GL1 & CWA : Projet

Document de conception final

Groupe 1 :

CHOUMILOFF Sacha

EL OUALI Ayman

ESCARBAJAL Quentin

GUILLEMIN Sébastien

TRIDARD Jérémy

VERNEY Mathieu

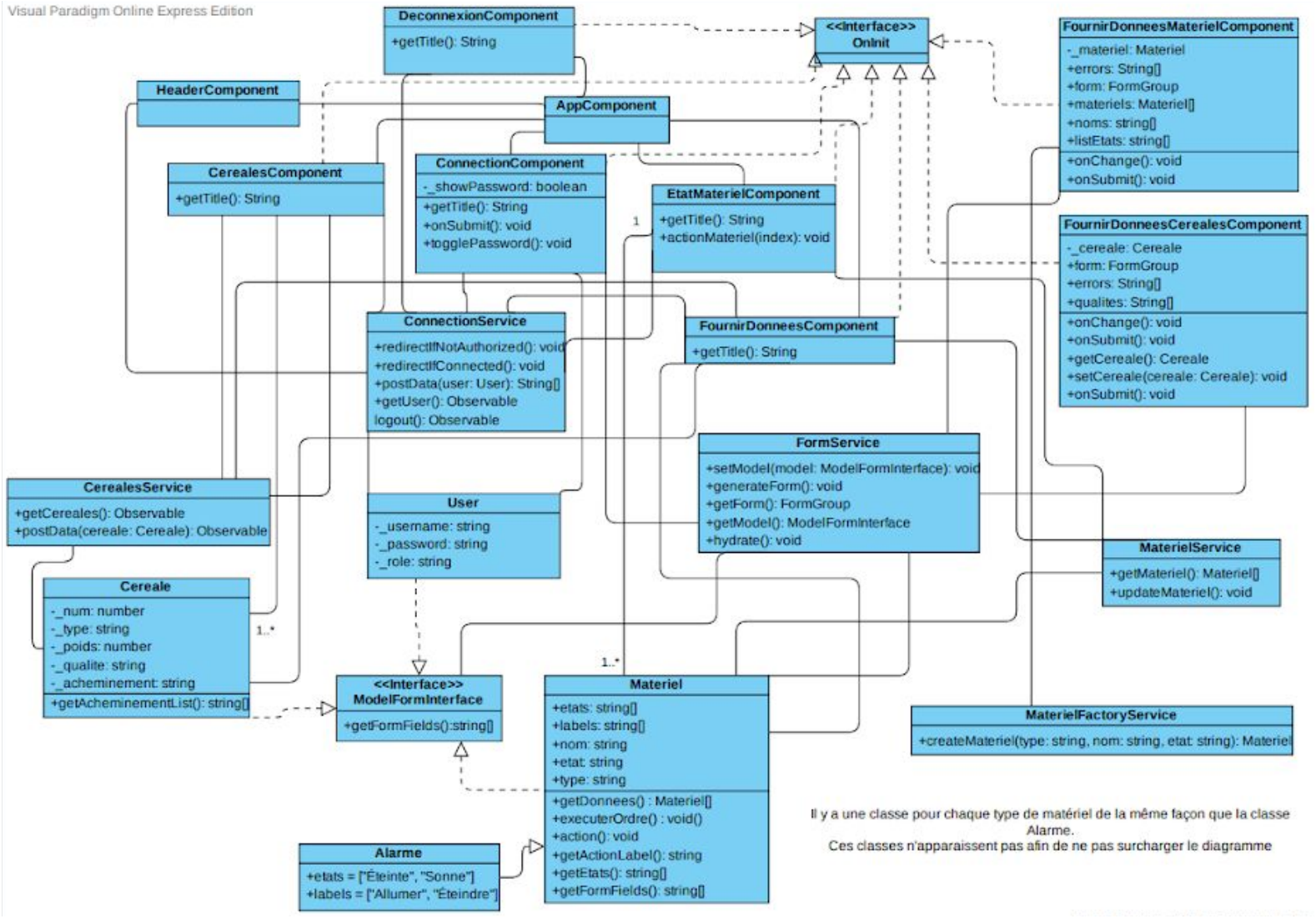
ZOUMAROU WALIS Faïzath Jedida

Ce document présente les diagrammes de classe finaux correspondant à notre application ainsi que les explications relatives aux tests que nous avons réalisés.

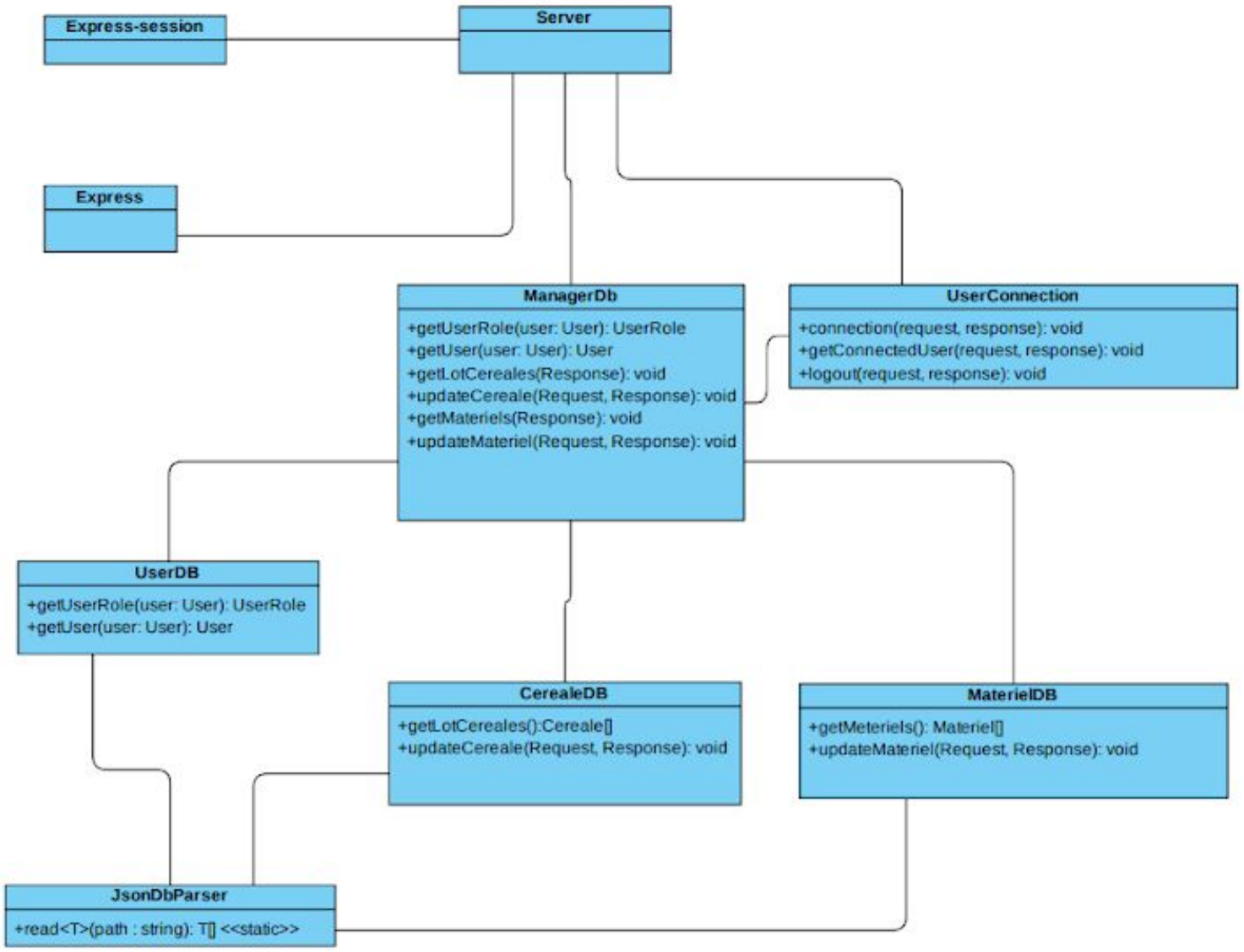
I/ Diagrammes de classe:

Nous avons fait deux diagrammes de classes : un pour la partie Angular et un pour le serveur NodeJS.

1) Angular :



2) NodeJS


Express-session
Server
Express
ManagerDb
+getUserRole(user: User): UserRole
+getUser(user: User): User
+getLotCereales(Response): void
+updateCereale(Request, Response): void
+getMateriels(Response): void
+updateMateriel(Request, Response): void
UserConnection
+connection(request, response): void
+getConnectedUser(request, response): void
+logout(request, response): void
UserDB
+getUserRole(user: User): UserRole
+getUser(user: User): User
CerealeDB
+getLotCereales():Cereale[]
+updateCereale(Request, Response): void
MaterielDB
+getMeteriels(): Materiel[]
+updateMateriel(Request, Response): void
JsonDbParser
+read<T>(path : string): T[] <<static>>

II/ Test:

Pour lancer les tests il suffit de se mettre dans le dossier du projet et de lancer la commande “ng test”. Cette commande va exécuter l’ensemble des fichiers de tests (fichiers “.spec.ts”) et indiquer pour chaque test s’il a échoué ou réussi. Pour chaque test, s’il y a avait plusieurs cas à tester, chaque cas testé est précisé en commentaire (de la même manière que nous le faisions avec un tableau au début du semestre).

Afin de rendre les résultats des tests plus lisibles, une fenêtre chrome s’ouvre et les affiche. Pour voir la couverture du code, il suffit d'ouvrir le fichier “coverage/projet/index.html”. Ce fichier indique, pour chaque dossier, le pourcentage de code couvert.

Afin de tester notre code en continu, nous avons mis en place des tests automatiques lancés à chaque push sur le répertoire git. Ainsi, Github s’occupe de lancer l’ensemble des tests, comme nous le ferions avec la commande “ng test”, et avertit les collaborateurs du projet si un test ne passe pas. Ceci a été très efficace pour détecter une erreur suite à la modification d’une fonction par l’un des membres du groupe.

Nous avons essayé le plus possible d’écrire les tests avant de programmer, à la manière d’un développement TDD, permettant ainsi d’être certains que notre code fasse ce qui était attendu. Cependant, certaines parties de code n’ont pas pu être testées par manque de temps.

Pour tester les fonctions et méthodes effectuant des requêtes vers le serveur NodeJS, nous avons utilisé le mock “HttpTestingController” permettant de capter les requêtes http émises et de retourner une réponse http avec un statut et des données fictives.