Config Files

Nous donnons ici en annexe les fichiers de configurations pour Gradle, l'application Java, Tomcat , Webpack, et babel.

Gradle

Gradle est un build automation tool qui permet de créer des archives java facilement et de gérer les packages dont dépend notre programme. Ainsi, il me permet d'importer toutes les classes et annotations de Spring, le framework qui me permet de gérer mon API REST, mais aussi de compiler avec mon IDE IntelliJ IDEA, de construire une archive jar, d'utiliser les classes d'auth0.

```
buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:2.0.1.RELEASE")
    }
}

apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'idea'
apply plugin: 'org.springframework.boot'
apply plugin: 'io.spring.dependency-management'

jar {
    baseName = 'gs-serving-web-content'
    version =  '0.1.0'
}

bootJar {
    launchScript()
}

repositories {
    mavenCentral()
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

dependencies {
    compile("org.springframework.boot:spring-boot-starter-web")
    compile("org.springframework.boot:spring-boot-starter-data-jpa")
    compile('org.postgresql:postgresql')
    testCompile('org.springframework.boot:spring-boot-starter-test')
    compile 'com.auth0:java-jwt:3.4.0'
    compile 'com.auth0:jwks-rsa:0.5.0'
}
```

Java application.properties

Le fichier application.properties me permet de gérer les constantes et les options nécessaires à mon application Spring pour fonctionner. Ainsi, j'y définis le port utilisé par Tomcat pour le serveur (443 pour SSL) avec l'option de Spring qui permet de faire fonctionner SSL ainsi que l'emplacement du certificat auto-signé. J'y définis aussi les constantes nécessaires pour faire appel aux services d'auth0, notamment la création de compte et la vérification de tokens par la clé présente sur leurs serveurs. J'y définis aussi l'adresse de la base de données postgreSQL ainsi que les credentials nécessaires à Spring pour s'y connecter. J'y définis enfin la configuration d'hibernate, ORM, pour faire la traduction entre objet et tableaux dans la base de données.

```
# Define a custom port instead of the default 8080
server.port=443
```

```
# Tell Spring Security (if used) to require requests over HTTPS
security.require-ssl=true

# The format used for the keystore
server.ssl.key-store-type=PKCS12
# The path to the keystore containing the certificate
server.ssl.key-store=classpath:keystore.p12
# The password used to generate the certificate
server.ssl.key-store-password=P0rkch0p42!
# The alias mapped to the certificate
server.ssl.key-alias=kineticKeystore


#Authentification Domain
auth0.ISSUER=https://kineticexpress.auth0.com
auth0.AUDIENCE=https://localhost
auth0.CLIENT_ID=7bug4gdhtEeo77gXDz0dBl1KBjVr5I1O
auth0.CLIENT_SECRET=A9_2PNqixCjxJ7hlGIJn8licpeiOX0-UTRiHcV7LNuCq3dPtHhYN9-_yQJ670Afc


# Details for our datasource
spring.datasource.url = jdbc:postgresql://kinetic.crkqlzp9ctvw.us-east-
2.rds.amazonaws.com:5432/kinetic
spring.datasource.username = kinetic_admin
spring.datasource.password = P0rkch0p42!

# Hibernate properties
spring.jpa.database-platform = org.hibernate.dialect.PostgreSQL94Dialect
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.hibernate.naming.implicit-strategy =
org.hibernate.boot.model.naming.ImplicitNamingStrategyJpaCompliantImpl
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.current_session_context_class=org.springframework.orm.hib
ernate5.SpringSessionContext

# Disable feature detection by this undocumented parameter. Check the
org.hibernate.engine.jdbc.internal.JdbcServiceImpl.configure method for more details.
spring.jpa.properties.hibernate.temp.use_jdbc_metadata_defaults: false
```

Tomcat

J'utilise Tomcat comme serveur pour Java afin de répondre aux requêtes https. La classe ConnectorConfig.java me permet de rediriger toutes les requêtes http vers des requêtes https.

```
package qinetic.configuration;

import org.springframework.boot.web.servlet.server.ServletWebServerFactory;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import org.apache.catalina.Context;
import org.apache.catalina.connector.Connector;
import org.apache.tomcat.util.descriptor.web.SecurityConstraint;
import org.apache.tomcat.util.descriptor.web.SecurityCollection;
import org.springframework.boot.web.embedded.tomcat.TomcatServletWebServerFactory;


@Configuration
public class ConnectorConfig {

    @Bean
    public ServletWebServerFactory servletContainer() {
        TomcatServletWebServerFactory tomcat = new TomcatServletWebServerFactory() {
            @Override
            protected void postProcessContext(Context context) {
                SecurityConstraint securityConstraint = new SecurityConstraint();
                securityConstraint.setUserConstraint("CONFIDENTIAL");
                SecurityCollection collection = new SecurityCollection();
```

```
                collection.addPattern("/*");
                securityConstraint.addCollection(collection);
                context.addConstraint(securityConstraint);
            }
        };
        tomcat.addAdditionalTomcatConnectors(redirectConnector());
        return tomcat;
    }

    private Connector redirectConnector() {
        Connector connector = new
Connector("org.apache.coyote.http11.Http11NioProtocol");
        connector.setScheme("http");
        connector.setPort(80);
        connector.setSecure(false);
        connector.setRedirectPort(443);
        return connector;
    }

}
```

Webpack

Webpack est un bundler me permettant de compiler mon application React eet de centraliser tous les modules js en un. Il me permet de charger sass, préprocesseur CSS, et de transformer ces fichiers en CSS. En faisant appel à Babel, il peut *transpiler* la syntaxe nouvelle d'EcmaScript 6 (Javascript) vers une syntaxe plus ancienne lisible par les anciens navigateurs.

```
var path = require('path');

module.exports = {
    mode: 'development',
    entry: './src/js/main.js',
    output: {
        filename: 'bundle.js',
        path: path.resolve(__dirname, 'dist')
    },
    module: {
        rules: [
            {
                test: /\.sass$/,
                use: [
                    {
                        loader: 'style-loader',
                    },

                    {
                        loader: 'css-loader',
                        options: {
                            sourceMap: true,
                            modules: true,
                            localIdentName: '[local]___[hash:base64:5]'
                        }
                    },
                    {
                        loader: "sass-loader" // compiles Sass to CSS
                    }
                ],
            },
            { test: /\.(png|woff|woff2|eot|ttf|svg)$/, loader: 'url-loader?limit=100000'
},
            { test: /\.(js|jsx)$/, exclude: /node_modules/, loader: "babel-loader" }
        ]
    }
};
```

Babel

Babel est un transpiler qui permet d'utiliser la syntaxe la plus récente d'ES 2017 mais aussi des fonctionnalités expérimentales du langages qui n'existe pas forcément encore dans la documentation actuelle.

```
/*
    ./.babelrc
*/
{
  "presets":[
    "env", "react", "stage-0"
  ]
}
```