET	PersonnelAdm
Suppression d'un enseignant	/enseignant/{idEns}	DELETE	PersonnelAdm
Liste de tous les étudiants	/etudiant/	GET	PersonnelAdm
Ajout d'un étudiant	/etudiant/	POST	PersonnelAdm
Consultation des informations d'un étudiant	/etudiant/{NumEtu}	GET	PersonnelAdm
Suppression d'un étudiant	/etudiant/{NumEtu}	DELETE	PersonnelAdm
Liste de tous les modules	/module/	GET	PersonnelAdm
Ajout d'un module	/module/	POST	PersonnelAdm
Consultation des informations d'un module	/module/{CodeMod}	GET	PersonnelAdm
Suppression d'un module	/module/{CodeMod}	DELETE	PersonnelAdm
Modification des informations d'un module	/module/{CodeMod}	PUT	PersonnelAdm
Liste de tous les semestres	/semestre/	GET	PersonnelAdm
Ajout d'un semestre	/semestre/	POST	PersonnelAdm
Ajout d'un semestre	/semestre/	POST	PersonnelAdm
Consultation des informations d'un semestre	/semestre/{nomSemestre}	GET	PersonnelAdm
Suppression d'un semestre	/semestre/{nomSemestre}	DELETE	PersonnelAdm
Modification des informations d'un semestre	/semestre/{nomSemestre}	PUT	PersonnelAdm
Ajout d'un sous-Module	/SMod/	POST	PersonnelAdm
Liste de de tous les sous-Modules	/SMod/	GET	PersonnelAdm
Consultation des informations d'un sous-Module	/SMod/{NomSMod}	GET	PersonnelAdm
Ajout d'un groupe	/groupe/	POST	PersonnelAdm
Liste de de tous les groupes	/groupe/	GET	PersonnelAdm
Consultation des informations d'un groupe	/groupe/{NomG}	GET	PersonnelAdm
Ajout d'une salle	/salle/	POST	PersonnelAdm
Liste de de toutes les salles	/salle/	GET	PersonnelAdm
Consultation des informations d'une salle	/salle/{salle}	GET	PersonnelAdm

B. Spring REST Docs :

Pour accéder aux fichier regroupant et détaillant l'ensemble des Spring REST Docs, voir les fichiers PDF :

- *projet-s2-groupe-16/src/main/asciidoc/index.pdf*
  Qui contient une description de quelques requêtes OK
- *projet-s2-groupe-16/src/main/asciidoc/index-erreurs.pdf*
  Qui contient une description de quelques requêtes KO