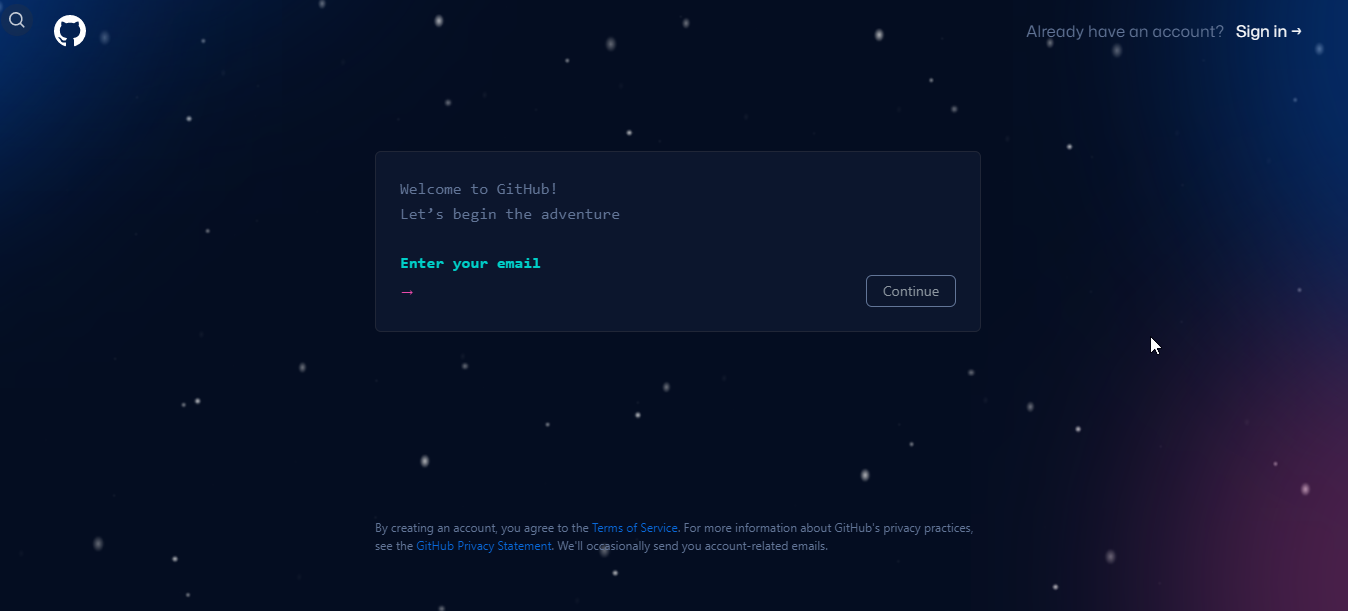
API REST CON CONEXIÓN A OPEN WEATHER

1.- Creamos el repositorio en github, entramos a la siguiente liga:

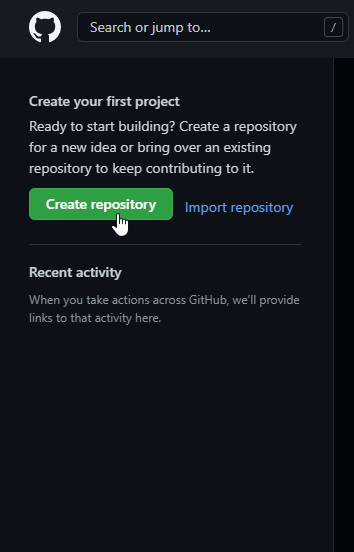
<https://github.com/signup?ref_cta=Sign+up&ref_loc=docs+header&ref_page=docs>

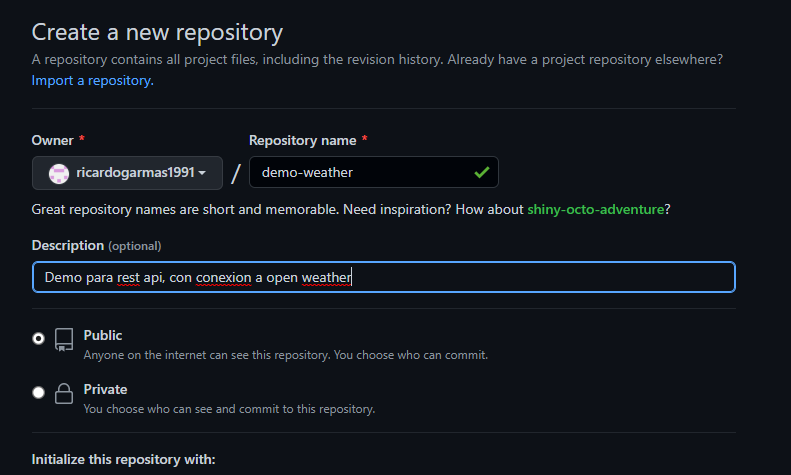


2.- ingresamos nuestro correo si es que no tenemos una cuenta, así como también nos pedirá un password, y un usuario.

3.- seguimos todos los pasos que nos piden, para verificar la cuenta, así como seleccionar si la cuenta es de estudiante o profesor.

4.- cuando este la cuenta creada y verificada, creamos un repositorio:

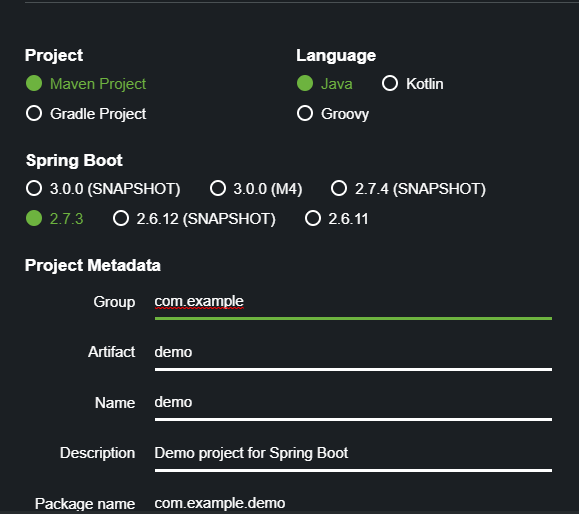




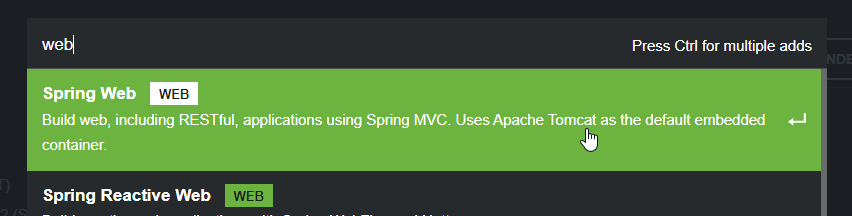
5.- Teniendo el Branch, comenzamos a crear el proyecto, entrando a la siguiente liga:

<https://start.spring.io/>

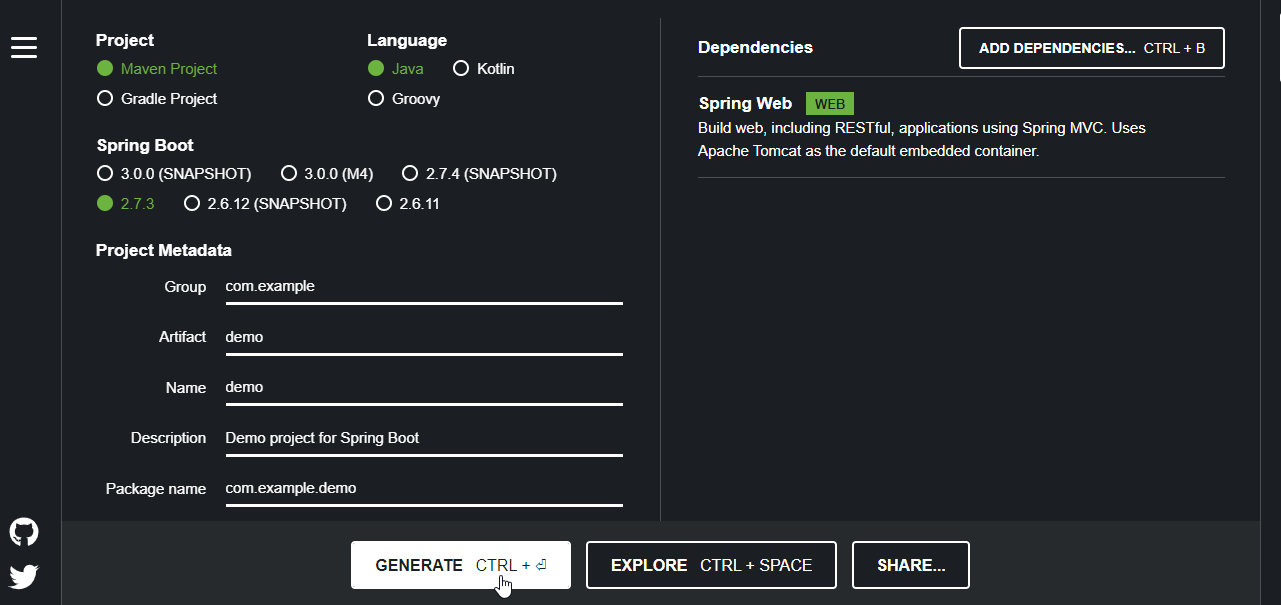
esto para poder creare de manera rápida y eficiente el proyecto, seleccionando si se quiere con Maven/Gradle, que lenguaje, que versión de spring boot, así como el nombre del proyecto, que grupo, etc.



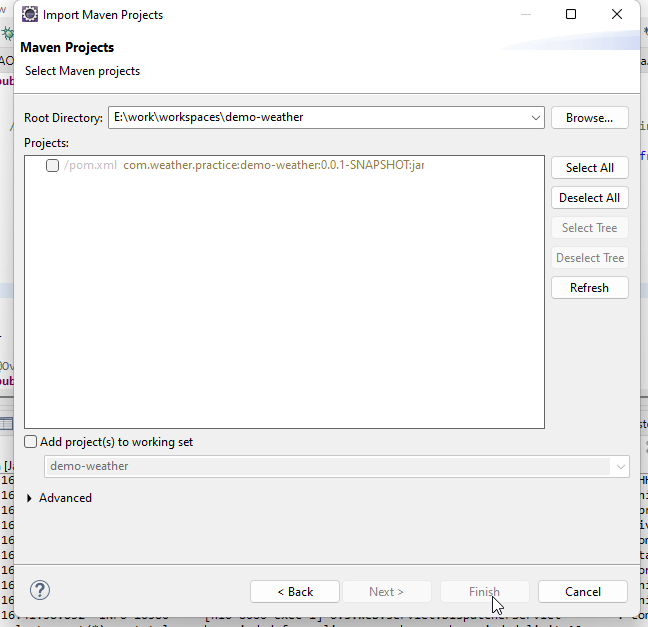
6.- Dentro de la parte derecha, hay un apartado de dependencias, en el cual para la app rest, agregamos la web.



7.- por último, solo presionamos en generar, esto generara el proyecto de tipo Maven.



8.- dentro de eclipse, nos vamos a File/Import/Maven/Existing Maven Project y buscamos el proyecto que acabamos de generar y presionamos finish.



9.- Maven comenzada a descargar todas las dependencias (JARS) necesarios para poder trabajar, así como también dentro del pom.xml, se agregan otras dependencias, en este caso se agregaron las siguientes dependencias, para conexión con JPA, api rest de Open Weather, gson para interpretar la respuesta de Open Weather.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.squareup.okhttp3</groupId>

<artifactId>okhttp</artifactId>

<version>4.10.0</version>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

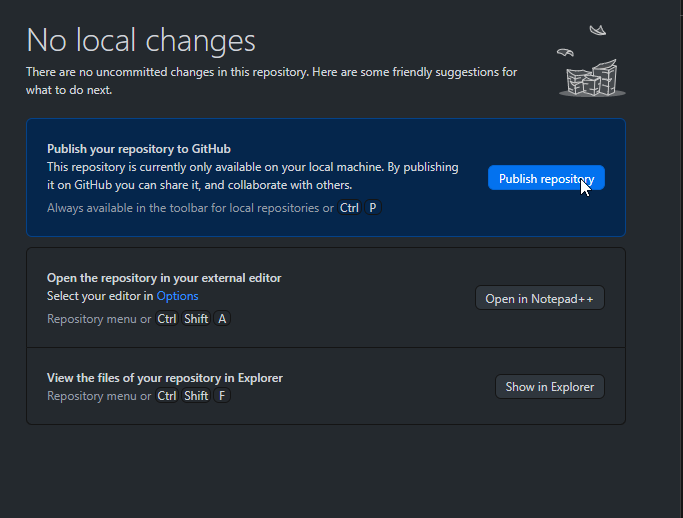
<artifactId>gson</artifactId>

<version>2.8.5</version>

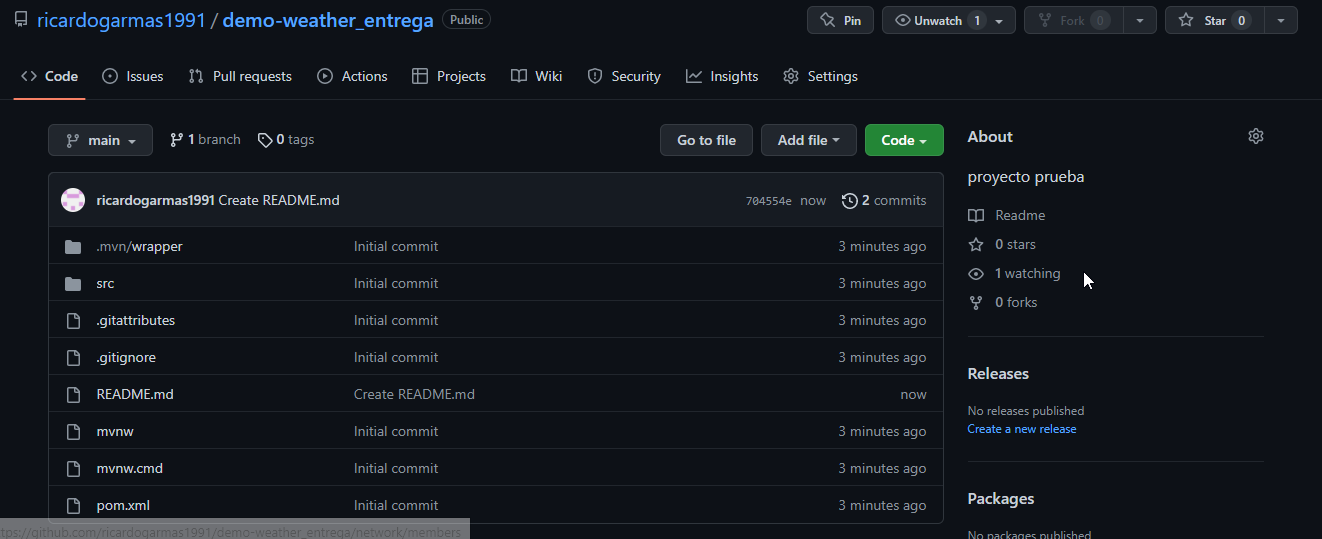
</dependency>

10.-Terminado el proyecto, se sube agregando los archivos a GITHUB

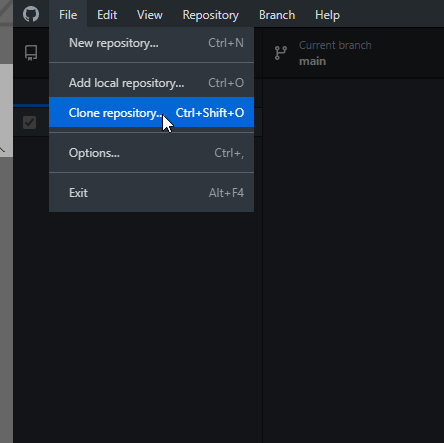
11.- creado el repositorio, descargamos guthub desktop y nos loggeamos con las credenciales anteriores y seleccionamos Publish repository, nos pedirá la ruta del proyecto, así como también verificara si ya esta creado en github.



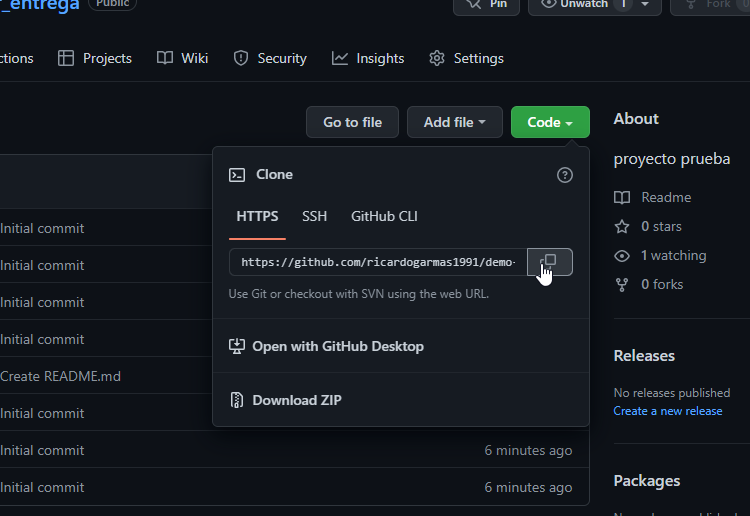
12.- En este paso ya se abra subido el repositorio a GitHub, el cual ya esta listo, para poder clonarse para modificaciones:



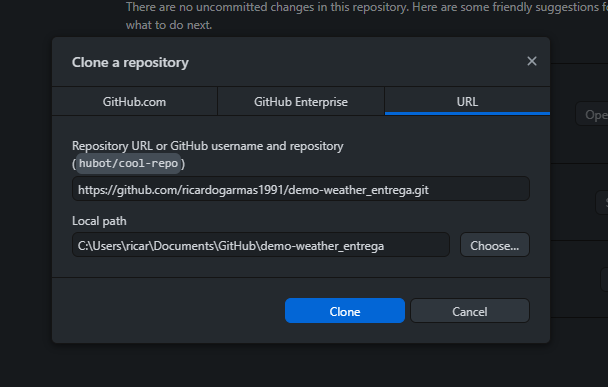
13.- para esto dentro de guthub desktop, entramos a File/Clone repository



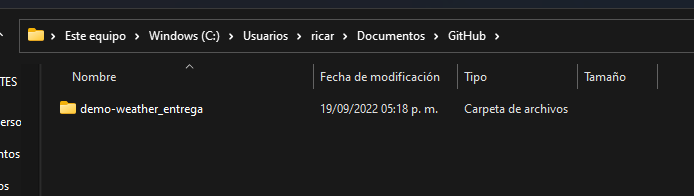
14.- Entramos a github, dentro del apartado de Code, copiamos la URL



15.- Pegamos la Url dentro del apartado de URL en desktop y clickeamos CLONE



16.- verificamos que el proyecto ya se haya bajado y hacemos los mismos pasos para importar en eclipse:



17.- para probarlo, necesitamos PostMan, se crearon 2 metodos, los cuales fueron los siguientes:

<http://127.0.0.1:8080/apiExample/clima/Mexico>

<http://127.0.0.1:8080/apiExample/clima>

esto para la solicitud de clima según la ciudad seleccionada y el segundo para obtener todas las consultas realizadas de las ultimas 10 ciudades.

18.- Anexo también el script de la BD, para guardar la información, la base de datos se nombro como ApiExample:

**CREATE** **TABLE** clima (

id **int** **NOT** **NULL** **AUTO\_INCREMENT**,

nombre\_ciudad **varchar**(250) ,

longitud **float** ,

latitud **float** ,

descripcion\_clima **varchar**(250) ,

temperatura\_actual **float** ,

temperatura\_minima **float** ,

temperatura\_maxima **float** ,

velicidad\_del\_viente **float** ,

error **varchar**(250) ,

hora\_de\_consulta **TIMESTAMP** ,

**CONSTRAINT** id\_climakey

**PRIMARY** **KEY** (id)

);