

# Équida

Doc Technique

MARTIN Justine  
BOTTON Léa

# Table des matières

<b>1</b>	<b>Contexte et présentation</b>	<b>1</b>
A	Présentation du contexte . . . . .	1
B	Choix techno . . . . .	1
i	MySQL . . . . .	1
ii	Spring Boot . . . . .	1
iii	Ionic . . . . .	1
iv	Gradle . . . . .	1
C	Organisation du projet . . . . .	1
i	Git et branches . . . . .	1
ii	Les différents dossiers . . . . .	1
iii	Trello . . . . .	1
D	Interactions entre les différentes parties du projet . . . . .	2
i	Les différentes parties . . . . .	2
ii	Configuration Gradle . . . . .	3
<b>2</b>	<b>Core</b>	<b>5</b>
A	Organisation des packages . . . . .	5
B	Exemple d'Entity . . . . .	5
C	Exemple de Repository . . . . .	5
D	Exemple de Service . . . . .	5
E	Exemple d'exception (NotFoundException) . . . . .	5
F	Authentification . . . . .	5
G	Utils . . . . .	5
<b>3</b>	<b>Application web</b>	<b>6</b>
A	Organisation des packages . . . . .	6

i	Packages . . . . .	6
ii	Resources . . . . .	6
B	Parler configuration de l'application . . . . .	6
i	application.properties . . . . .	6
ii	Configuration par le code . . . . .	6
C	Fichiers resources . . . . .	6
i	Freemarker . . . . .	6
ii	Assets (images, js, ...) . . . . .	6
D	Parler authentication . . . . .	6
i	Gestion template et controller . . . . .	6
ii	Interceptor . . . . .	6
E	Exemple Route . . . . .	6
F	Exemple Form . . . . .	7
G	Classe InputOutputAttribute . . . . .	7
H	Exemple Controller . . . . .	7
<b>4</b>	<b>Application mobile</b>	<b>8</b>
A	API REST . . . . .	8
B	Organisation des packages . . . . .	8
C	Ionic . . . . .	8
D	Organisation des packages . . . . .	8
<b>5</b>	<b>Ressources</b>	<b>9</b>

# 1 Contexte et présentation

## A Présentation du contexte

## B Choix techno

### i MySQL

### ii Spring Boot

### iii Ionic

### iv Gradle

Gradle est un "build automation system" (moteur de production). Il est un équivalent plus récent et complet à Maven. Il possède de meilleures performances, possède un bon support dans de nombreux IDE et permet d'utiliser de nombreux dépôts, dont ceux de Maven.

## C Organisation du projet

### i Git et branches

#### Branches

#### Nomenclature

### ii Les différents dossiers

#### Doc

#### SQL

#### Sources

### iii Trello

## D Interactions entre les différentes parties du projet

### i Les différentes parties

Le projet Équida est composé de 2 applications. Une l'application web, qui est également l'application principale, une application mobile qui est à usage principal des utilisateurs. Les 2 applications s'appuient sur la même base de données. L'application web y est directement connectée. L'application mobile, elle, passe par une API. En effet, si celle ci se connecterait directement à la base de données comme c'est le cas pour l'application web, une personne mal intentionnée serait en mesure de décompiler l'application mobile afin d'obtenir les identifiants de la base de données. L'utilisation de cette API empêche donc notamment ce problème de sécurité.

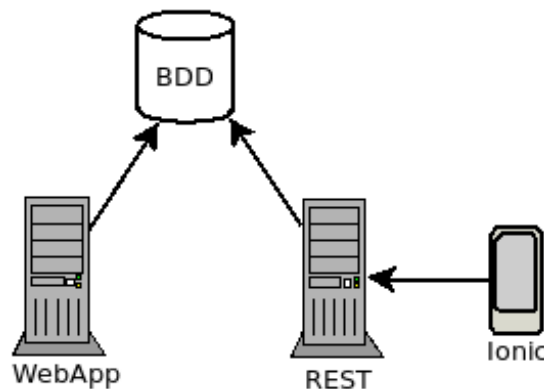


FIGURE 1.1 – La connexion à la BDD selon le projet

L'API ainsi que l'application web utilisent sur le Framework Spring Boot. Ces 2 applications font donc parti de 2 projets différents, "webapp" pour la partie web et "rest" pour l'api. Celles si demandant un code identique pour les Services, les Entités ainsi que les Repository, le choix a donc été fait de faire un projet commun dénommé "core" dans lequel on peut retrouver tout le code qui sera commun aux 2 autres parties, non seulement concernant les éléments cités plus haut mais également concernant les exceptions ou certains outils.

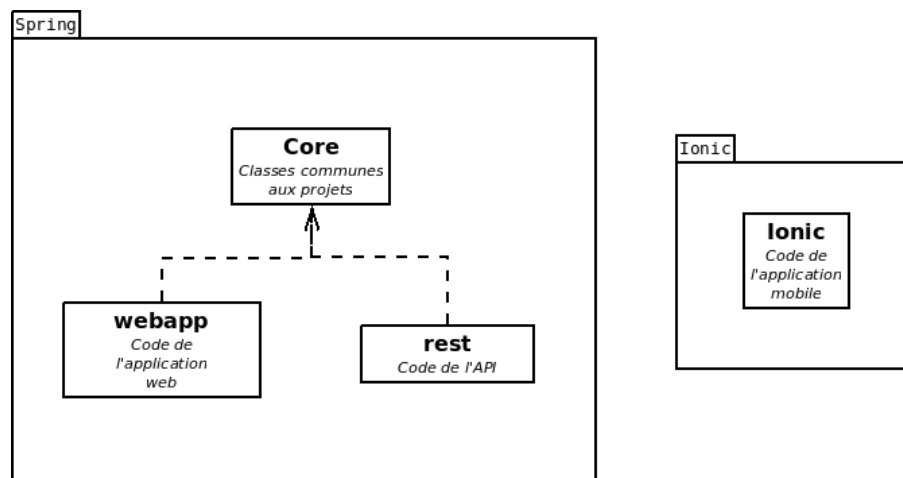


FIGURE 1.2 – Les dépendances entre les projets

## ii Configuration Gradle

Pour gérer correctement les différents projets basés sur Spring, leur dépendances ainsi que leur configuration nous avons donc utilisé Gradle comme mentionné plus haut. Dans le dossier "src/Spring" on retrouve le "build.gradle" qui se charge de configurer tout le projet. On peut observer la configuration suivante pour tout les projets.

```
allprojects {
    apply plugin : 'java'
    apply plugin : 'io.spring.dependency-management'
    apply plugin : 'org.springframework.boot'

    ext {
        springBootVersion = '2.1.3.RELEASE'
    }

    repositories {
        mavenCentral()
        jcenter()
        maven {
            url 'https://plugins.gradle.org/m2/'
        }
    }

    dependencies {
        implementation 'org.springframework.boot:spring-boot-starter'
        implementation 'org.springframework.boot:spring-boot-starter-web'
        implementation 'org.springframework.boot:spring-boot-starter-actuator'
        implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
        implementation 'org.springframework.boot:spring-boot-starter-security'

        testImplementation 'org.springframework.boot:spring-boot-starter-test'
    }
}
```

**FIGURE 1.3** – Configuration Gradle de tous les projets

On définit donc la version de Spring à utiliser, en plus des dépendances commune à chaque projet (spring-boot-starter-web, spring-boot-starter-data-jpa, ...). On va par la suite définir les dépendances uniques à chaque projet.

```

project(':core') {
    jar {
        enabled = true
    }

    dependencies {
        implementation 'com.h2database:h2'
        implementation 'mysql:mysql-connector-java'
    }
}

project(':rest') {
    dependencies {
        implementation project(':core')
    }
}

project(':webApp') {
    dependencies {
        implementation project(':core')

        implementation 'org.springframework.boot:spring-boot-starter-freemarker'
    }
}

```

**FIGURE 1.4** – Configuration Gradle individuelle des projets

De même, concernant le projet core, on active uniquement la compilation en jar (comme une lib) et non pas en jar bootable (comme c'est le cas lorsque l'on utilise Spring Boot).

D'autres scripts "build.gradle" se trouvent dans chaque dossier du projet, cependant, ceux-ci ne configurent que le nom du projet à l'issue du build, la version du JDK utilisée ainsi que le package de base du projet.

## 2 Core

### **A Organisation des packages**

### **B Exemple d'Entity**

### **C Exemple de Repository**

### **D Exemple de Service**

### **E Exemple d'exception (NotFoundException)**

### **F Authentification**

### **G Utils**



## **3 Application web**

### **A Organisation des packages**

**i Packages**

**ii Resources**

### **B Parler configuration de l'application**

**i application.properties**

**ii Configuration par le code**

### **C Fichiers ressources**

**i Freemarker**

**Page de base**

**Page d'erreur**

**Autre**

**ii Assets (images, js, ...)**

### **D Parler authentication**

**i Gestion template et controller**

**ii Interceptor**

### **E Exemple Route**

**F   Exemple Form**

---

**G   Classe InputOutputAttribute**

---

**H   Exemple Controller**

---

## 4 Application mobile

**A API REST**

**B Organisation des packages**

**C Ionic**

**D Organisation des packages**

## 5 Ressources

Github du projet : <https://github.com/justine-martin-study/Equida>

Trello : <https://trello.com/b/jrKixhpu/equida-spring>