Équida

Doc Technique

MARTIN Justine
BOTTON Léa

Table des matières

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 meilleure performances, possède un bon support dans de nombreux IDE et permet d'utiliser de nombreux dépots, dont ceux de Maven.

C Organisation du projet

i Git et branches

Branches

Nommenclature

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 connecté. 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.

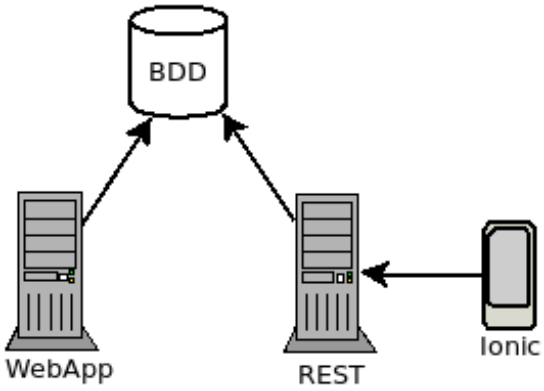

FIGURE 1.1 – La connexion à la BDD selon le projet

L'API ainsi que l'application web sont basées sur le Framework Spring Boot. Ces 2 applications font donc partie 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 dénomé "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.

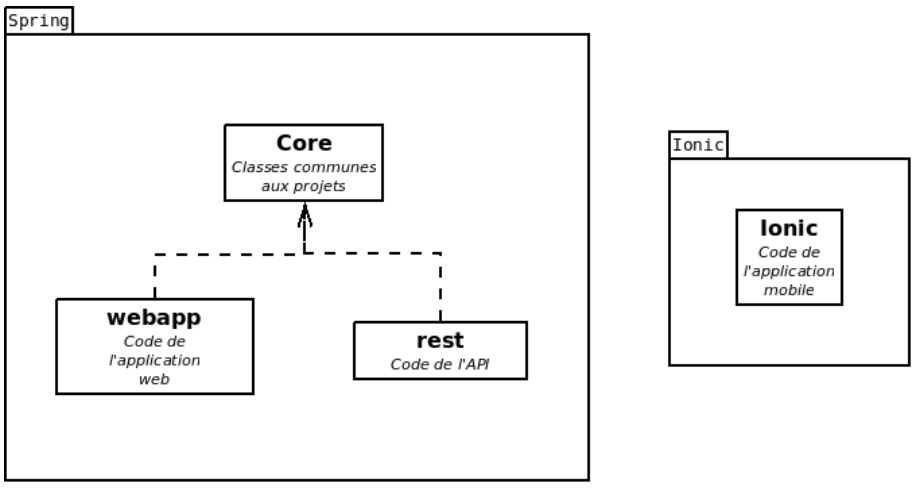

FIGURE 1.2 – Les dépendances entre les projets

ii Configuration Gradle

Pour gérer correctement les différents projets (core, webapp et rest), leur dépendances ainsi que leur configuration nous avons donc utilisé Gradle comme mentionné plus haut. Ainsi dans le dossier "src/Spring" on retrouve le "build.gradle" qui se charge de configurer tout le projet. Ainsi 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 dépendences 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'autre scripts "build.gradle" se trouve dans chaque dossiers du projet, cependant, ceux ci ne configure que le nom du projet à l'issue du build, la version du JDK utilisé 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 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 meilleure performances, possède un bon support dans de nombreux IDE et permet d'utiliser de nombreux dépots, dont ceux de Maven.

C Organisation du projet

i Git et branches

Branches

Nommenclature

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 connecté. 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.

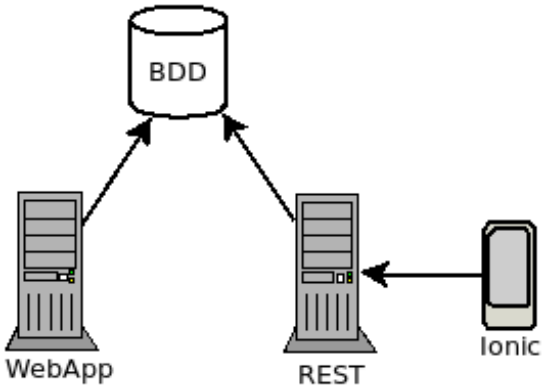

FIGURE 3.1 – La connexion à la BDD selon le projet

L'API ainsi que l'application web sont basées sur le Framework Spring Boot. Ces 2 applications font donc partie 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 dénomé "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.

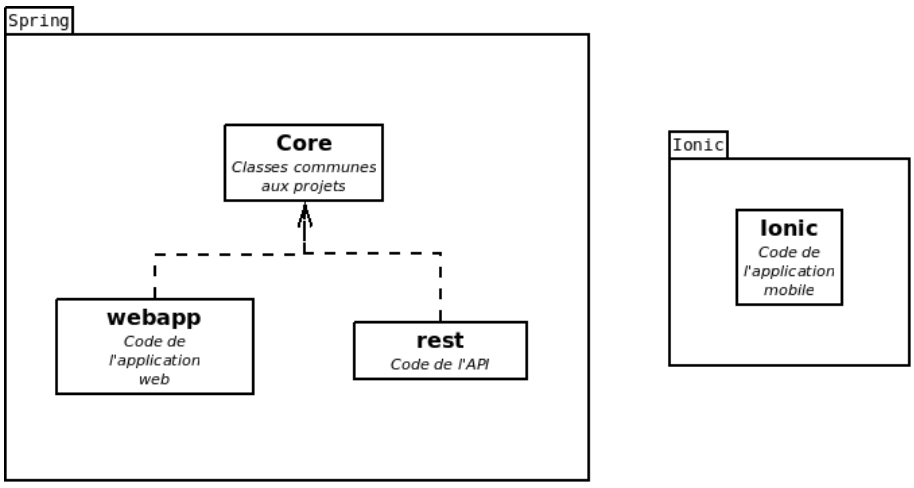

FIGURE 3.2 – Les dépendances entre les projets

ii Configuration Gradle

Pour gérer correctement les différents projets (core, webapp et rest), leur dépendances ainsi que leur configuration nous avons donc utilisé Gradle comme mentionné plus haut. Ainsi dans le dossier "src/Spring" on retrouve le "build.gradle" qui se charge de configurer tout le projet. Ainsi 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 3.3 – Configuration Gradle de tous les projets

On définit donc la version de Spring à utiliser, en plus des dépendences 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 3.4 – 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'autre scripts "build.gradle" se trouve dans chaque dossiers du projet, cependant, ceux ci ne configure que le nom du projet à l'issue du build, la version du JDK utilisé ainsi que le package de base du projet.

4 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 resources

i Freemarker

Page de base

Page d'erreur

Autre

ii Assets (images, js, ...)

D Parler authentification

i Gestion template et controller

ii Interceptor

E Exemple Route

F Exemple Form

G Classe InputOutputAttribute

H Exemple Controller

5 Application mobile

A API REST

B Organisation des packages

C Ionic

D Organisation des packages

6 Ressources

Github du projet : https://github.com/justine-martin-study/Equida
Trello : https://trello.com/b/jrKixhpu/equida-spring