

TP noté de Config Poste de Travail

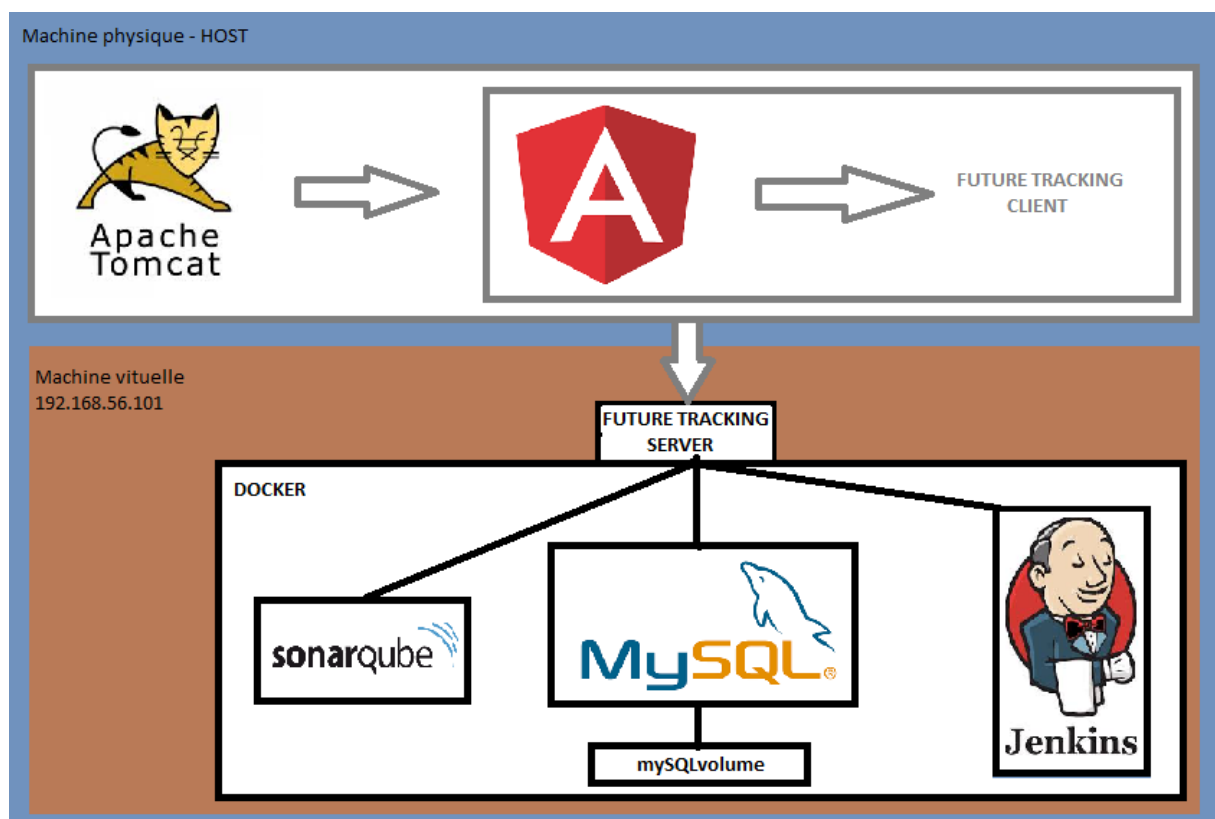
Contexte :

Future Tracking est une startup qui a mis au point des systèmes innovants pour générer des livres de recette au format PDF à partir d'une base de données de recettes. Elle a obtenu une importante levée de fonds et doit embaucher de nouveaux développeurs.

Afin d'accélérer la mise en place et faciliter le démarrage des nouveaux arrivants, le Directeur technique demande à l'équipe aux compétences Full Stack qui a une bonne expérience des outils et process de développement, d'organiser au plus vite l'installation et la configuration des nouveaux postes de développement.

L'équipe chargée de la création de l'environnement de développement doit réaliser des sprints d'une semaine.

ARCHITECTURE N-TIERS



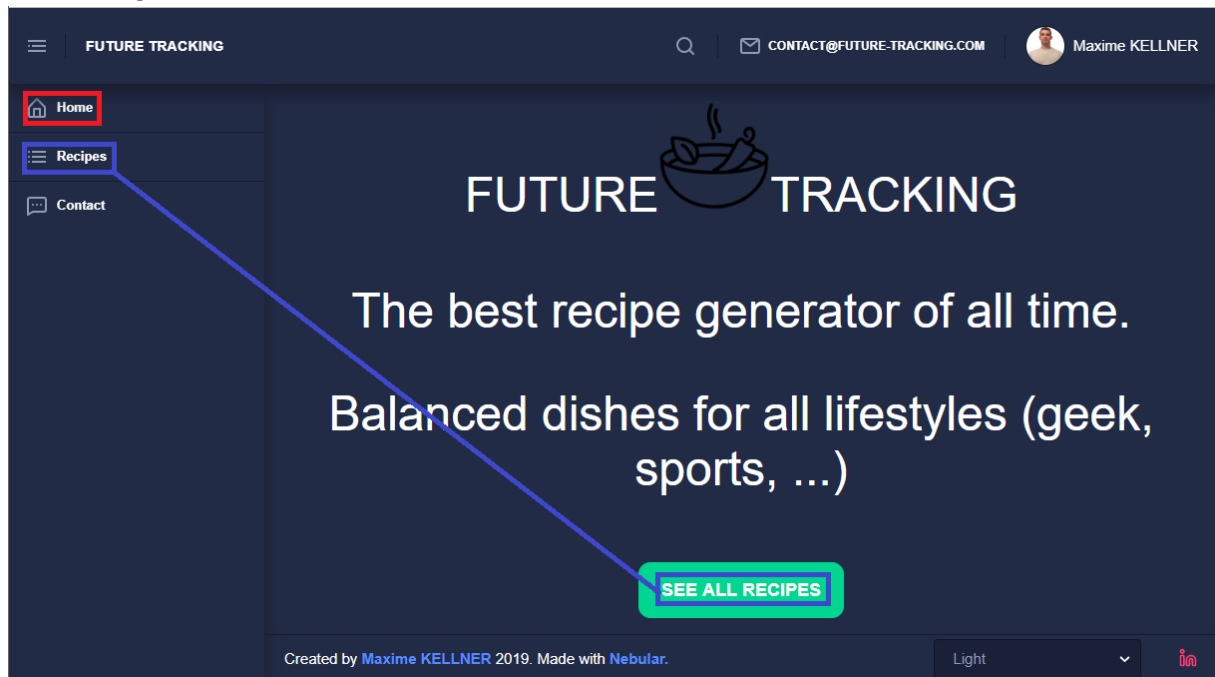
Sur notre machine physique, nous aurons le client « FutureTracking Client » créé avec Angular et qui correspond au site de recette, celui-ci sera déployé sur Apache Tomcat.

Ensuite, nous créerons une machine virtuelle (IP : 192.168.56.101) qui permettra de simuler l'architecture N-tiers en ayant l'application Backend lancé sur un autre serveur (FutureTracking-BackEnd).

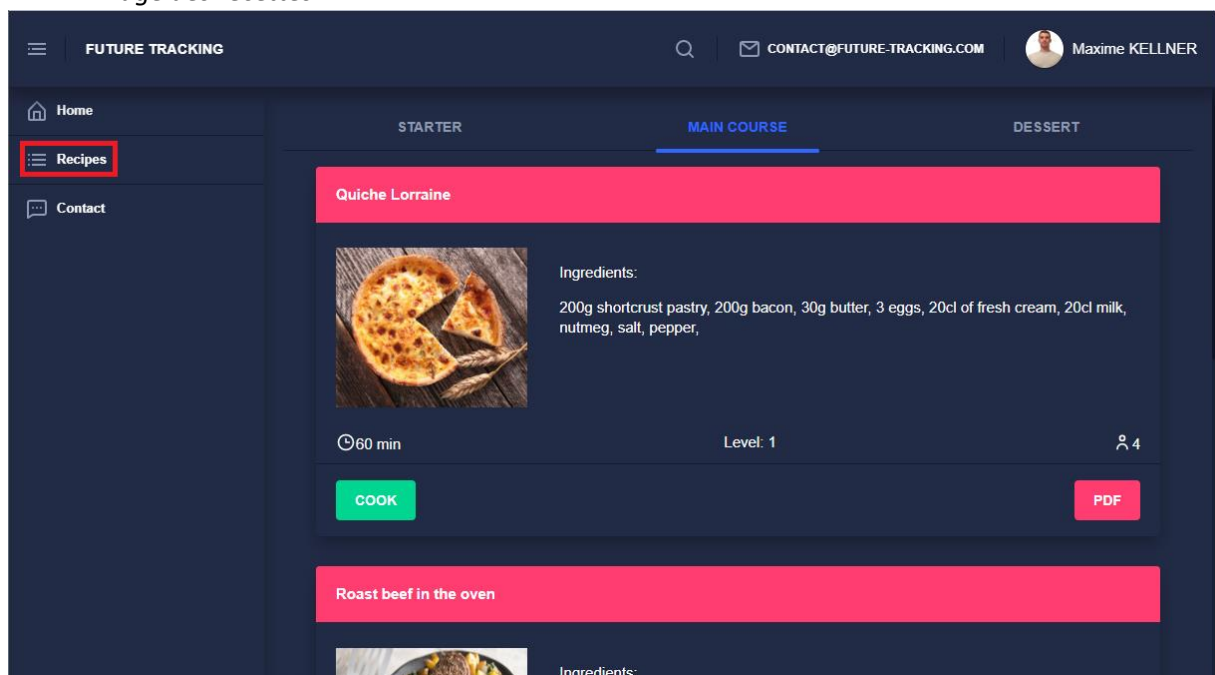
L'outil docker sera également installé sur la machine virtuelle dont l'os est Debian 9 et grâce auquel nous créerons les conteneurs « Jenkins », « Sonarqube » et « MySql » ainsi que le volume « mySQLvolume » pour la persistance des données utilisées dans MySql.

APERCU DE L'APPLICATION WEB FUTURE TRACKING (app client)

➔ Page d'accueil



➔ Page des recettes



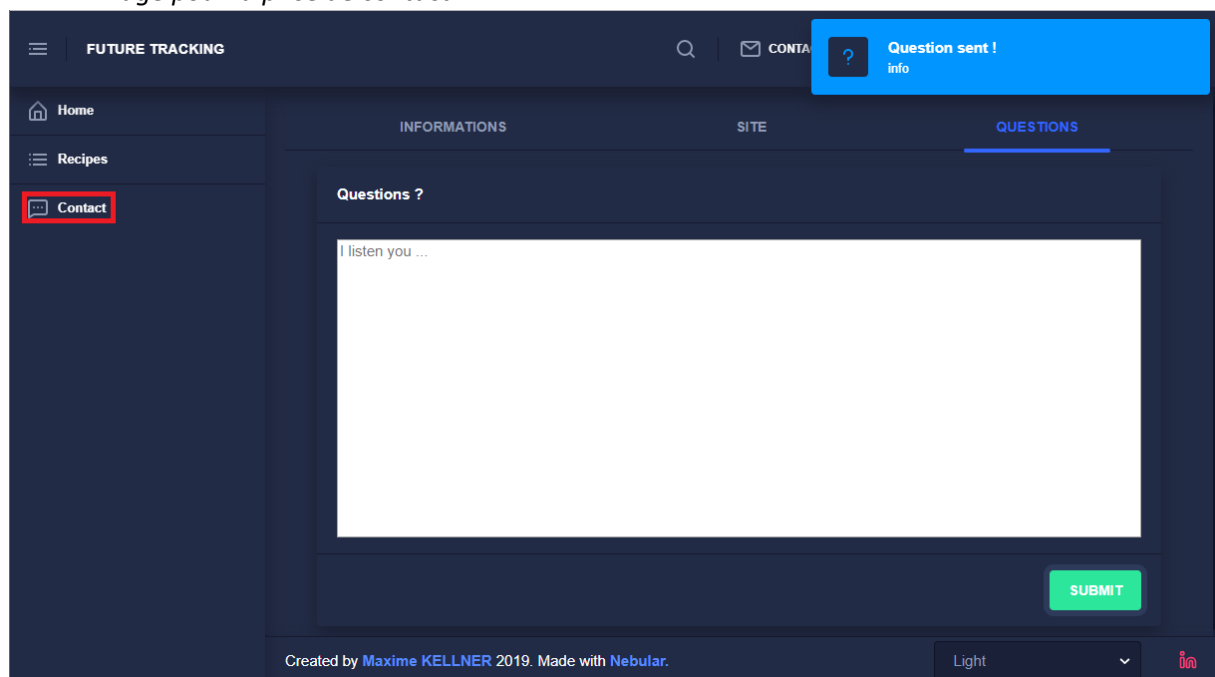
Lors de l'appel à l'onglet « Recipes » sera délivré toutes les recettes contenues dans la base de données à laquelle nous accèderons grâce aux webservices développer grâce à Springboot et Maven en langage JAVA dans l'application côté backend.

Une recette est de la forme :

RECIPE	
id	INT AUTO_INCREMENT PRIMARY KEY
category	INT NOT NULL
name	VARCHAR(255) NOT NULL
description	VARCHAR(255) NOT NULL
ingredients	VARCHAR(255) NOT NULL
person	INT NOT NULL
duration	INT NOT NULL
level	INT NOT NULL
url	VARCHAR(255) NOT NULL

Nous aurions pu avoir une table « INGREDIENT » et référencer les ingrédients via une clé étrangère dans RECIPE (en sachant qu'il aurait même fallu une table « RECIPE_INGREDIENT » comme une recette possède plusieurs ingrédients). Cependant, ici, la modélisation de notre BDD n'est pas l'objectif et pour un gain de temps nous utiliserons cette table uniquement.

➔ *Page pour la prise de contact*



INSTALLATION DU LOGICIEL DOCKER SUR DEBIAN (machine virtuelle)

1 – Dans un premier temps, exécuter les commandes « apt-get update » et « apt-get upgrade » pour être mis à jour.

2 – Exécuter la commande « curl https://releases.rancher.com/install-docker/18.03 »

INSTALLATION DES CONTAINEURS ET VOLUMES

Ce projet aura besoin de :

- ⇒ JENKINS
- ⇒ SONARQUBE
- ⇒ MySQL
 - Un VOLUME pour la persistance de nos données

Les manipulations seront à effectuer en tant qu'administrateur, pour cela taper la commande « su » et saisissez votre mot de passe de root.

A- JENKINS

Etape 1 : Installation de Jenkins

Exécution de la commande :

`docker run -d -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home jenkins/jenkins:its`
(-v jenkins_home crée automatiquement le volume jenkins_home associe au path qui suit)

Cette commande va induire le « pull » de l'image dans une première phase et ensuite lancer l'image avec la redirection sur le port 8080.

Par conséquent, nous pourrions accéder à l'interface Jenkins depuis notre machine physique via l'adresse *192.168.56.101(adresse machine virtuelle) :8080*

Etape 2 : Configuration de Jenkins

Tapez l'adresse 192.168.56.101:8080 dans la barre de navigation du navigateur voulu et il y aura une étape de configuration de Jenkins lors de la première connexion.

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`C:\Program Files\Jenkins\secrets\initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

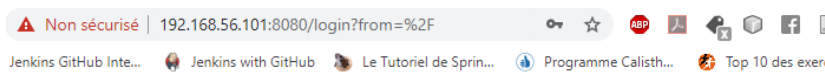
Continue

Récupérer alors le mot de passe (situé dans la console normale avec -ti ou depuis le path indiqué avec -ti) puis copié le et cliquez sur « Continue ».

Getting Started

✓ Folders	OWASP Markup Formatter	Build Timeout	Credentials Binding	<pre>** JDK Tool ** Script Security ** Command Agent Launcher Folders ** bouncycastle API ** Struts ** Pipeline: Step API ** SCM API</pre>
Timestampers	Workspace Cleanup	Ant	Gradle	
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View	
Git	Subversion	SSH Slaves	Matrix Authorization Strategy	
PAM Authentication	LDAP	Email Extension	Mailer	
				** - required dependency

L'assistant de configuration vous proposera de télécharger tous les plugins recommandés, faite donc ce choix et patientez.



Welcome to Jenkins!

Sign in

☐ Keep me signed in

Vous pourrez, par-la-suite, saisir un nom d'utilisateur et mot de passe que vous utiliserez dans la suite du projet pour l'utilisation de Jenkins. Dans notre cas, il s'agira de « maxime / maxime » (pensez à cocher la case pour rester connecté).

Après connexion, vous arriverez donc à l'écran suivant :

Etape 3: Installation des prérequis pour le client

Lorsque nous allons réaliser notre build à l'aide de « pipeline », il nous faudra pouvoir exécuter certaines commandes depuis Jenkins.

Etant un projet « Angular », Jenkins aura par conséquent besoin de nodejs et @angular/cli mais aussi de Sonarqube-Scanner et de zip pour la création de l'artefact pour pouvoir réaliser la tâche d'après-build.

➔ Nous devons nous connecter en tant que root au Shell du conteneur sur lequel nous avons Jenkins

Grâce à un « **docker ps** », nous savons que le nom de notre container est « **fervent_raman** », nous exécutons donc la commande suivante : **docker exec -ti -user root fervent_raman /bin/bash**

NodeJS

Nous avons besoin de NodeJS PPA avant de commencer, exécuter alors :

1. apt-get install curl software-properties-common
2. curl -sL https://deb.nodesource.com/setup_12.x | bash

Avant d'exécuter :

3. **apt-get install nodejs**

```
root@fb8c4c250108:/# apt-get install nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 17.6 MB of archives.
After this operation, 88.0 MB of additional disk space will be used.
Get:1 https://deb.nodesource.com/node_12.x stretch/main amd64 nodejs amd64 12.14.0-1nodesource1 [
MB]
Fetched 17.6 MB in 1s (13.4 MB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used
/usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76, <> line 1.)
debconf: falling back to frontend: Readline
Selecting previously unselected package nodejs.
(Reading database ... 21114 files and directories currently installed.)
Preparing to unpack .../nodejs 12.14.0-1nodesource1_amd64.deb ...
Unpacking nodejs (12.14.0-1nodesource1) ...
Setting up nodejs (12.14.0-1nodesource1) ...
```

@angular/cli

Exécuter «**npm install -g @angular/cli**»

```
root@fb8c4c250108:/# npm install -g @angular/cli
/usr/bin/ng -> /usr/lib/node_modules/@angular/cli/bin/ng

> @angular/cli@8.3.21 postinstall /usr/lib/node_modules/@angular/cli
> node ./bin/postinstall/script.js

? Would you like to share anonymous usage data with the Angular Team at Google under
Google's Privacy Policy at https://policies.google.com/privacy? For more details and
how to change this setting, see http://angular.io/analytics. No
+ @angular/cli@8.3.21
added 251 packages from 186 contributors in 37.596s
```

Sonarqube-Scanner

Exécuter « **npm install -g sonarqube-scanner** »

```
root@fb8c4c250108:/# npm install -g sonarqube-scanner
/usr/bin/sonar-scanner -> /usr/lib/node_modules/sonarqube-scanner/dist/bin/sonar-scanner
+ sonarqube-scanner@2.5.0
```

ZIP

Exécuter « **apt-get install zip** »

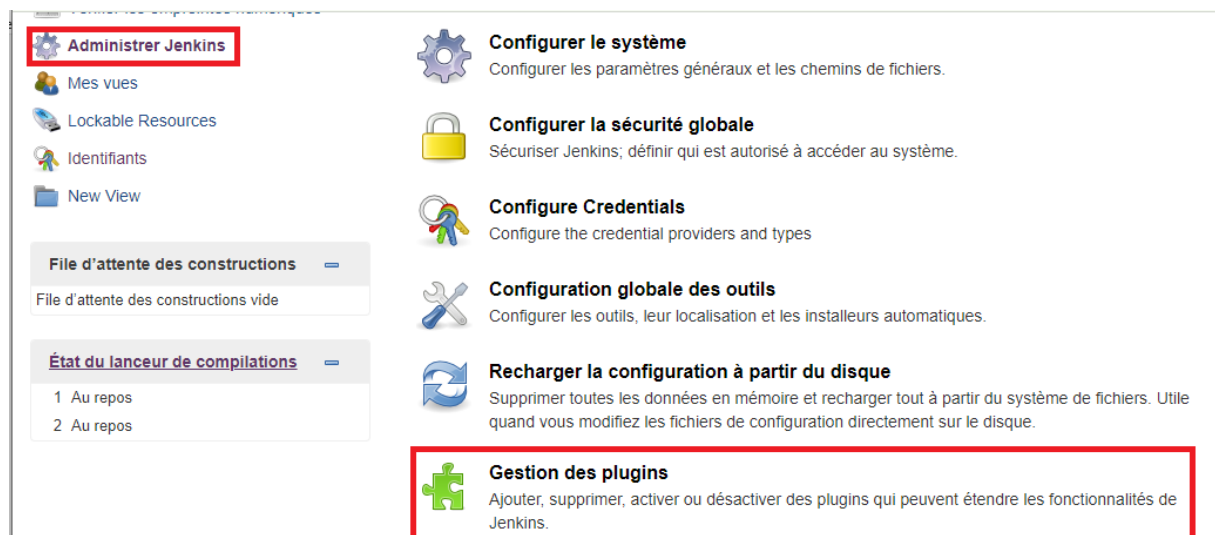
```
root@fb8c4c250108:/# apt-get install zip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  zip
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 234 kB of archives.
After this operation, 623 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian stretch/main amd64 zip amd64 3.0-11+b1 [234 kB]
Fetched 234 kB in 0s (601 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used.
/usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76, <= line 1.)
debconf: falling back to frontend: Readline
Selecting previously unselected package zip.
(Reading database ... 25882 files and directories currently installed.)
Preparing to unpack .../zip_3.0-11+b1_amd64.deb ...
Unpacking zip (3.0-11+b1) ...
Setting up zip (3.0-11+b1) ...
```

Etape 4: Installation des prérequis pour l'application serveur

Nous aurons besoin d'installer Maven pour cela, il suffira d'exécuter la commande « **apt-get install maven** ».

Etape 5 : Paramétrage du credential pour pouvoir communiquer avec le serveur GitHub.

Afin de pouvoir se connecter et récupérer les ressources sur GitHub depuis Jenkins, il sera nécessaire de configurer une clé identifiant et d'installer le Plugin GitHub.



Il faudra recherche le plugin en saisissant son nom dans la barre de recherche :

Filtre:

Mises à jour

Disponibles

Installés

Avancé

Installer	Nom ↓	Version	Installé
<input type="checkbox"/>	Docker Commons Provides the common shared functionality for various Docker-related plugins.	1.16	1.15
<input type="checkbox"/>	Folders This plugin allows users to create "folders" to organize jobs. Users can define custom taxonomies (like by project type, organization type etc). Folders are nestable and you can define views within folders. Maintained by CloudBees, Inc.	6.10.1	6.10.0
<input type="checkbox"/>	GIT server Allows Jenkins to act as a Git server.	1.9	1.8

Télécharger maintenant et installer après redémarrage

Update information obtained: 7 h 37 mn ago

Vérifier maintenant

De sélectionner le plugin une fois affiché

☒ [GitHub Authentication plugin](#)
Authentication plugin using GitHub OAuth to provide authentication and authorization capabilities for GitHub and GitHub Enterprise.

Et de cliquer sur « Installer sans redémarrage » ou « Télécharger maintenant et installer après redémarrage » cela dépend de l'état de la machine Jenkins à cet instant.

Maintenant, nous devons à nouveau revenir dans « Administrer Jenkins », puis cliquer sur « Configurer Credentials » pour ajouter une clé d'authentification sur GitHub

Administrer Jenkins

Mes vues

Lockable Resources

Identifiants

Configurer le système
Configurer les paramètres généraux et les chemins de fichiers.

Configurer la sécurité globale
Sécuriser Jenkins; définir qui est autorisé à accéder au système.
Configurer la sécurité globale

En cliquant sur « Ajouter », nous aurons la fenêtre suivante, dans laquelle il faudra rentrer vos identifiants de connexions à GitHub dans la zone rouge puis nous cliquons sur « Ajouter »

Ajouter des identifiants

Domain

Identifiants globaux (illimité)

Type

Nom d'utilisateur et mot de passe

Portée

Global (Jenkins, esclaves, items, etc...)

Nom d'utilisateur

maximek.travail@gmail.com

Mot de passe

.....

ID

GitHub API token

Description

GitHub Api Access

Ajouter

Annuler

Pour ce qui est des autres champs, tout doit être exactement comme sur la capture d'écran !

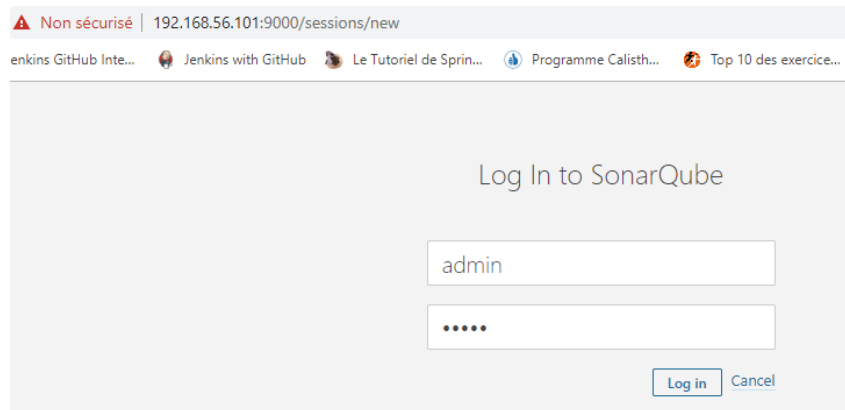
Voilà ! JENKINS EST PRÊT !

B- SONARQUBE

Pour installer Sonarqube, nous n'aurons besoin d'exécuter que la commande suivante :

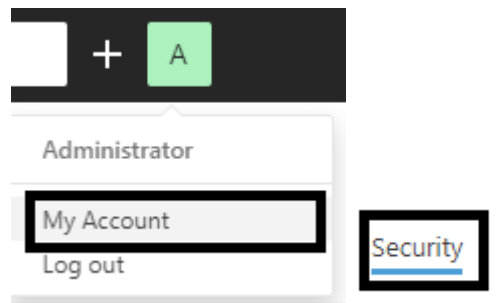
« **`docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube`** »

Ainsi, vous aurez accès à Sonarqube sur l'adresse 192.168.56.101:9000 grâce à la redirection de port et pourrez-vous connecter par défaut avec le couple admin/admin (login/motdepasse)



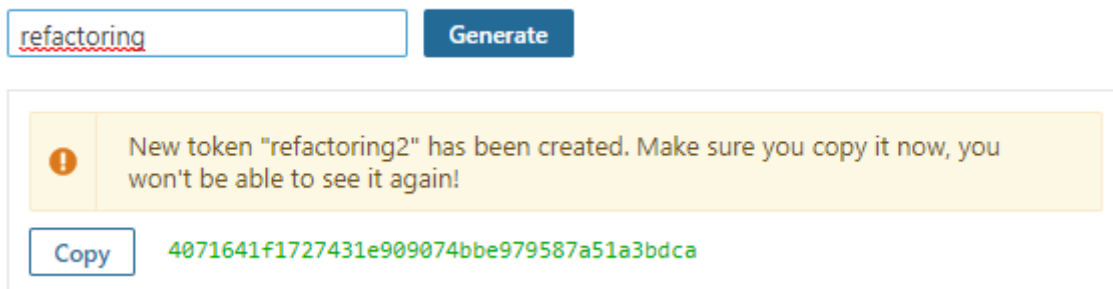
Ensuite, nous créons un Token afin de pouvoir se connecter à SonarQube depuis nos application (token à bien conserver)

- 1- Cliquez sur « MyAccount »



- 2- Taper « refactoring ou autre chose » puis cliquez sur « Générer »

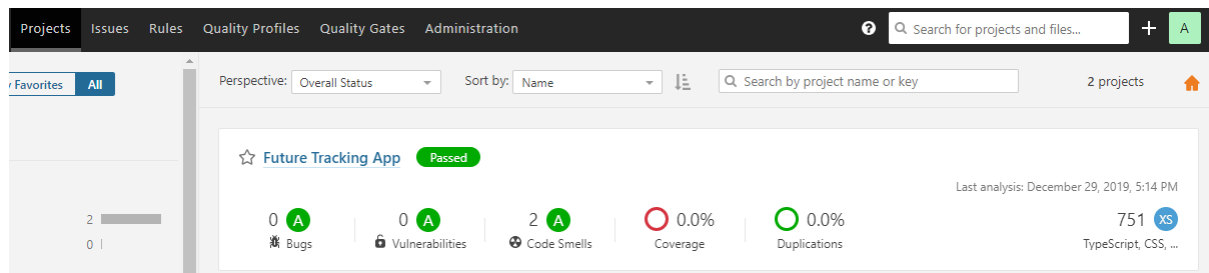
Generate Tokens



Il sera recommandé de précieusement conserver la clé générée car celle-ci n'est pas récupérable après actualisation de la page !

Et voila, a tout moment vous pourrez consulter vos token, et les supprimer si necéssaire :

Name	Last use	Created	
refactoring	21 hours ago	December 12, 2019	Revoke



SONARQUBE EST PRÊT !

C- Volume pour MySQL (impératif de le faire avant de créer le conteneur de MySQL)

Exécuter la commande « ***docker volume create --name mysqlvolume*** » pour créer le volume que nous utiliserons pour MySQL.

« ***docker volume ls*** » pour vérifier la présence du volume

local mysqlvolume

« ***docker volume inspect mysqlvolume*** » pour vérifier la disponibilité du volume

```
root@osboxes:/home/osboxes# docker volume inspect mysqlvolume
[
  {
    "CreatedAt": "2019-12-27T13:53:09-05:00",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/mysqlvolume/_data",
    "Name": "mysqlvolume",
    "Options": {},
    "Scope": "local"
  }
]
```

LE VOLUME EST PRÊT !

D- MySQL

Etape 1 : Installation du conteneur MySQL

Exécuter la commande suivante :

« ***docker run --name mysql_futureTracking -e MYSQL_ROOT_PASSWORD=root -d -p 3306:3306 -v mysqlvolume:/var/lib/mysql mysql*** »

Pour savoir à quel path associer un volume il suffit de vérifier le DockerFile de mysql avec le tag latest dans notre cas et de voir que le path `:/var/lib/mysql` est le path de stockage.

On fait un « **docker ps** » pour vérifier l'exécution de notre conteneur :

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
MES	mysql	"docker-entrypoint.s..."	4 seconds ago	Up 3 seconds	33060/tcp, 0.0.0.0::3306->3306/tcp

MySQL EST PRÊT !

```
root@osboxes:/etc# docker exec -it mysql_futureTracking /bin/bash
root@fc7abfe01111:/# mysql -V
mysql Ver 8.0.18 for Linux on x86_64 (MySQL Community Server - GPL)
```

INSTALLATION ET CONFIGURATION DE L'OUTIL « SQLDEVELOPPER »

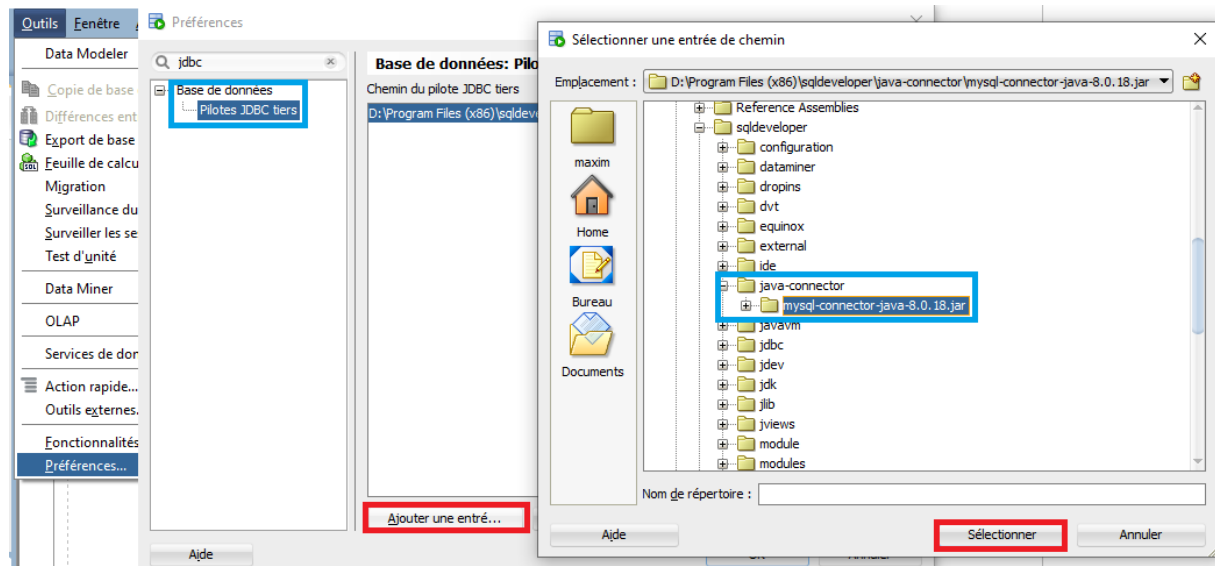
Etape 1 : Installation du pilote de gestion des connexions MySQL

MySQL n'a pas d'interface web pour gérer la BDD, il faut passer par un outil comme SQLDeveloper (sur lequel on ajoute la library jdbc-connector pour pouvoir prendre en charge les connexions vers une BDD MySQL)

Lien de SQLDeveloper : <https://www.oracle.com/fr/tools/downloads/sqldev-v192-downloads.html>

Lien de jdbc-connector : <https://dev.mysql.com/downloads/connector/j/> (plateform independant)

Puis ouvrez SQL > Onglet « Outils » > Sélectionnez « Préférences » > paramètre « Pilotes JDBC tiers » puis il suffit d'ajouter le .jar du jdbc-connector comme indiqué sur la capture ci-dessous !



Etape 2 : Etablissement d'une « instance » de base de données et création d'un utilisateur pour le projet FutureTracking

1. Se connecter au Shell du docker MySQL : « **docker exec -it mysql_futureTracking /bin/bash** »

```
root@osboxes:/etc# docker exec -it mysql_futureTracking /bin/bash
root@fc7abfe01111:/# mysql -V
mysql Ver 8.0.18 for Linux on x86_64 (MySQL Community Server - GPL)
```

2. Créer une BDD dédiée au projet FutureTracking

Pour cela, nous aurons besoin de nous connecter à MySQL en tant que root avec la commande « **mysql -u root -p** » et nous créerons ensuite la BDD avec la commande « **CREATE DATABASE futureTracking ;** »

```
root@9f6efca41d18:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 53
Server version: 8.0.18 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE futureTracking;
Query OK, 1 row affected (0.07 sec)
```

Base de données CREEE !

3. Créer un utilisateur et lui donner tous droits sur la BDD « futureTracking »

Toujours en étant connecté en tant que root au service MySQL, il suffit de créer un utilisateur avec la commande « **CREATE USER 'future'@'%' IDENTIFIED BY 'future'** », et de lui donner tous les droits sur tout le contenu de la BDD grâce à la commande « **GRANT ALL PRIVILEGES ON 'futureTracking.*' TO 'future'@'%' ;** »

Ensuite on enregistre la configuration des droits données à notre nouvel utilisateur :
« **FLUSH PRIVILEGES ;** »

```
mysql> CREATE USER 'future'@'%' IDENTIFIED BY 'future';
Query OK, 0 rows affected (0.38 sec)

mysql> GRANT ALL PRIVILEGES ON futureTracking.* TO 'future'@'%' ;
Query OK, 0 rows affected (0.36 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.11 sec)

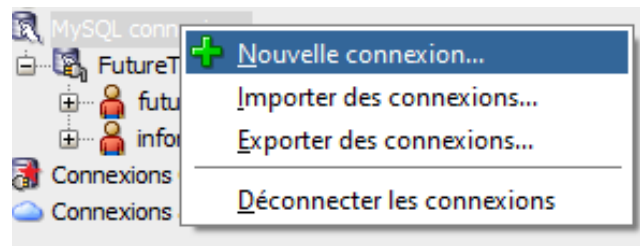
mysql> █
```

A cet instant, nous avons un utilisateur « future » attribué pour le projet « Future Tracking » et une BDD qui lui est propre !

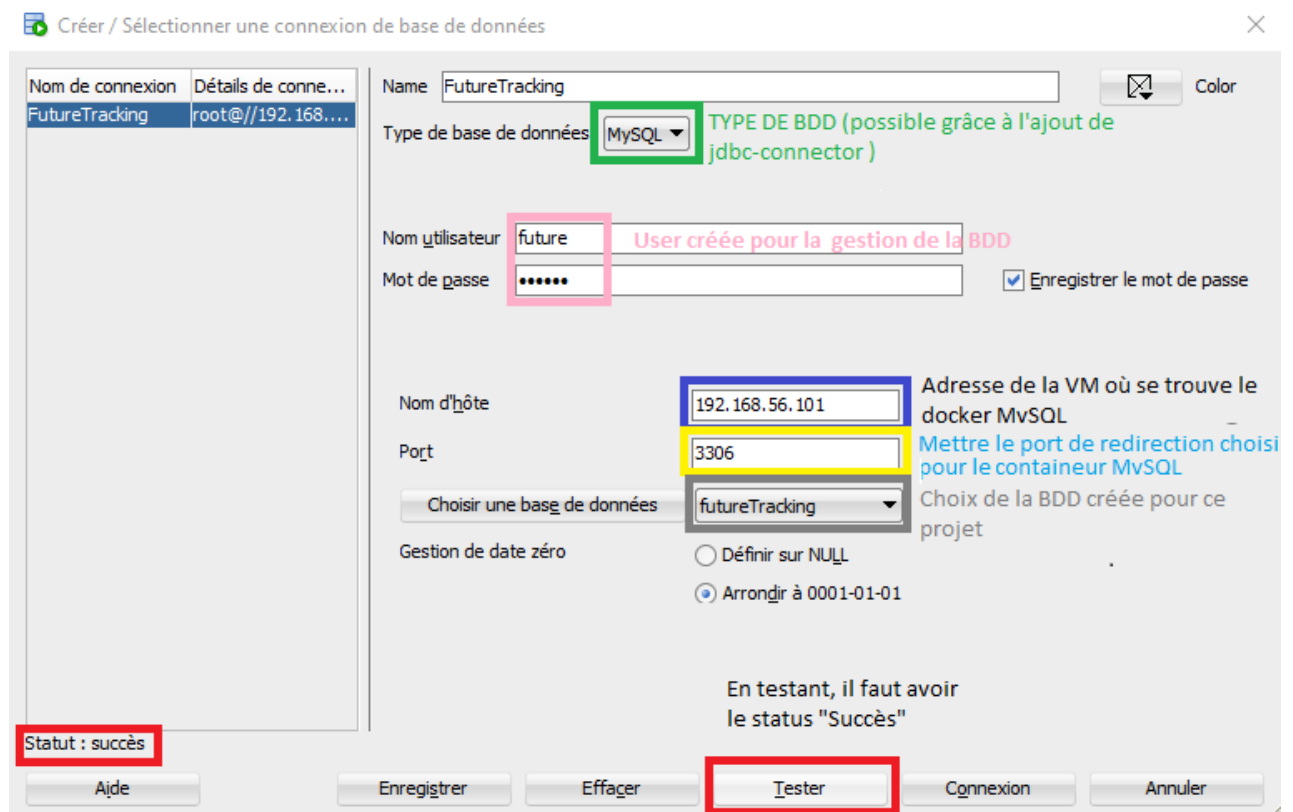
Il ne manque plus qu'à se connecter à cette BDD à l'aide de SQLdeveloper que nous avons déjà configuré pour accepter une connexion à MySQL.

Etape 3 : Connexion à la BDD MySQL via SQLdeveloper

Pour créer une connexion, clique-droit sur MySQL connexion puis de faire « Nouvelle connexion »



Une nouvelle fenêtre s'affiche alors et dans laquelle, après avoir choisis « MySQL » dans le type de BDD voulue, nous rentrerons les informations suivantes :



Et enfin, cliquez sur « **Connexion** » !

Nous sommes enfin prêts à utiliser pleinement MySQL !

CREATION DE LA TABLE RECIPE via SQLdeveloper

En effectuant un double-clic (gauche) sur la base de données futureTracking qui sera affichée une fois la connexion établie, il vous sera possible d'accéder à un éditeur de texte qui permettra l'exécution des commandes SQL.

Ce script sera très simple et nous permettra de réaliser le stockage des données de notre projet, voici donc le script que nous insérerons pour la création de la table et l'insertion des données

Script :

```
1 | DROP TABLE IF EXISTS TB_RECIPE;
2 |
3 | CREATE TABLE TB_RECIPE (
4 |   id_recipe INT AUTO_INCREMENT PRIMARY KEY,
5 |   name VARCHAR(255) NOT NULL,
6 |   category INT NOT NULL,
7 |   description VARCHAR(255) NOT NULL,
8 |   ingredients VARCHAR(255) NOT NULL,
9 |   person INT NOT NULL,
10 |  duration INT NOT NULL,
11 |  level INT NOT NULL,
12 |  url VARCHAR(255) NOT NULL
13 | );
14 |
15 | INSERT INTO TB_RECIPE VALUES (1, "Jeanne's timpani", 0, "", "4 slices of smoked salmon, 2 zucchini, 3 eggs, 10 cl thick fresh cream, 1 small garlic clove, 1 tablespoon olive
16 | INSERT INTO TB_RECIPE VALUES (2, "Carpaccio of scallops with lemon caviar with lemon caviar", 0, "", "12 large scallops, 3 lemongrass sticks, 5 cl olive oil, 1 lemon caviar,
17 | INSERT INTO TB_RECIPE VALUES (3, "Quiche Lorraine", 1, "", "200g shortcrust pastry, 200g bacon, 30g butter, 3 eggs, 20cl of fresh cream, 20cl milk, nutmeg, salt, pepper", 60,
18 | INSERT INTO TB_RECIPE VALUES (4, "Roast beef in the oven", 1, "", "400g barded beef roast, 1 garlic clove, 2 tablespoons olive oil, 12cl of water, 1 pepper, 1 salt, 1 thyme",
19 | INSERT INTO TB_RECIPE VALUES (5, "Potato gratin", 1, "", "1kg of potato, 60g grated cheese, 25cl of fresh cream, 30g butter, 1 garlic clove, pepper, salt", 70, 4, 1, "https://
20 | INSERT INTO TB_RECIPE VALUES (6, "Chocolate fondant", 2, "", "200g dark chocolate, 150g butter, 150g granulated sugar, 50g flour, 3 eggs", 40, 4, 1, "https://assets.tmeccosys.
```


Exécution du script avec la commande « **F5** » sur votre clavier ou en cliquant sur  et si tout se passe bien vous devriez avoir ceci :

Table TB_RECIPE supprimé(e) .

Table TB_RECIPE créé(e) .

1 ligne inséré.

1 ligne inséré.

PROJET « FutureTracking-BackEnd »

A- Création du projet

Je souhaite donc créer une application me permettant d'appeler des webservices qui me fournirait les données en BDD et retournerai un JSON, etc...

Nous allons donc générer un projet Springboot Maven utilisant JAVA et le plugin pour utiliser Hibernate:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Mais il est préférable de passer par le site <https://start.spring.io/> qui initialisera un projet voulu (Maven / Gradle) avec la version de java voulue, etc...

Après avoir rempli le formulaire, il vous sera possible de générer un zip du projet initialisé et d'ouvrir ce projet par exemple avec NetBeans dans notre cas

Voici comment remplir le formulaire en n'omettant pas d'ajouter le plugin pour Hibernate :

The screenshot shows the Spring Initializr web form. The 'Project' tab is selected, with 'Maven Project' chosen. The 'Language' is 'Java'. Under 'Spring Boot', version '2.2.2' is selected. In the 'Project Metadata' section, the 'Group' is 'com', the 'Artifact' is 'futureTracking', and the 'Options' are expanded to show 'Name' as 'futureTracking', 'Description' as 'recipe site', 'Package name' as 'com.futureTracking', 'Packaging' as 'Jar', and 'Java' version as '8'. The 'Dependencies' section shows a search bar with 'Web, Security, JPA, Actuator, Devtools...' and a list of 'Selected dependencies' including 'Spring Data JPA'. At the bottom, there are buttons for 'Generate - Ctrl + G', 'Explore - Ctrl + Space', and 'Share...'. Footer text includes '© 2019 Pivotal Software', 'Spring Initializr and PWS', and '1 of Year Theme Credits'.

B- Gestion de version (GITHUB)

Il nous faudra par la suite, ajouter le projet à GitHub afin d'avoir un gestionnaire de version et de pouvoir bénéficier du Team collaboration.

Pour cela, l'outil « Git Bash » sera nécessaire ! Pour le télécharger : <https://gitforwindows.org/> (Exécutez le fichier et suivez les étapes)

Une fois installé, rendez-vous dans le répertoire où se trouve le projet « FutureTracking-BackEnd » et réaliser dans l'ordre les commandes suivantes :


1. Aller sur GitHub, créer un nouveau projet « FutureTracking-BackEnd » vide, puis récupérer le lien de clone <https://github.com/MaxeeZ/FutureTracking-BackEnd.git>
2. Dans le terminal Git Bash, à votre première utilisation, réaliser les commandes suivantes :
 - a. `git config --global user.name "nom GitHub"` (MaxeeZ)
 - b. `git config --global user.email email@deGitHub` (maximek.travail@gmail.com)

3. Dans le terminal Git Bash, à la racine du répertoire, effectuer la commande « **git init** »
4. Effectuer la commande :
« **git remote add origin** <https://github.com/MaxeeZ/FutureTracking-BackEnd.git> »
5. Effectuer la commande : « **git add .** »
6. Effectuer la commande : « **git commit -m « Initial Commit »** // pour initialiser le répertoire avec toutes nos ressources
7. Effectuer la commande : « **git push -u origin master** »
8. **Vérifier** que le répertoire GitHub est bien initialisé !

The screenshot shows the GitHub interface for a repository named 'MaxeeZ / FutureTracking-BackEnd'. The repository is private and has 1 watch, 0 stars, and 0 forks. The main navigation bar includes links for Code, Issues (0), Pull requests (0), Actions, Projects (0), Security, Insights, and Settings. Below the navigation bar, there is a section for repository statistics: 5 commits, 1 branch, 0 packages, and 0 releases. A 'Clone or download' button is visible, which has opened a dropdown menu showing options to clone with HTTPS or SSH, and links to open in desktop or download ZIP. The file list shows several files and folders, including .mvn/wrapper, src, .gitignore, Jenkinsfile, README.md, exemple.log, mvnw, mvnw.cmd, and pom.xml, with their respective commit messages and timestamps.

C- Intégration du projet dans Jenkins depuis GitHub

1. Ajouter un projet dans jenkins

Cliquez sur  **Nouveau Item** puis tapez nom « FutureTracking-BackEnd » et sélectionner « Pipeline » pour le nouvel item créé.

The screenshot shows the Jenkins 'New Item' form. The 'Saisissez un nom' (Enter a name) field is filled with 'FutureTracking-BackEnd'. Below the form, the 'Pipeline' option is selected, which is described as 'Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (form'.

Après avoir cliqué sur « OK », il faudra se rendre à la section « Pipeline », puis indiqué qu'il s'agira de la construction d'une build à partir d'un JenkinsFile qui sera récupéré depuis le gestionnaire de version GitHub, pour cela il faudra renseigner l'adresse du projet GitHub que nous avons initialisé tout à l'heure !

Pour pouvoir accéder à l'API GitHub en rapport avec notre compte, nous utiliserons l'identifiant qui est lié à notre compte GitHub et enfin il n'y aura plus qu'à valider la configuration.

Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories:

Repository URL:

Credentials: [Ajouter](#)

[Avancé...](#)

[Add Repository](#)

Branches to build:

Branch Specifier (blank for 'any'):

[Add Branch](#)

Navigateur de la base de code: (Auto)

Additional Behaviours: [Ajouter](#)

Script Path:

Lightweight checkout: ☒

[Pipeline Syntax](#)

[Sauver](#) [Apply](#)

Le résultat :

S	M	Nom du projet ↓	Dernier succès	Dernier échec	Dernière durée
		FutureTracking-BackEnd	3 mn 12 s - #15	s. o.	52 s 

La build est pour l'instant au rouge car il recherche un fichier de test non construit encore car il n'y a pour l'instant aucun test.

D- JENKINSFILE

Nous utiliserons la machine « maître » par défaut et aucun slave pour l'exécution de nos builds :

```
agent {
    label 'master'
}
```

Nous vérifierons tous les 5 minutes le répertoires GitHub afin de réaliser de nouvelles builds en cas d'évolution / modification sur le répertoire :

```
triggers {
    pollSCM('H/5 * * * *')
}
```

Nous paramétrons une durée de vie de 90 jours à la build avant suppression :

```
options {
    disableConcurrentBuilds()
    buildDiscarder(logRotator(numToKeepStr: '30', daysToKeepStr: '90'))
}
```

Durant le build :

```
stages {

    stage('Clean and checkout project') {
        steps{
            deleteDir()
            checkout(changelog: false, scm: scm)
        }
    }

    stage('Build') {
        steps{
            sh "mvn package"
        }
    }

    stage('Post-Build') {
        steps {
            sh 'mvn sonar:sonar'
        }
    }

}
```

Nous aurons une première étape de nettoyage du répertoire de build de Jenkins.

Ensuite nous effectuerons la build grâce à la commande « **mvn package** » qui permettra également d'exécuter tous les tests.

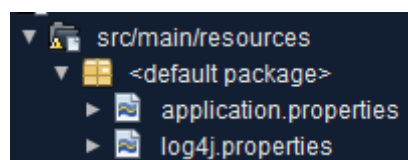
Après la build, nous enverrons le code sur Sonarqube

E- LOG4J

Pour ajouter Log4J, il faudra ajouter dans les dépendances, celle de log4j :

```
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.16</version>
</dependency>
```

Ensuite, pour configurer Log4J, nous aurons besoin de créer un fichier nommé « log4j.properties » (fichier à créer dans src/main/resources) dans lequel nous inscrivons les propriétés suivantes :



```
log4j.rootLogger=DEBUG, stdout, R
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{dd-MM-yyyy HH:mm:ss,SSS} %-5p (%F:%L) - %m%n
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.File=exemple.log
log4j.appender.R.File.MaxFileSize=100KB
log4j.appender.R.MaxBackupIndex=1
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{dd-MM-yyyy HH:mm:ss,SSS} %-5p (%F:%L) - %m%n
```

Utilisation de la sortie console et fichier et avons défini que tous les messages appartenant au niveau ou étant de niveau supérieur à DEBUG seront affichés

Sortie console

Sortie dans un fichier nommé "exemple.log" avec une taille limite de 100KB

28/11/2019 05:41:40,234 ERROR (classe:numéro de ligne) - message d'erreur

30-12-2019 15:10:19,235 ERROR (Log4PropertiesConfigurationExemple.java:21) - test

Pour chaque utilisation dans une classe, il faudra créer un membre static « logger » qui sera propre à la classe où il sera utilisé !

```
@SpringBootApplication
public class FutureTrackingApplication {

    static Logger logger = Logger.getLogger(FutureTrackingApplication.class);

    public static void main(String[] args) {
        logger.info("Test de mon logger");
        SpringApplication.run(FutureTrackingApplication.class, args);
    }
}

Building futureTracking 0.0.1-SNAPSHOT

30-12-2019 15:18:51,245 INFO (FutureTrackingApplication.java:13) - Test de mon logger
```

Nom de la classe courante

Et résultat avec le fichier.log :

```
30-12-2019 15:18:51,245 INFO (FutureTrackingApplication.java:13) - Test de mon logger
```

- futureTracking
 - .mvn
 - src
 - target
 - .gitignore
 - HELP.md
 - Jenkinsfile
 - README.md
 - exemple.log
 - futureTracking.log

F- SONARQUBE

Afin de pouvoir utiliser la commande « mvn sonar : sonar » il faudra paramétrer SonarQube dans le projet Maven, pour cela nous nous servirons du fichier « **pom.xml** » et plus précisément de la section « **properties** »

```
<sonar.projectKey>dis.futureTracking</sonar.projectKey>
<sonar.host.url>http://192.168.56.101:9000</sonar.host.url>
<sonar.login>76481c978fa48a86960b39d30d54cd9f9ac382a7</sonar.login>
```

Adresse VM ou se trouve docker + port de sortie de Sonarqube
token créé dans sonarqube

Localisation des fichiers tests :

```
<sonar.tests>src/test</sonar.tests>
<sonar.junit.reportsPath>target/surefire-reports</sonar.junit.reportsPath>
<sonar.surefire.reportsPath>target/surefire-reports</sonar.surefire.reportsPath>
```

Ici pour les tests je souhaiterai utiliser **junit** et **maven-surefire** ainsi que d'autres outils en complément (utilisation du clean de mvn, etc...), il va falloir inclure d'autres plugins et dépendances pour cela :

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

```
<plugin>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>2.22.1</version>
  <configuration>
    <argLine>${argLine} -Xmx4096m -XX:MaxPermSize=512M ${itCoverageAgent}</argLine>
  </configuration>
</plugin>
```

Utilisation du scanner de sonarqube :

```
<plugin>
  <groupId>org.sonarsource.scanner.maven</groupId>
  <artifactId>sonar-maven-plugin</artifactId>
  <version>3.6.0.1398</version>
</plugin>
```

Je voudrai également avoir la couverture de code, pour cela il sera utile d'utiliser « **jacoco** » qui permet le calcul du rapport de couverture de code que Sonarqube ne calcule pas !

```
<sonar.jacoco.reportPath>target/jacoco.exec</sonar.jacoco.reportPath>
<sonar.binaries>target/classes</sonar.binaries>
<sonar.java.coveragePlugin>jacoco</sonar.java.coveragePlugin>
<sonar.core.codeCoveragePlugin>jacoco</sonar.core.codeCoveragePlugin>
<sonar.jacoco.reportPath>target/jacoco.exec</sonar.jacoco.reportPath>
```

```

<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.7.2.201409121644</version>
  <configuration>
    <append>true</append>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <execution>
      <id>post-unit-test</id>
      <phase>test</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>


```

Dans l'installation du plugin maven-surefire, il ne faut pas oublier la ligne « **argLine** », car cela peut complètement gêner le fonctionnement de jacoco (je n'ai pas encore trouvé la réponse) il me semble qu'il s'agit sans doute de la manière dont sont écrits les résultats !

☆ [futureTracking](#)

0  Bugs

0  Vulnerabilities

20  Code Smells

 15.2%
Coverage

 0.0%
Duplications

Last analysis: January 2, 2020, 1:26 PM

459  Java, XML

G- JASPER

Pour installer le plugin, il suffira d'installer dans les dépendances le code suivant :

```

<dependency>
  <groupId>net.sf.jasperreports</groupId>
  <artifactId>jasperreports</artifactId>
  <version>6.9.0</version>
  <type>jar</type>
</dependency>

```

Il faudra dans un premier temps, déterminer les données que nous voulons utiliser lors de la génération d'un fichier PDF pour extraire une recette.

Voici un fichier JSON qui représente l'architecture de l'objet d'une recette

```

{
  "id": 1,
  "name": "Jeanne\u0027s timpani",
  "category": 0,
  "description": "",
  "ingredients": "4 slices of smoked salmon, 2 zucchini, 3 eggs, 10 cl thick fresh cream, 1 small garlic clove, 1 tablespoon olive",
  "person": 4,
  "duration": 18,
  "level": 1,
  "url": "https://assets.afcdn.com/recipe/20140921/15292_w800h600c1cx1224cy1224.jpg"
}

```

En suivant ce tutoriel : <https://www.youtube.com/watch?v=UG1RU393FtE>, il sera possible de générer un fichier que nous nommerons « jasperReportFormat.jrxml » qui contiendra les lois propres à la manière de rédiger le rapport effectué par Jasper y compris le design du fichier PDF depuis le fichier JSON donné ici en exemple.

Ainsi, en suivant ce tutoriel, nous devrions obtenir le design suivant :

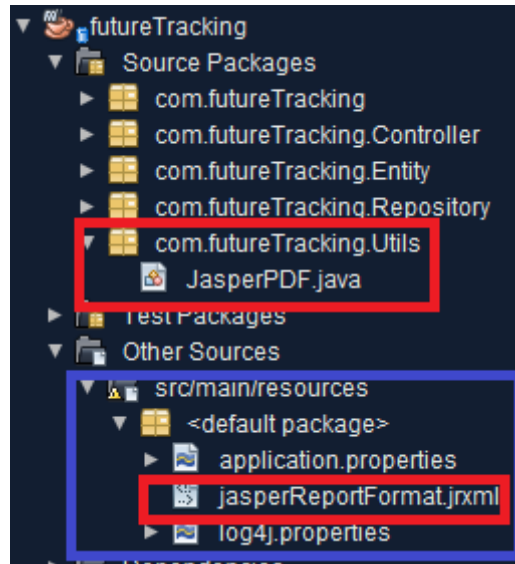
<div> <div>PDF REPORT ON RECIPE</div> <div> <div>\$F{id}</div> <div>\$F{name}</div> </div> </div>							
En-tête de Page							
NAME	CATEGORY	DESCRIPTION	INGREDIENTS	PERSON	DURATION	LEVEL	URL
\$F{name}	\$F{category}	\$F{description}	\$F{ingredients}	\$F{person}	\$F{duration}	\$F{level}	\$F{url}
Detail 1							
Pied de Colonne							

Et qui une fois compilé, donnera le fichier « *.jrxml » (extrait du fichier capturé) suivant :

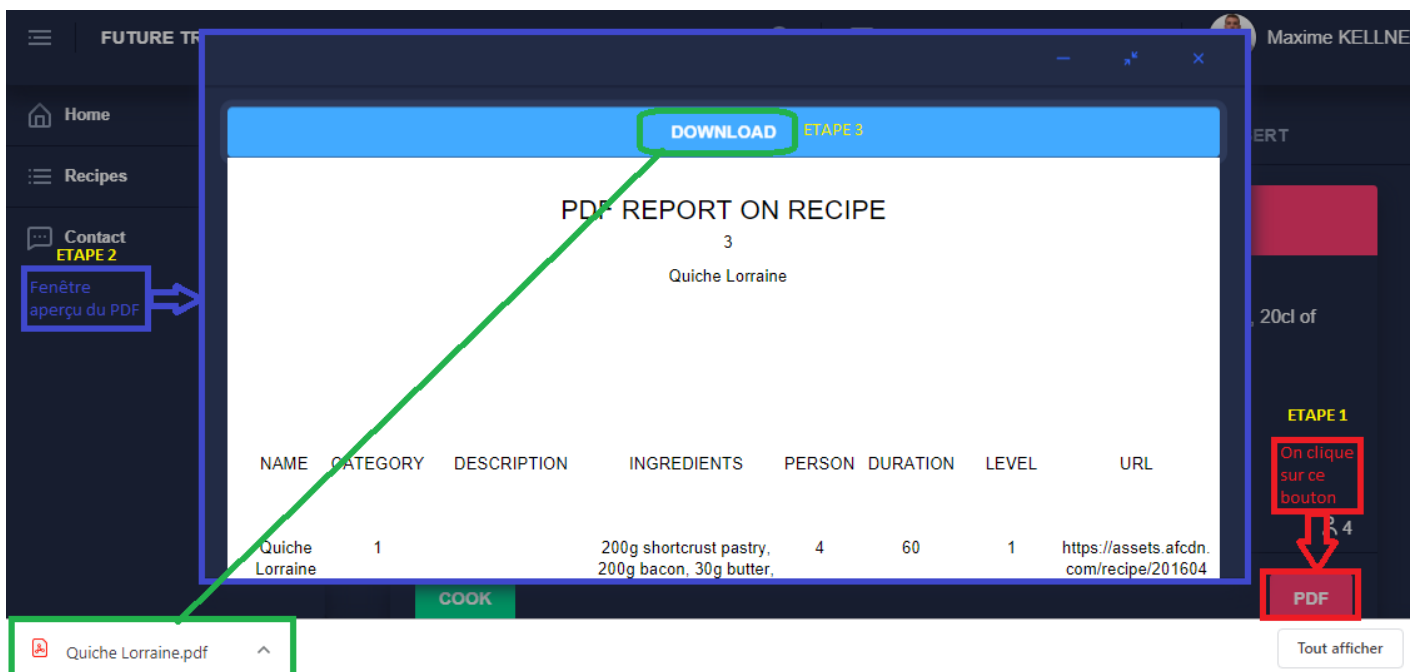
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created with JasperSoft Studio version 6.11.0.final using JasperReports Library version 6.11.0 -->
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports http://jasperreports.sourceforge.net/jasperreports.xsd">
  <property name="com.jaspersoft.studio.data.defaultdataadapter" value="JsonFileDataAdapter"/>
  <queryString language="json">
    <![CDATA[id]]>
  </queryString>
  <field name="id" class="java.lang.String">
    <property name="net.sf.jasperreports.json.field.expression" value="id"/>
    <fieldDescription><![CDATA[id]]></fieldDescription>
  </field>
  <field name="name" class="java.lang.String">
    <property name="net.sf.jasperreports.json.field.expression" value="name"/>
    <fieldDescription><![CDATA[name]]></fieldDescription>
  </field>
  <field name="category" class="java.lang.String">
    <property name="net.sf.jasperreports.json.field.expression" value="category"/>
    <fieldDescription><![CDATA[category]]></fieldDescription>
  </field>
  <field name="description" class="java.lang.String">
    <property name="net.sf.jasperreports.json.field.expression" value="description"/>
    <fieldDescription><![CDATA[description]]></fieldDescription>
  </field>
  <field name="ingredients" class="java.lang.String">
    <property name="net.sf.jasperreports.json.field.expression" value="ingredients"/>
    <fieldDescription><![CDATA[ingredients]]></fieldDescription>
  </field>
  <field name="person" class="java.lang.String">
    <property name="net.sf.jasperreports.json.field.expression" value="person"/>
    <fieldDescription><![CDATA[person]]></fieldDescription>
  </field>
  <field name="duration" class="java.lang.String">
    <property name="net.sf.jasperreports.json.field.expression" value="duration"/>
    <fieldDescription><![CDATA[duration]]></fieldDescription>
  </field>
  <field name="level" class="java.lang.String">
    <property name="net.sf.jasperreports.json.field.expression" value="level"/>
    <fieldDescription><![CDATA[level]]></fieldDescription>
  </field>
</jasperReport>
```

Ce fichier contient une première section qui présente les données que nous allons fournir et dans la deuxième section il s'agira du formatage (design, police, etc...) du fichier PDF.

Le format de notre fichier « jasperReportFormat.jrxml » sera stocké en tant que ressource du projet afin d'être utilisé :



Par conséquent, nous obtiendrons en appuyant sur le bouton ci-dessous :



L'application client fait en effet appel à l'adresse :

« 192.168.56.101 :8181/getRecipes/ID_Recipe_current (ici c'est 3) »

Grâce au mapping nous pourrons par la suite appeler la méthode **RecipeIdExportToPDF**(idRecipe : 3) située dans les webservices qui nous permettra de :

- 1- **Récupérer les informations** de la recette 3
- 2- **Construire un rapport PDF** via jasper (JasperPrint) depuis les informations récupérées et le fichier « jasperReportFormat.jrxml » pour la construction du PDF (appel de la méthode ExportPDF(String JSONObject // informations recettes 3).

ExportPDF (String JSONObject) :

```
public static void ExportPDF(String JSONObject) {  
  
    logger.debug("Appel de la méthode: ExportPDF(" + JSONObject + ")");  
  
    try {  
        try {  
            InputStream recipeReportStream;  
            recipeReportStream = getResourceAsStream("/jasperReportFormat.jrxml");  
            JasperReport jasperReport = JasperCompileManager.compileReport(recipeReportStream);  
  
            //Convert json string to byte array.  
            ByteArrayInputStream jsonDataStream = new ByteArrayInputStream(JSONObject.getBytes());  
  
            //Create json datasource from json stream  
            JsonDataSource ds = new JsonDataSource(jsonDataStream);  
  
            //Create HashMap to add report parameters  
            Map parameters = new HashMap();  
  
            //Add title parameter. Make sure the key is same name as what you named the parameter in jasper report.  
            parameters.put("title", "ReportRecipeJasperPDF");  
  
            //Create Jasper Print object passing report, parameter json data source.  
            JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, parameters, ds);  
  
            //Export and save pdf to file  
            JasperExportManager.exportReportToPdfFile(jasperPrint, System.getProperty("java.io.tmpdir") + "/ReportRecipeJasperPDF");  
  
        } catch (JRException ex) {  
            throw new RuntimeException(ex);  
        } catch (Exception ex) {  
            throw new RuntimeException(ex);  
        }  
    } catch (RuntimeException ex) {  
        throw new RuntimeException(ex);  
    }  
}
```

- 3- **Créer une ressource qui peut être transmise** dans une réponse à requête http GET (appel de la méthode downloadPDF())

downloadPDF()

```
public static ByteArrayResource downloadReport() throws IOException {  
  
    logger.debug("Appel de la méthode: downloadReport()");  
  
    Path path = Paths.get(System.getProperty("java.io.tmpdir") + "/ReportRecipeJasperPDF");  
  
    ByteArrayResource resource = new ByteArrayResource(Files.readAllBytes(path));  
  
    return resource;  
}
```

- 4- Ensuite, nous allons **construire un objet ResponseEntity** qui permettra la transmission du fichier PDF en fournissant directement le type des ressources qui sont transmises (on crée un header pour cela)

Vous trouverez ci-dessous la méthode **RecipeIdExportToPDF** :

```
@ResponseBody
@GetMapping("/getRecipes/{idRecipe}")
public ResponseEntity<Resource> RecipeByIdExportToPDF(@PathVariable @NotNull @DecimalMin("0") Integer idRecipe) throws IOException {

    logger.debug("Appel de la méthode: RecipeByIdExportToPDF(Integer idRecipe) via url: /getRecipes/" + idRecipe);

    Optional<Recipe> recipesIterable = this.recipeRepository.findById(idRecipe);

    GsonBuilder gsonBuilder = new GsonBuilder();
    Gson gson = gsonBuilder.create();

    Recipe recipe = new Recipe();

    if (recipesIterable.isPresent()) {
        recipe = recipesIterable.get();
    }

    String JSONObject = gson.toJson(recipe);

    JasperPDF.ExportPDF(JSONObject);

    HttpHeaders headers = new HttpHeaders();
    headers.add("Cache-Control", "no-cache, no-store, must-revalidate");
    headers.add("Pragma", "no-cache");
    headers.add("Expires", "0");
    headers.add("Expires", "0");
    headers.add("Content-Type", "application/pdf");
    headers.add("Content-Disposition", "attachment; filename=" + recipe.getName() + ".pdf");

    ByteArrayResource resource = JasperPDF.downloadReport();

    File file = new File(System.getProperty("java.io.tmpdir") + "/ReportRecipeJasperPDF");

    return ResponseEntity.ok()
        .headers(headers)
        .contentType(MediaType.parseMediaType("application/pdf"))
        .body(resource);
}
```

JASPER REPORT EST PRÊT !

H- HIBERNATE

Comme précédemment expliqué, la configuration de Hibernate s'est réalisé dans le fichier **application.properties et persistence.xml** qui se trouve dans le répertoire ressource du projet.

```
1 spring.datasource.url=jdbc:mysql://192.168.56.101:3306/futuretracking
2 spring.datasource.connectionProperties=useUnicode=true;characterEncoding=utf-8;
3 spring.datasource.username=future
4 spring.datasource.password=future
5
6 spring.datasource.driverclass = com.mysql.jdbc.Driver
7 spring.jpa.show-sql=true
8 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
9 spring.jpa.hibernate.ddl-auto = none
10 spring.jpa.properties.hibernate.current_session_context_class=org.springframework.orm.hibernate5.SpringSessionContext
11
12 server.port=8181
```

Utilisateur créé pour la gestion de la BDD

Adresse de la VM contenant le docker sql + indication du port de redirection qui a été choisi pour mysql + choix de la BD créée précédemment

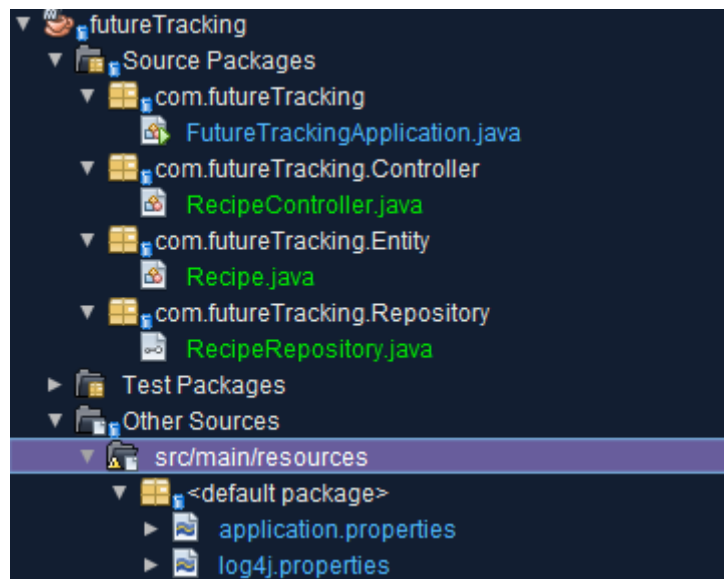
Les webservice seront accessibles sur le port 8181 de la machine

La dépendance nécessaire pour la gestion de la classe driver :

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
```

Maintenant que nous avons tous les outils, il nous faut faire du ORM avec Hibernate pour récupérer par exemple toutes les recettes stockées en BDD.

Voici l'architecture de notre application :



Dans le package « Entity » se trouve la classe « Recipe » qui prend pour architecture celle de l'objet qui le représente en BDD.

Dans le package « Repository », nous allons développer une interface « Repository » pour l'objet « Recipe » qui nous permettra grâce à l'héritage de la classe « CrudRepository<Recipe, Integer> » de récupérer les méthodes CRUD génériques (Create, Read, Update, Delete).

Dans le package « Controller », nous créerons une classe qui nous permettra de faire le mapping sur l'appel d'une méthode qui se servira de l'interface « RecipeRepository » afin d'utiliser les méthodes génériques.

Pour la classe Entity, il ne faudra pas oublier les annotations qui permettent de spécifier ce qu'une classe représente.

Est-ce un Repository (@Repository) ? Est-ce une classe liée à une table (@Entity / @Table(name = « «RECIPE » ») / @Id et @GeneratedValue / @Column(name= « nomcolonne » / pas de cardinalités ici donc pas besoin de @OneToMany, etc...)

Nous avons en **BDD** :

RECIPE	
id	INT AUTO_INCREMENT PRIMARY KEY
category	INT NOT NULL
name	VARCHAR(255) NOT NULL
description	VARCHAR(255) NOT NULL
ingredients	VARCHAR(255) NOT NULL
person	INT NOT NULL
duration	INT NOT NULL
level	INT NOT NULL
url	VARCHAR(255) NOT NULL

Nous avons dans **ENTITY** :

```
21 @Entity
22 @Table(name = "RECIPE")
23 public class Recipe implements Serializable {
24     static Logger logger = Logger.getLogger(Recipe.class)
25
26     @Id
27     @GeneratedValue
28     @Column(name = "id_recipe")
29     private int id;
30
31     @Column(name = "name")
32     private String name;
33
34     @Column(name = "category")
35     private int category;
36
37     @Column(name = "description")
38     private String description;
39
40     @Column(name = "ingredients")
41     private String ingredients;
42
43     @Column(name = "person")
44     private int person;
45
46     @Column(name = "duration")
47     private int duration;
48
49     @Column(name = "level")
50     private int level;
51
52     @Column(name = "url")
53     private String url;
54
55     public Recipe() {
56     }
57
58     public int getId() { ...4 lines }
59
60     public void setId(int id) {
61         logger.debug("setId(int id)");
62         this.id = id;
63     }
64
65     public String getName() {
66         logger.debug("getName()");
67         return name;
68     }
69
70     public void setName(String name) {
71         logger.debug("setName(String name)");
72         this.name = name;
73     }
74
75     public int getCategory() { ...4 lines }
76
77     public void setCategory(int category) { ...4 lines }
```

Nous avons dans **REPOSITORY** :

```
6 package com.futureTracking.Repository;
7
8 import com.futureTracking.Entity.Recipe;
9 import org.springframework.data.repository.CrudRepository;
10
11 /**
12  *
13  * @author maxim
14  */
15 public interface RecipeRepository extends CrudRepository<Recipe, Integer> {
16 }
17
```

Nous avons dans **CONTROLLER** :

```

24 @RestController
25 public class RecipeController {
26
27     static Logger logger = Logger.getLogger(RecipeController.class);
28
29     @Autowired
30     private RecipeRepository recipeRepository;
31
32     @ResponseBody
33     @GetMapping("/getRecipes") Mapping pour appel de la méthode showAllRecipes
34     public String showAllRecipes() {
35
36         logger.debug("Appel de la méthode: showAllRecipes() via url: /getRecipes"); 1 LOG
37
38         List<Recipe> recipes = new ArrayList<>();
39         Iterable<Recipe> recipesIterable = this.recipeRepository.findAll();
40
41         recipesIterable.forEach(recipes::add);
42
43         GsonBuilder gsonBuilder = new GsonBuilder();
44         Gson gson = gsonBuilder.create();
45
46         String JSONObject = gson.toJson(recipes);
47
48         return JSONObject; On retourne ici, un JSON pour une manipulation plus agréable par le client des données
49     }
50 }
51

```

Et enfin, nous avons dans le package par défaut, la main :

```

8  @EnableAutoConfiguration
9  @SpringBootApplication
10 public class FutureTrackingApplication {
11
12     static Logger logger = Logger.getLogger(FutureTrackingApplication.class);
13
14     public static void main(String[] args) {
15         logger.info("Démarrage de l'application Future Tracking BackEnd");
16         SpringApplication.run(FutureTrackingApplication.class, args);
17     }
18
19 }

```

Script des données contenu en BDD :

```
INSERT INTO recipe VALUES (1, "Jeanne's timpani", 0, "", "4 slices of smoked salmon, 2 zucchini, 3 eggs, 10 cl thick crepe", 1);
```

```
INSERT INTO recipe VALUES (2, "Carpaccio of scallops with lemon caviar with lemon caviar", 0, "", "12 large scallops, 3 l", 1);
```

```
INSERT INTO recipe VALUES (3, "Quiche Lorraine", 1, "", "200g shortcrust pastry, 200g bacon, 30g butter, 3 eggs, 20cl of", 1);
```

```
INSERT INTO recipe VALUES (4, "Roast beef in the oven", 1, "", "400g barded beef roast, 1 garlic clove, 2 tablespoons oli", 1);
```

```
INSERT INTO recipe VALUES (5, "Potato gratin", 1, "", "1kg of potato, 60g grated cheese, 25cl of fresh cream, 30g butter,", 1);
```

```
INSERT INTO recipe VALUES (6, "Chocolate fondant", 2, "", "200g dark chocolate, 150g butter, 150g granulated sugar, 50g f", 1);
```

Résultat après appel du webservice :

```
[[{"id":1,"name":"Jeanne","weight":20,"category":"","description":"","ingredients":"4 slices of smoked salmon, 2 zucchini, 8 eggs, 10 cl thick fresh cream, 1 small garlic clove, 1 tablespoon olive oil, ml mint, salt, pepper","person":4,"level":18,"url":"https://assets.afcdn.com/recipe/20140921/15292_w800b60c1c1224cy1224.jpg"}, {"id":2,"name":"Carpaccio de scallops with lemon juice with lemon carving","category":"","description":"","ingredients":"12 large scallops, 3 lemongrass sticks, 5 cl olive oil, 1 lemon carving, 2 chervil branches","person":4,"duration":20,"level":1,"url":"https://resize-  
le.lndmedia.fr/618,619,face,filling,img_vl_plain_site_storage/images/uploads/repaille/ele-a-table/carpaccio-de-saint-jacques-375281189660275-2-fie-  
F1Carpaccio-de-saint-jacques-375281189660275-2-fie-1","id":3,"name":"Quiche Lorraine","category":"","description":"","ingredients":"200g shortcrust pastry, 200g bacon, 30g butter, 3 eggs, 20cl of fresh cream, 20cl milk, nutmeg, salt, pepper","person":4,"duration":60,"level":1,"url":"https://assets.afcdn.com/recipe/20160404/37706_w1024h768c1sc1500cy1000.jpg"}, {"id":4,"name":"Roast beef in the oven","category":"","description":"","ingredients":"400g barded beef roast, 1 garlic clove, 2 tablespoons olive oil, 12cl of water, 1 pepper, 1 salt, 1 thyme","person":2,"duration":45,"level":1,"url":"https://s3-eu-central-1.amazonaws.com/media.utoque.fr/img_w1536_h1024/recipes/images/roti-de-boeuf-aux-pommes-grainelle-et-blette-poelee/roti-de-boeuf-aux-thym-pommes-grainelle-et-blette-poelee-2"}, {"id":5,"name":"Potato gratin","category":"","description":"","ingredients":"1kg of potato, 60g grated cheese, 25cl of fresh cream, 50g butter, 1 garlic clove, pepper, salt","person":4,"duration":70,"level":1,"url":"https://assets.afcdn.com/recipe/20180123/377042_w1024h768c1sc1626cy1750c0x0c0b3250cy3500.jpg"}, {"id":6,"name":"Chocolate fondant","category":"","description":"","ingredients":"200g dark chocolate, 150g butter, 150g granulated sugar, 50g flour, 3 eggs, 20cl of fresh cream, 20cl milk, nutmeg, salt, pepper","person":4,"duration":40,"level":1,"url":"https://assets.tmeccoy.com/image/upload/t_web/67x639/img/recipe/rs/Assets/f8547f4-dfb2c-44c-847a-7aa136d1b8/Devantes-ba70846-905f-411-9245-6f611b61218.jpg"}]]
```

HTML TEST

Le petit bonus avec jacoco, c'est que celui-ci réalise déjà des rapports de test en HTML, tout comme maven-surefire-report-plugin qui permet également d'avoir des rapports HTML dès que dans sa configuration nous spécifions un fichier de sortie. La différence étant que sur jacoco, nous avons surtout des détails sur la couverture de code effectué et à la fois qui est passé au vert dans les tests.

Ainsi, en se rendant à la racine du projet, après avoir exécuter une première fois les tests, vous trouverez dans le répertoire suivant : « **futureTracking\target\site\jacoco** », un fichier **index.html** que je vous invite à ouvrir à l'aide de votre navigateur et qui vous permettra d'avoir un rapport sur tous les tests réalisés y compris le rapport de couverture de code.

1 - Jacoco

futureTracking

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.futureTracking.Controller	<div><div></div></div>	0 %	<div><div></div></div>	0 %	5	5	33	33	4	4	1	1
com.futureTracking.Entity	<div><div></div></div>	0 %		n/a	20	20	48	48	20	20	1	1
com.futureTracking.Utils	<div><div></div></div>	82 %		n/a	1	4	7	25	1	4	0	1
com.futureTracking	<div><div></div></div>	0 %		n/a	3	3	5	5	3	3	1	1
Total	303 of 401	24 %	2 of 2	0 %	29	32	93	111	28	31	3	4

com.futureTracking.Utils

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
JasperPDF	<div><div></div></div>	82 %		n/a	1	4	7	25	1	4	0	1
Total	21 of 119	82 %	0 of 0	n/a	1	4	7	25	1	4	0	1

JasperPDF

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
ExportPDF(String)	<div><div></div></div>	80 %		n/a	0	1	6	19	0	1
JasperPDF()	<div><div></div></div>	0 %		n/a	1	1	1	1	1	1
downloadReport()	<div><div></div></div>	100 %		n/a	0	1	0	4	0	1
static {...}	<div><div></div></div>	100 %		n/a	0	1	0	1	0	1
Total	21 of 119	82 %	0 of 0	n/a	1	4	7	25	1	4

2 – Surefire-report

Pom.xml :

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-report-plugin</artifactId>
      <version>3.0.0-M4</version>
      <configuration>
        <outputDirectory>${basedir}/target/newsite</outputDirectory>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-site-plugin</artifactId>
      <version>2.1</version>
      <configuration>
        <outputDirectory>${basedir}/target/newsite</outputDirectory>
      </configuration>
    </plugin>
  </plugins>
</reporting>
```

En ajoutant ce plugin, lors de l'exécution de la commande mvn site, nous allons avoir accès au rapport HTML suivant :

« NetBeansProjects » futureTracking » target » site

Rechercher dans : site

Nom	Modifié le	Type	Taille
css	02/01/2020 14:50	Dossier de fichiers	
images	02/01/2020 14:58	Dossier de fichiers	
jacoco	02/01/2020 14:57	Dossier de fichiers	
dependencies.html	02/01/2020 14:58	Chrome HTML Do...	152 Ko
dependency-info.html	02/01/2020 14:58	Chrome HTML Do...	5 Ko
dependency-management.html	02/01/2020 15:00	Chrome HTML Do...	382 Ko
index.html	02/01/2020 15:00	Chrome HTML Do...	4 Ko
licenses.html	02/01/2020 15:00	Chrome HTML Do...	4 Ko
plugin-management.html	02/01/2020 15:00	Chrome HTML Do...	10 Ko
plugins.html	02/01/2020 15:00	Chrome HTML Do...	7 Ko
project-info.html	02/01/2020 15:00	Chrome HTML Do...	6 Ko
project-reports.html	02/01/2020 15:00	Chrome HTML Do...	3 Ko
scm.html	02/01/2020 15:00	Chrome HTML Do...	5 Ko
summary.html	02/01/2020 15:00	Chrome HTML Do...	5 Ko
surefire-report.html	02/01/2020 14:58	Chrome HTML Do...	9 Ko
team.html	02/01/2020 15:00	Chrome HTML Do...	5 Ko

FICHIERS GENERES PAR "MVN SITE"


Et en exécutant surefire-report.html :

futureTracking

Last Published: 2020-01-02 | Version: 0.0.1-SNAPSHOT

futureTracking

Project Documentation
Project Information
Project Reports
Surefire Report

Built by: 

Surefire Report

Summary

Test Cases

[Summary] [Package List] [Test Cases]

RecipeTest

nameAnnotation	0.01
tableAnnotation	0.005
entity	0.001
durationAnnotation	0
ingredientsAnnotation	0
levelAnnotation	0.001
typeAnnotations	0.061
descriptionAnnotation	0
personAnnotation	0
urlAnnotation	0
idAnnotation	0.006
categoryAnnotation	0

JasperPDFTest

testDownloadReport	0.621
testExportPDF	3.618

Copyright © 2020. All rights reserved.

Pour résumer, afin d'avoir l'effet BackEnd avec une architecture N-tiers, le .jar généré par le projet FutureTracking-BackEnd » a été exécuté avec la commande « java -jar futureTracking-0.0.1-SNAPSHOT.jar » sur la machine virtuelle.

PROJET « FutureTracking-Client » (BRANCHE « MASTER » UNIQUEMENT)

A- APACHE TOMCAT – Migration du client Angular sur TOMCAT

Etape 1 : Télécharger et installer TOMCAT

Suivre le lien : <https://o7planning.org/fr/11583/installation-et-configuration-de-tomcat-server>

Etape 2 : Mettre notre client Angular sur TOMCAT

Il suffit tout simplement de réaliser un build de notre client avec la commande « ng build » et de copier toute la sortie du build dans le fichier www, le fichier de build est par défaut stocker dans le fichier « dist » qui se trouve à la racine du projet Angular.

Etape 3 : Accéder à votre projet

Pour cela, il suffit d'accéder via l'url : « **localhost : portTomcat/repertoireDeNotreSiteAngular** ».

B- Gestion de version (GITHUB)

- 1- Créer un répertoire « FutureTracking-Client »
- 2- Récupérer le lien de partage (dans mon cas : <https://github.com/MaxeeZ/FutureTracking>)
- 3- Réaliser exactement les mêmes manipulations que pour le projet « FutureTracking-BackEnd »

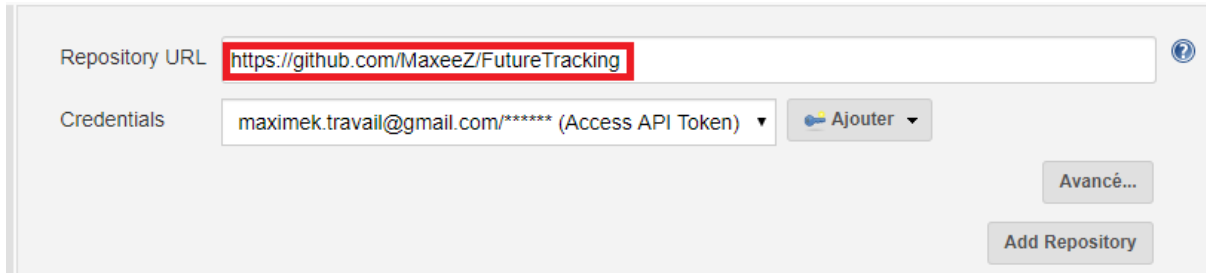
The screenshot shows the GitHub repository page for 'MaxeeZ / FutureTracking'. The repository is private and has 12 commits, 1 branch, 0 packages, and 0 releases. The latest commit is 'MaxeeZ Fix: Update url webservice API REST' from 44 minutes ago. The repository contains several files and folders, including .scannerwork, e2e, src, .editorconfig, .gitignore, JenkinsFile, README.md, angular.json, browserslist, karma.conf.js, package-lock.json, package.json, and sonar-project.properties. The repository is currently on the master branch.

File/Folder	Commit Message	Time Ago
.scannerwork	Fix: sonar-scanner properties url	4 days ago
e2e	initial commit	10 days ago
src	Fix: Update url webservice API REST	44 minutes ago
.editorconfig	initial commit	10 days ago
.gitignore	Feat: Link Client Side to WebServices	2 days ago
JenkinsFile	Fix: Update JenkinsFile	4 days ago
README.md	merge	6 days ago
angular.json	Initial Commit	6 days ago
browserslist	initial commit	10 days ago
karma.conf.js	initial commit	10 days ago
package-lock.json	Feat: Add download of jasper report PDF	5 hours ago
package.json	Feat: Add download of jasper report PDF	5 hours ago
sonar-project.properties	Fix: sonar-scanner properties url	4 days ago

C- Intégration du projet dans Jenkins depuis GitHub

L'intégration du projet client s'effectue exactement de la même manière que pour le projet « FutureTracking-BackEnd ».

Il faudra cependant changer le nom du pipeline par « FutureTracking-Client » et remplacer lien d'accès au fichier JenkinsFile par le lien suivant : « <https://github.com/MaxeeZ/FutureTracking> »



D- JENKINSFILE

Le Nous utiliserons la machine « maître » par défaut et aucun slave pour l'exécution de nos builds :

```
agent {  
    label 'master'  
}
```

Nous vérifierons tous les 5 minutes le répertoires GitHub afin de réaliser de nouvelles builds en cas d'évolution / modification sur le répertoire :

```
triggers {  
    pollSCM('H/5 * * * *')  
}
```

Nous paramétrons une durée de vie de 90 jours à la build avant suppression :

```
options {  
    disableConcurrentBuilds()  
    buildDiscarder(logRotator(numToKeepStr: '30', daysToKeepStr: '90'))  
}
```

Cette fois-ci, j'ai émis la possibilité de configurer le nom de l'artefact qui sera générée à chaque lancement de build :

```
parameters {  
    string(defaultValue: 'app', description: 'Choose the name of repository where going to be store artefact file', name: 'artefact_name', trim: ·  
}
```


Durant le build :

```
stages {

    stage('Clean and checkout project') {
        steps{
            deleteDir()
            checkout(changelog: false, scm: scm)
        }
    }

    stage('Build') {
        steps{
            sh '''npm install
            ng build --prod

            cd dist
            zip -r ${artefact_name}.zip NebularApp'''
        }
    }

    stage('Post-Build') {
        steps {
            sh 'sonar-scanner'
        }
    }
}
```

Nous aurons une première étape de nettoyage du répertoire de build de Jenkins.

Ensuite nous effectuerons la build grâce à la commande « **npm build** » après avoir installé tous les packages nécessaires avec la commande « **npm install** ».

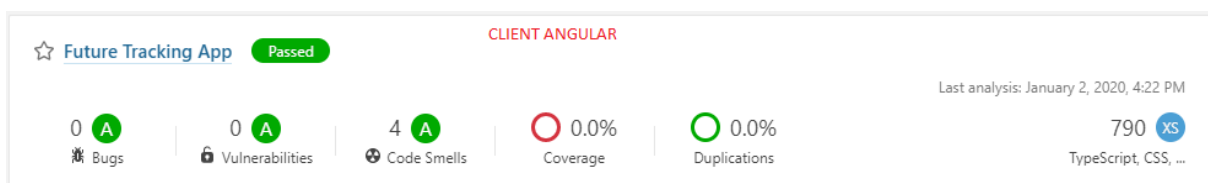
Ensuite, nous irons dans le fichier de sortie du build « **cd dist** » avant de construire l'artefact dont le nom aura été passé en paramètre !

Après la build, nous enverrons le code sur Sonarqube

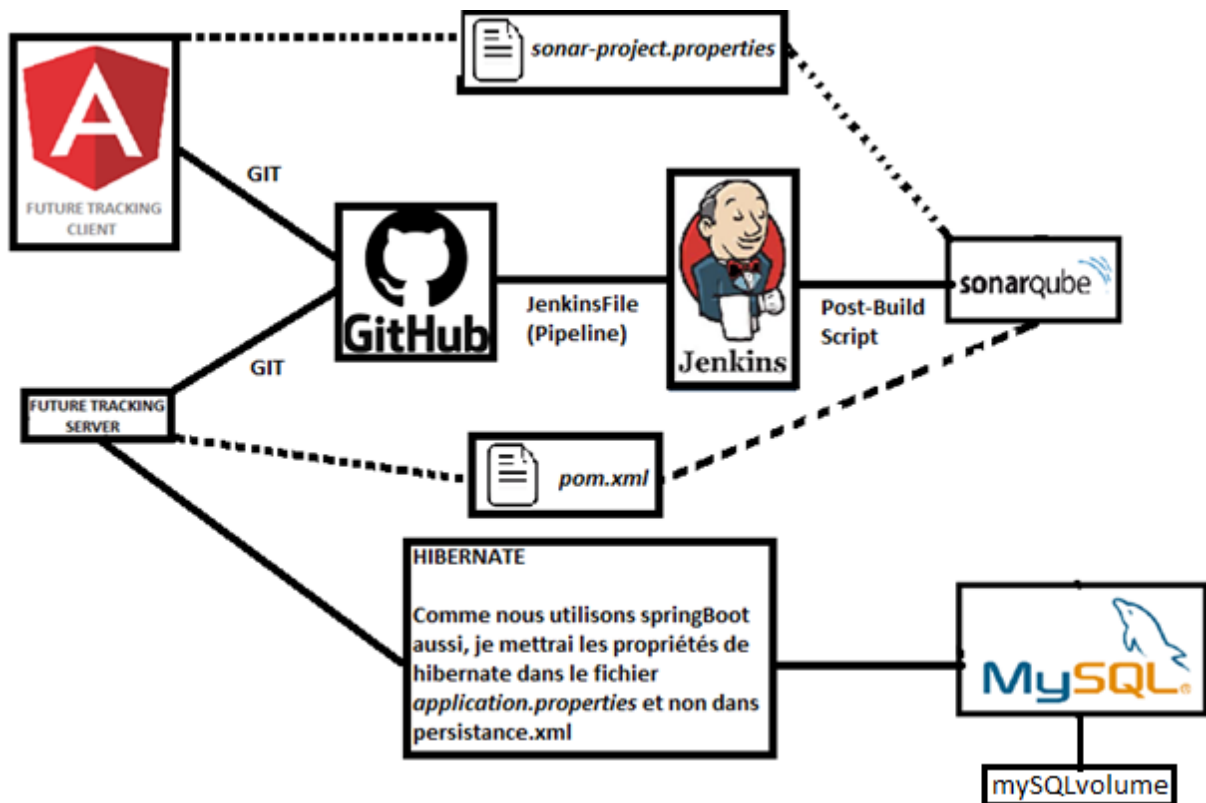
E – SONARQUBE

Afin de pouvoir utiliser la commande « sonar-scanner » il faudra paramétrer SonarQube dans le projet Angular, pour cela nous créerons un fichier « **sonar-project.properties** » que nous placerons à la racine du projet Angular.

```
sonar-project.properties
1 sonar.host.url=http://192.168.56.101:9000  Uri du docker se trouvant sur la VM au port 9000 (port de redirection de sonarqube)
2 sonar.projectKey=futureTracking-client-kellner
3 sonar.projectName=Future Tracking App  Nom du projet sur sonarQube
4 sonar.sources=src  Source des fichiers à prendre en compte
5 sonar.typescript.exclusions=**/node_modules/**,**/bower_components/**,**/*.spec.ts
6 sonar.login=76481c978fa48a86960b39d30d54cd9f9ac382a7  Token créé lors de la configuration de SonarQube
7 sonar.sourceEncoding=UTF-8
8 sonar.verbose=true
```



RESUME DE LA CONFIGURATION



Tous	+					
S	M	Nom du projet ↓	Dernier succès	Dernier échec	Dernière durée	
		FutureTracking-BackEnd	1 h 24 mn - #17	s. o.	2 mn 25 s	
		FutureTracking-Client	23 mn - #15	s. o.	6 mn 51 s	

Icône: [S](#) [M](#) [L](#)

[Légende](#)
[Atom feed pour tout](#)
[Atom feed de tous les échecs](#)
[Atom feed juste pour les dernières compilations](#)

[Future Tracking App](#)
Passed

CLIENT ANGULAR

Last analysis: January 2, 2020, 4:22 PM

0 Bugs

0 Vulnerabilities

4 Code Smells

0.0% Coverage

0.0% Duplications

790

TypeScript, CSS, ...

[futureTracking](#)
Passed

SERVEUR

Last analysis: January 2, 2020, 3:16 PM

0 Bugs

0 Vulnerabilities

20 Code Smells

15.2% Coverage

0.0% Duplications

479

Java, XML

GESTION DE PROJET