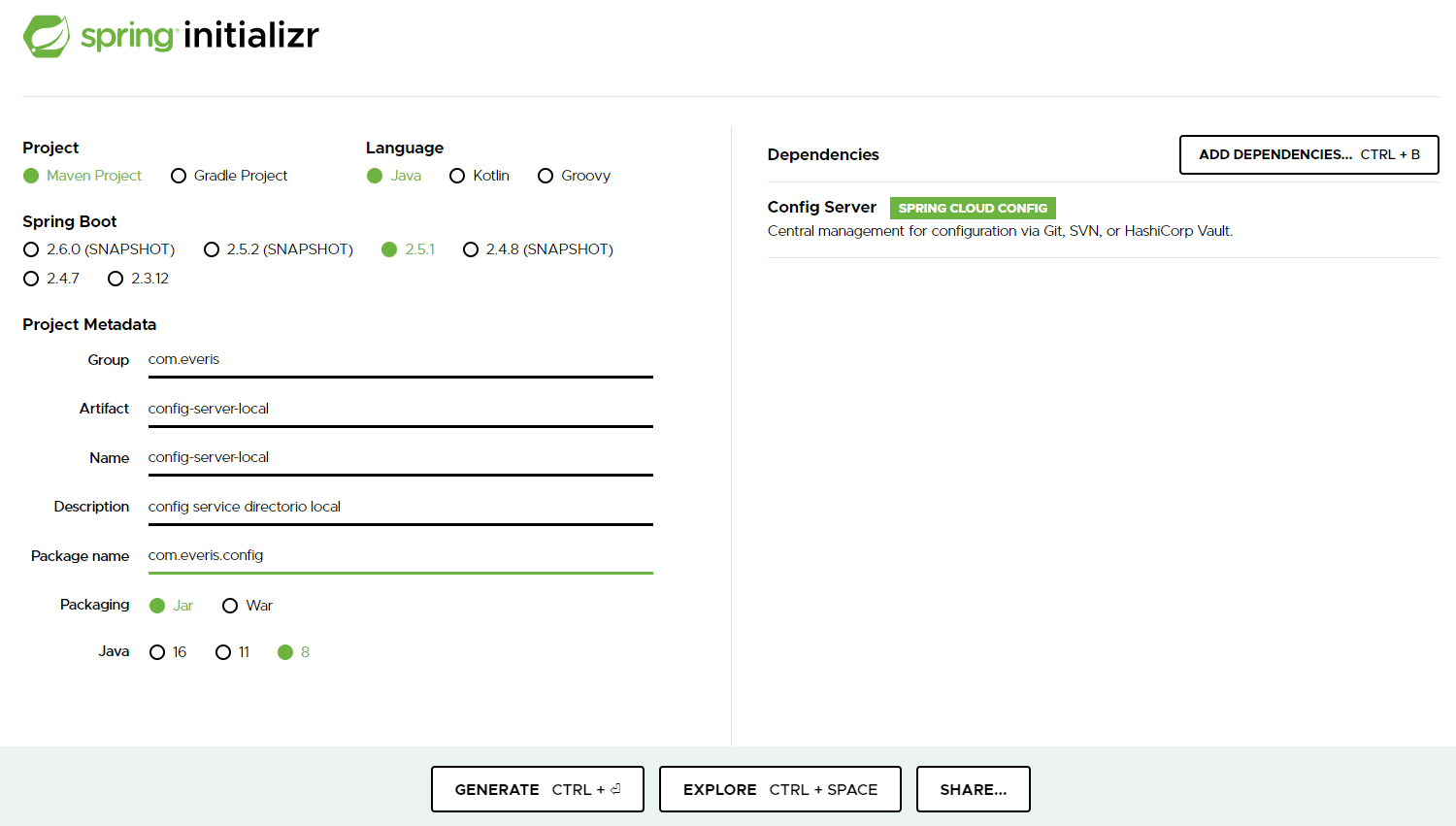
**Cloud-config-server**

* Es un Servidor de configuración centralizado ( en donde todas las configuraciones de nuestro microservicios se van a centralizar en nuestro proyecto)
* Jerarquía de configuración ( es decir a través de profile tu vas a definir tus configuraciones para diferentes entornos)
* Histórico de configuración (la configuración se puede registrar en un controlador de versiones) ya sea GitHub, GitLab, otros.
* Centralizar configuración de varios entornos ( a través de profile puedes configurar tu entorno de desarrollo, testing o producción)
* Recarga de valores de propiedades en caliente

Config-server : configuración con directorio local

**Configuración de Servidor**

1.- Generamos el proyecto :



El cual en el pom.xml se agrega la dependencia :

<dependency>

<groupId>org.springframework.cloud</groupId>

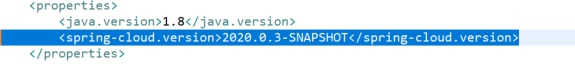
<artifactId>spring-cloud-config-server</artifactId>

</dependency>

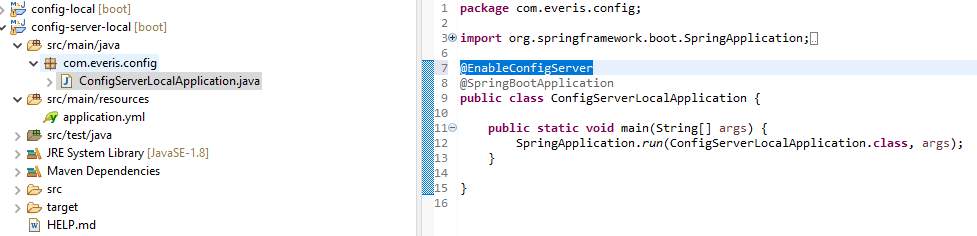
Como se puede apreciar se en el administrador de dependecias para cloud es por ello que debe agregarse la versión.



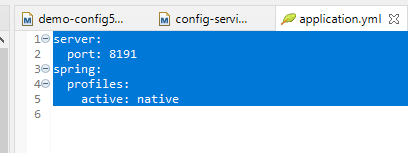
También se agrega una variable la versión en el caso no se tenga



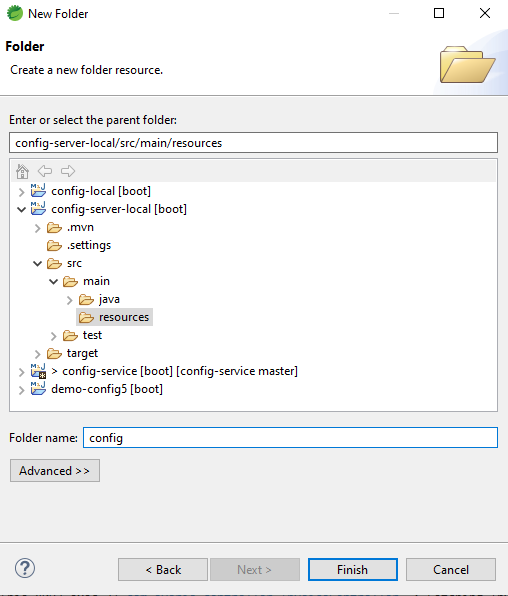
En la clase principal se agrega la etiqueta @EnableConfigServer para indicar que va ser el –servidor de configuración.



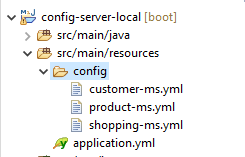
Luego vamos a nuestro archivo de configuración en **resources** y cambiamos el archivo application.properties por un application.yml y configuramos el puerto e indicamos donde se va ejecutar.



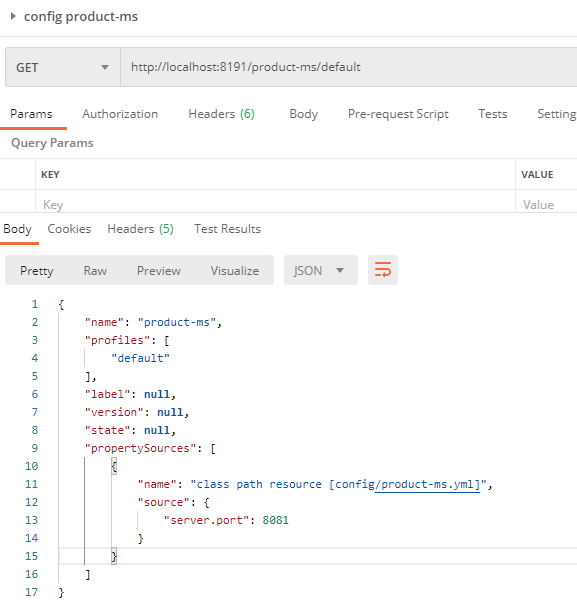
Se crea una carpeta llamada **config** en el el paquete resources



Y agregamos los archivos de configuración de los servicios: customer, product e Invoice en la carpeta config.

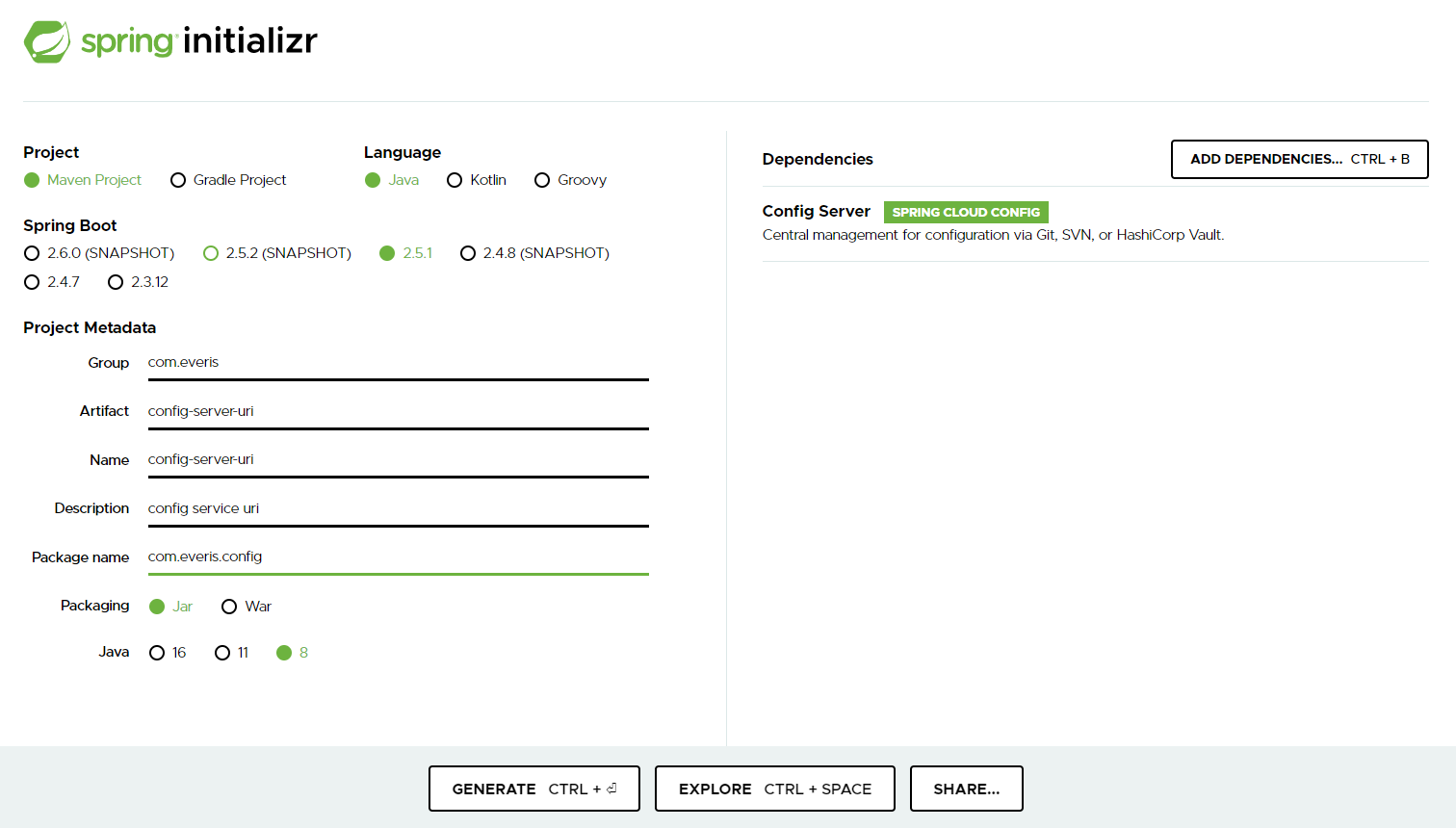


Probamos indicando el puerto del servidor y el nombre de los arhivos de config y obtendremos todas las propiedades del servicio.

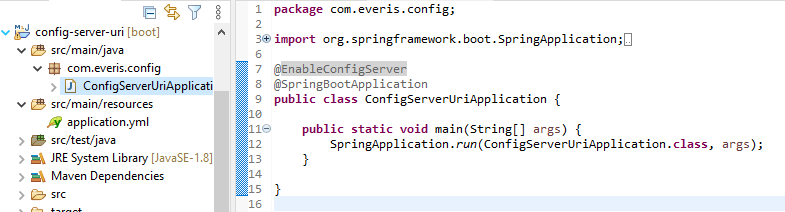


Config-server : configuración con url

**Configuración de Servidor**



En la clase principal se agrega la etiqueta @EnableConfigServer para indicar que va ser el –servidor de configuración.



Se configura el archivo application.yml:

server:

port: 8191

spring:

cloud:

config:

server:

git:

uri: https://github.com/avillane/repositorioprueba.git

search-paths: ConfigData

username: ${GIT\_USER}

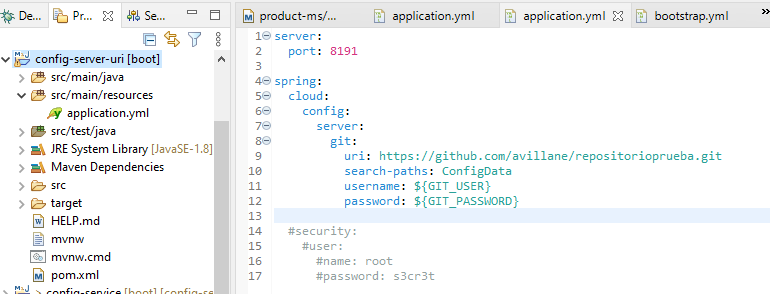
password: ${GIT\_PASSWORD}

#security:

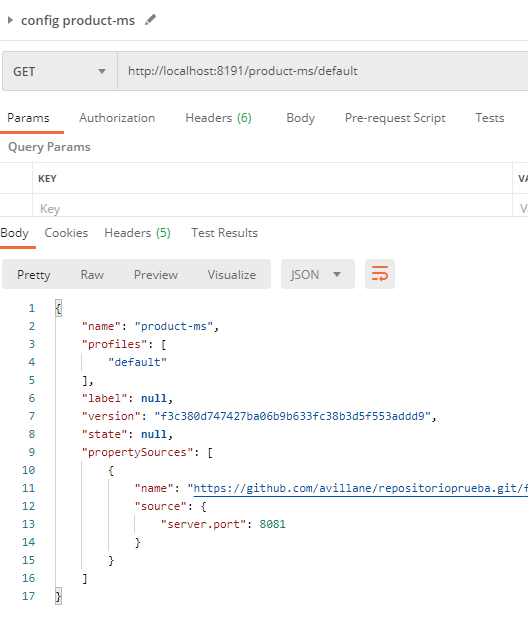
#user:

#name: root

#password: s3cr3t



Vamos a probar la configuración del servidor, para los servicios



**Configuración del cliente**

1. Configuración en el pom, se debe agregar la dependencia:

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-config</artifactId>

</dependency>

Y también el administrador de dependencia

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<repositories>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

</repository>

</repositories>

Y la versión del spring cloud :

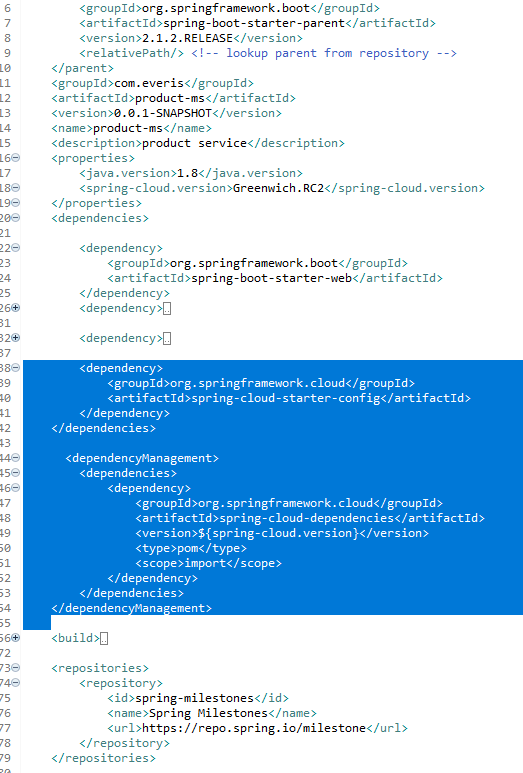
<properties>

<java.version>1.8</java.version>

<spring-cloud.version> Greenwich.RC2</spring-cloud.version>

</properties>

En resumen:



En el archivo application.yml, el nombre del aplicación a conectarse

spring:

application:

name: product-ms

En el archivo bootstrap.yml, el nombre

spring:

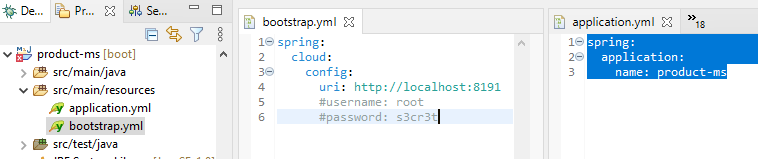
cloud:

config:

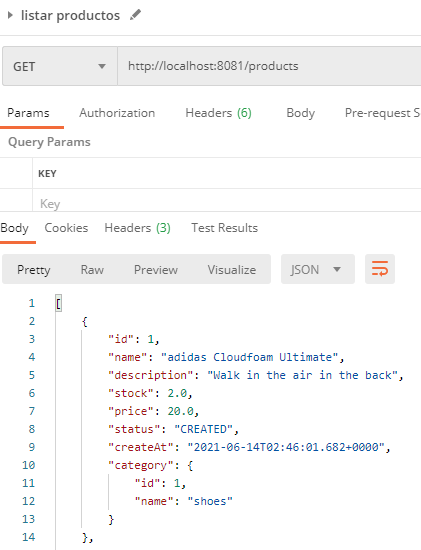
uri: http://localhost:8191

#username: root

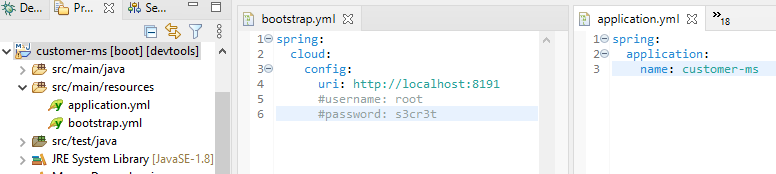
#password: s3cr3t

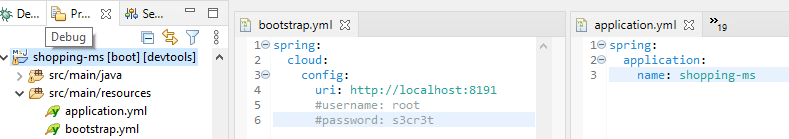


Ahora probamos el servicio product-ms, usando la config server.



Y para los servicios restantes, sólo cambias el name de aplicación por: customer-ms y shopping-ms





Y configurar el archivo xml, como el servicio product-ms.

