**Manual de Angular-Spring**

Se instala typescript -g

Se instala angular en el símbolo de sistema con permiso de administrador -g

**Capítulo 05 – Creando nuestra aplicación de angular**

mkdir angular

cd angular

ng serve -o para abrir el proyecto de angular en el navegador

ng serve es para desarrollar no para producción, para producción se debe transpilar convertir todo nuestro typscript en código estatico en html y javscript puro y lo podemos subir a cualquier servidor.

Se puede incluso usar el servidor de apache o node.

- Se borra todo el contenido de app.component.html y se pone un par de h1

- se instala el plugin : linter o un plugin para typescript

- Se me bloqueo la programación de etiquetas html, la razón es que tenia un plugin de django.

**Capitulo 06 – Estructura de directorio del proyecto Angular**

e2e este directorio sirve para hacer pruebas unitarias.

.editorconfig el tipo de documento html tenemos, aca se configura y demás ajustes.

angular.json:

“ourputPath” directorio de salida, todo elcontenido cuando se hace el deploy

en package.json:

el ambiente de desarrollo incluso para el testeo.

“devDependencies”

“dependencies”, si se borra node\_module se pueden volver a instalar con npm install y se recupera node\_module

**Capitulo 07 – Estructura de directorio del proyecto Angular parte 2**

enviroment.prod.ts se utiliza cuando se crea la carpeta /dist

en el main.ts

Se puede trabajar con app móviles

polyfils para aumentar la compatibilidad de nuestra aplicación

test.ts pruebas unitarias.

**Capítulo 08 Implementando boostrap**

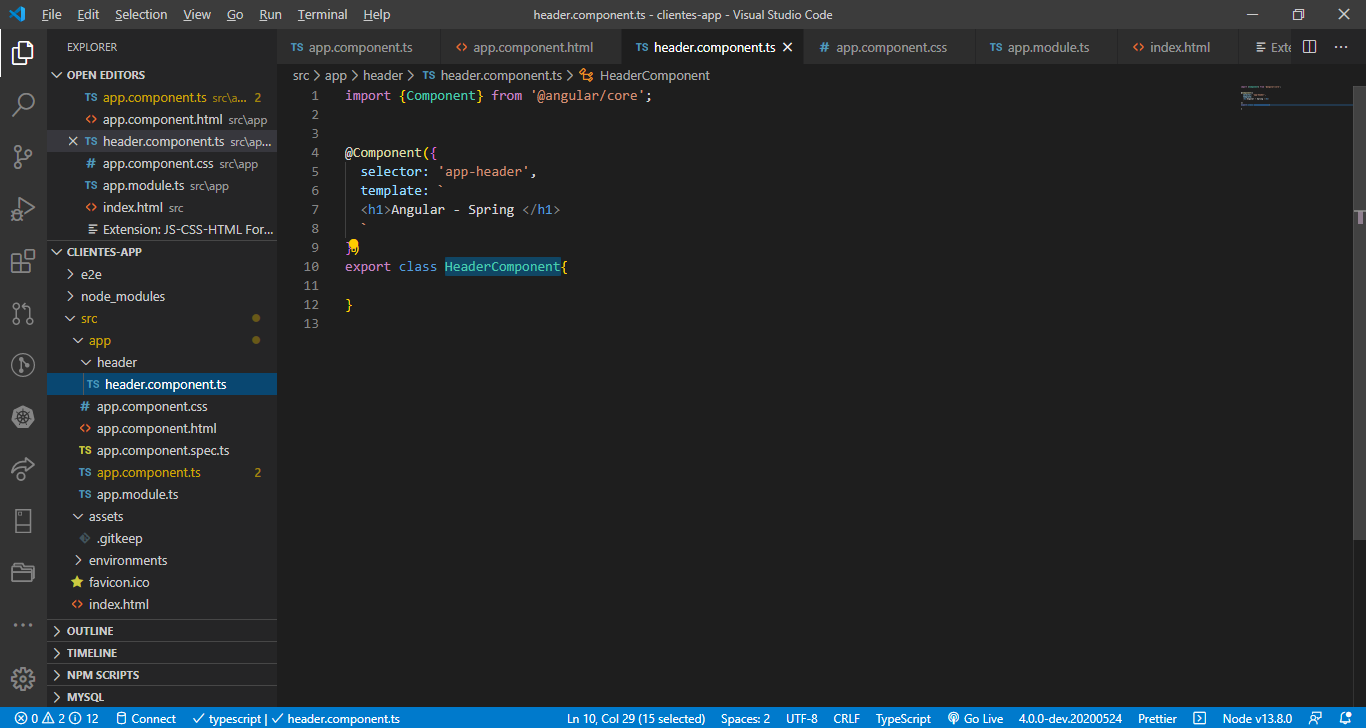
es bueno dejar los scripts al ultimo.

abajo del <app-root> </app-root> se coloca abajo para que no haiga errores:

<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>



ng generate component carpetaCreada’/componentName’ se crea en la raíz de app o src sea el caso para crear los cuatro componentes.

**Directivas estruturales \*ngFor**

- directiva.component.spec.ts es un archivo de testing

- con un plugin de typescript incluso se autocompleta los ng para diferentes tipos.

**Directivas estructurales \*ngIf**

**Creando componente clientes**

ng g class cliente

separar bien el footer hacia abajo.

**Creando archivo clientes.json.ts con la lista de objetos**

se crea un archivo diferente y se importa en el cliente.component.ts

**Creando la clase de servicio ClienteService y la inyección de Dependencia.**

**Configurando e integrando Bootstrap de forma local en nuestra app**

Los js y css se colocan en en angular.json

**Ocurrencias/soluciones**

**\* El footer se fusionaba con la table, pues para arreglarlo solo no la manipule y listo.**

**\* el rxjs se pone solo para importar el observable ya que es así para nuevas versiones.**

**\* angular es una sola pagina, spa(simple page application).**

**\* Desde la versión 6 y 7 de angular se pasó a llamar angular.json, pero en versiones anteriores de angular se llamaba angular.cli.json, pero básicamente son y sirven para lo mismo.**

**----------------------este footer se queda estático----------------------**

**Pero se cruza cuando hay una tabla grande**

footer {

    position: absolute;

    bottom: 0;

    height: 60px;

    width: 100%;

}

**En scripts de angular.json se coloca en la ruta de importaciones o en el arreglo [ src/.../../]**

Implementacion de Bootstrap desde npm

npm i [Bootstrap@4.0.0-beta.2](mailto:Bootstrap@4.0.0-beta.2) jquery-popper.js –save

se graban en dist/umd y se cambian las rutas en el angular.json

**Herramientas necesarios para el backend**

**JDK (JAVA SE DEVELOPMENT KIT) 10.0.2 y se instala modo administrador.**

**IDE ECLIPSE + SPRING TOOLS 4**

spring tolos = crea los proyectos con spring en eclipse e incluye un servidor, embedido.

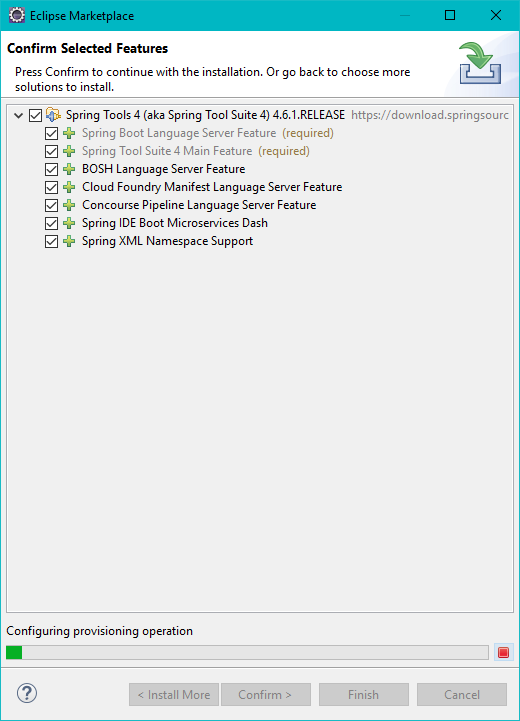
en su pagina eclipse.org(Eclipse IDE for Enterprise Java Developers (includes Incubating components))

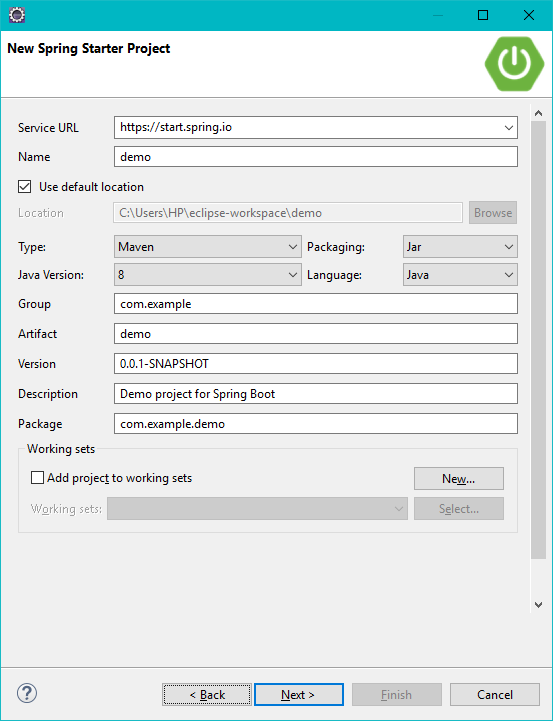
**MAVEN**

Se encargan de manejar las dependencias.

**MYSQL**

**POSTMAN**





**Maven/ gradle es lo mismo**

**Jar: Para aplicaciones que no utilizan jsp pero si vistas timelitf o apirest que retornan json**

**War: Para crear app basados en jsp o desplegar en servidor externo**

**Spring boot ya tiene un tomcat embebido**

**En spring.io se puede descargar eclipse configurado específicamente para spring en sección tolos de la página de spring.**

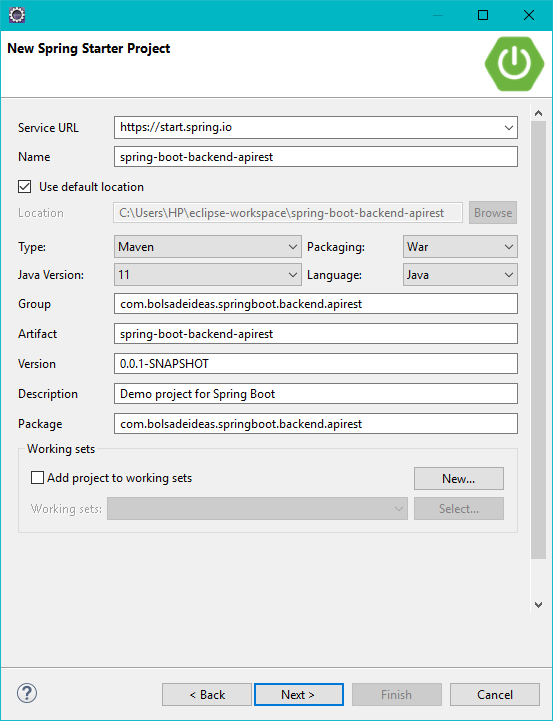
**REST**

Tranferencia de estado representacional

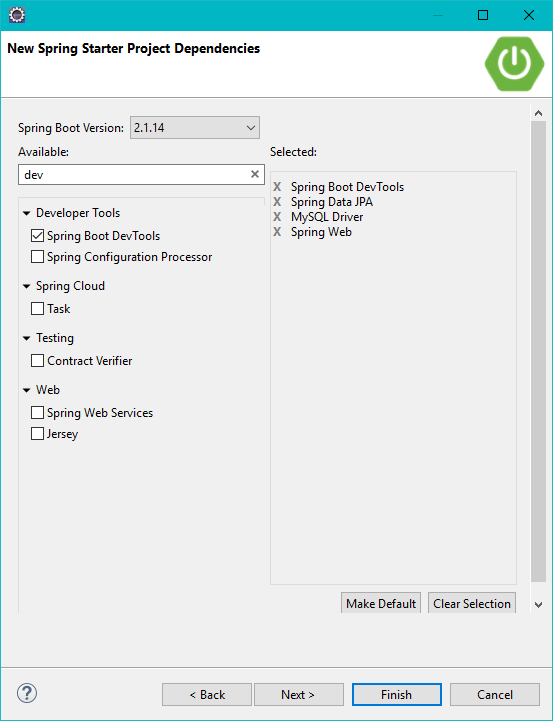
(Representational state transfer o REST)

Es un protocolo cliente servisor stateless en el backend no se maneja ningún tipo de sesiones.eso se maneja en angular utilizando el sessionStorage o localStorage del html5.

Maven tiene estructura: xml gradle tiene estructura: dsl



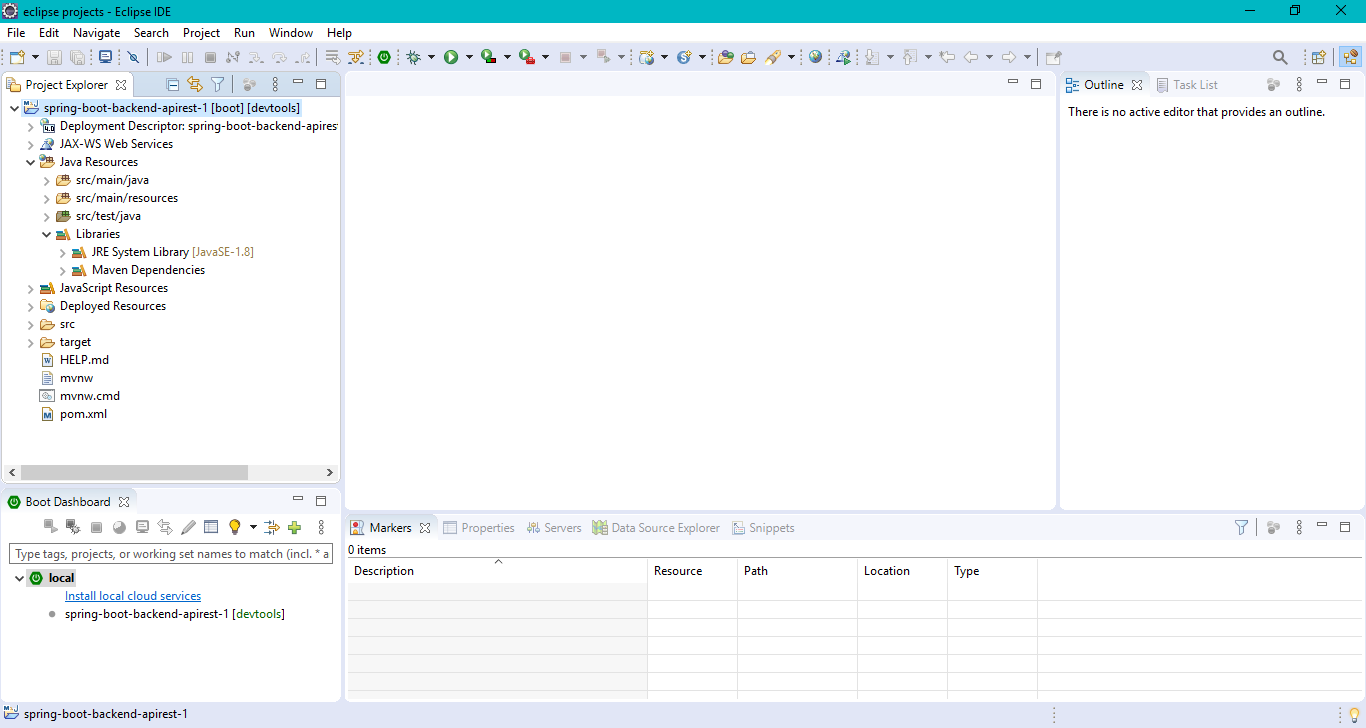
**dev tool: Permite que cualquier cambio que hagamos en la clase service o usuario de forma automática se refresque el servidor y se hace deploy con todos los cambios.**



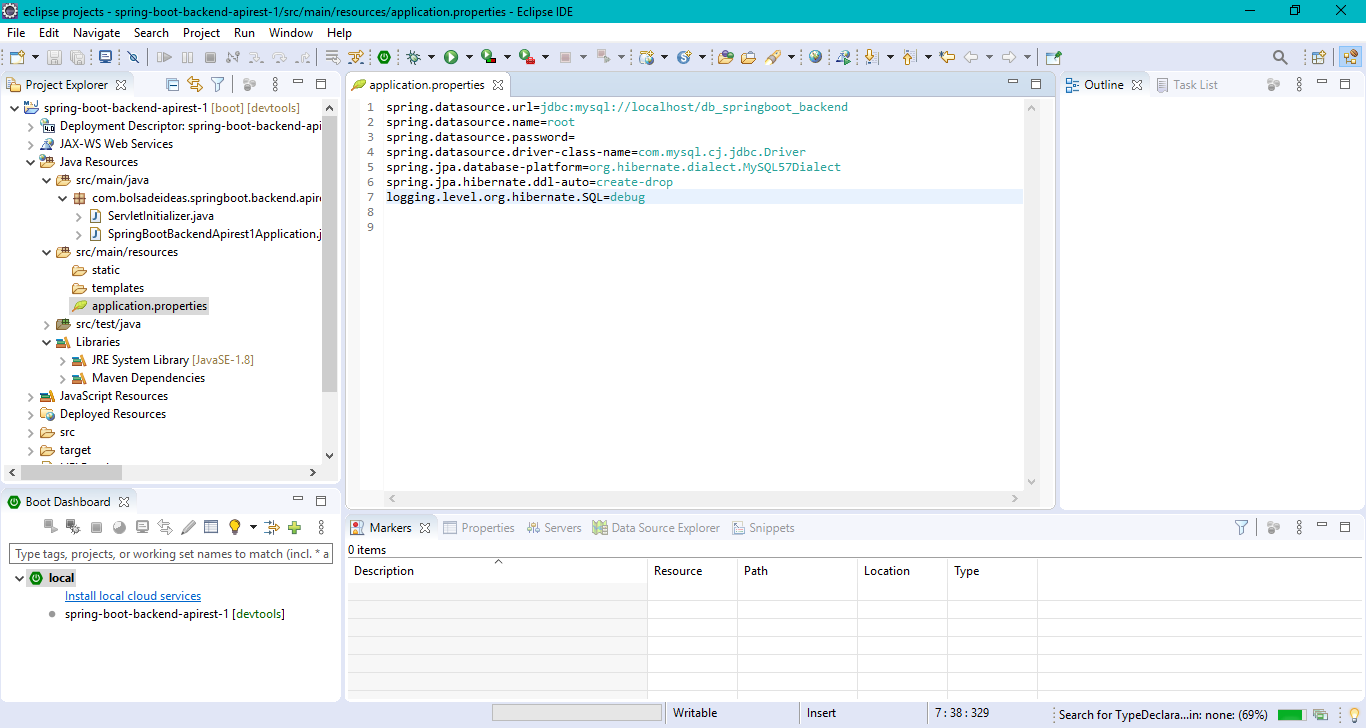
@SpringBootConfiguration Configuracion automática que podemos sobrescribir en el aplicación properties.

@EnableAutoConfiguration Para habilitar la autoconfiguración

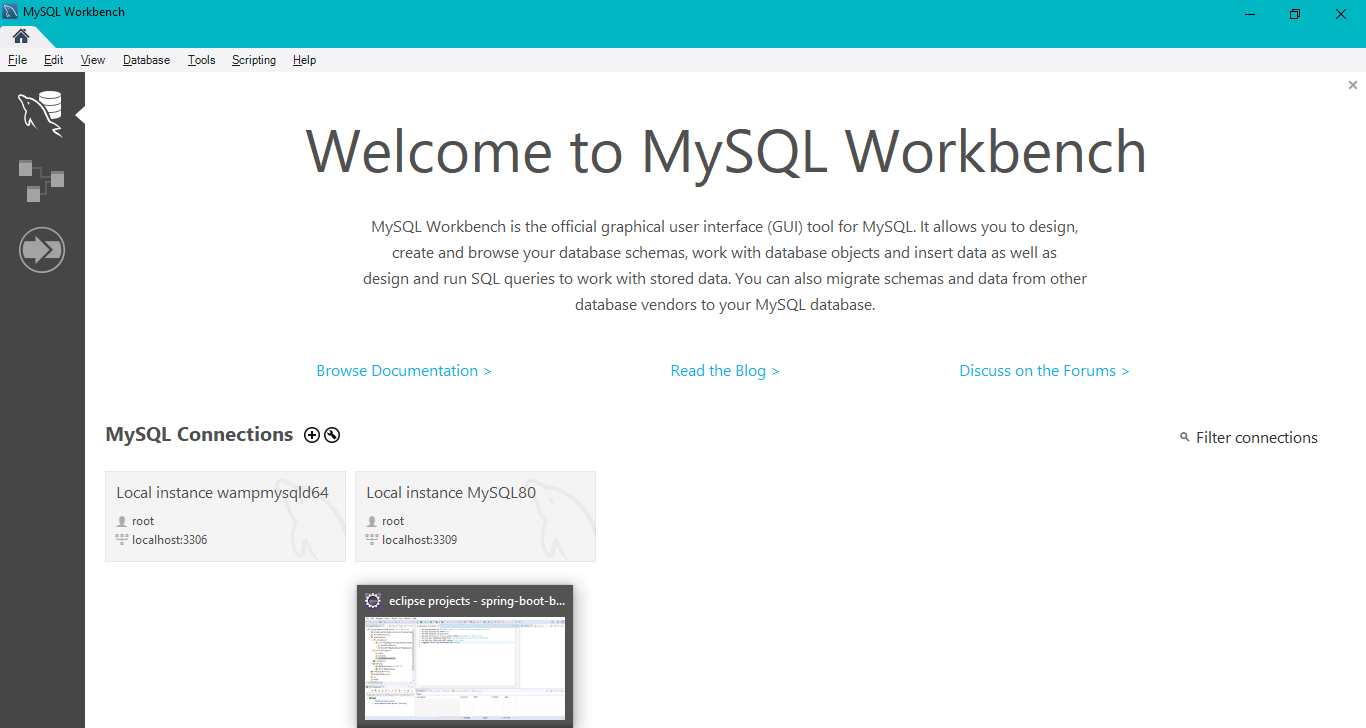
@ComponentScan Para registrar todas las clases o componentes que tiene spring controller o rest controller , para registrar nuestros controladores en el framework



**En libraries están las dependencias de Maven.**



**Instalacion de mysql workbench**



**Crear en mysql-front una db (CREATE DATABASE db\_springboot\_backend;)**

**luego en el proyecto de spring>run AS para ver la conexión**

spring.datasource.url=jdbc:mysql://localhost/db\_springboot\_backend?useSLL=false

spring.datasource.username=root

spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.hibernate.ddl-auto=create-drop

logging.level.org.hibernate.SQL=debug

spring.datasource.url=jdbc:mysql://localhost/db\_springboot\_backend?useSLL=false&allowPublicKeyRetrieval=true

spring.datasource.username=root

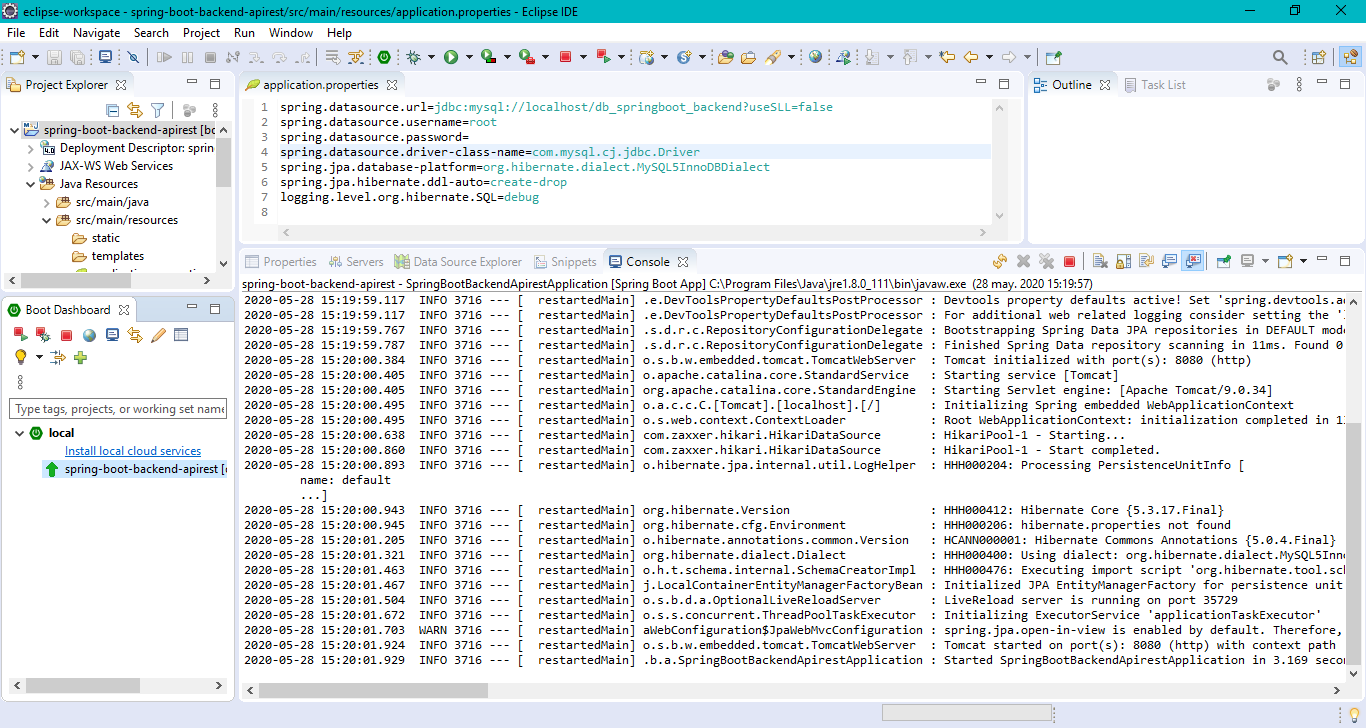
spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

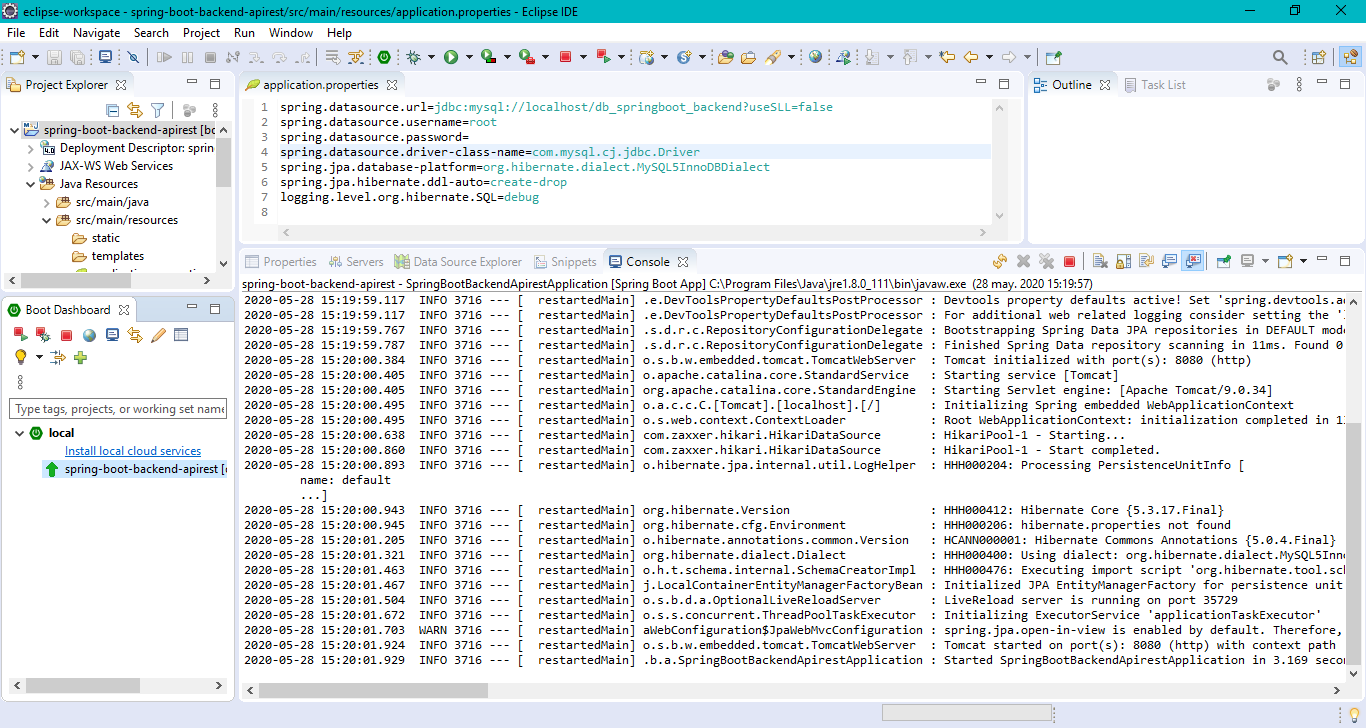
spring.jpa.database-platform=org.hibernate.dialect.MySQL57Dialect

spring.jpa.hibernate.ddl-auto=create-drop

logging.level.org.hibernate.SQL=debug



**Se crea la clase Cliente.java**



**implementar la interfaz serializable**

**El servidor está en 8080 y no gaurda la configuración de la bases de datos. Se clickea el cuadrado rojo y luego el cuadrado con triangulo.**

**Para que se crea la bd, la clase Cliente debe estar bien tipeada adentro no ‘-’ y no singulares**

**Añadiendo las clases Repository y Service de la lógica de negocio.**

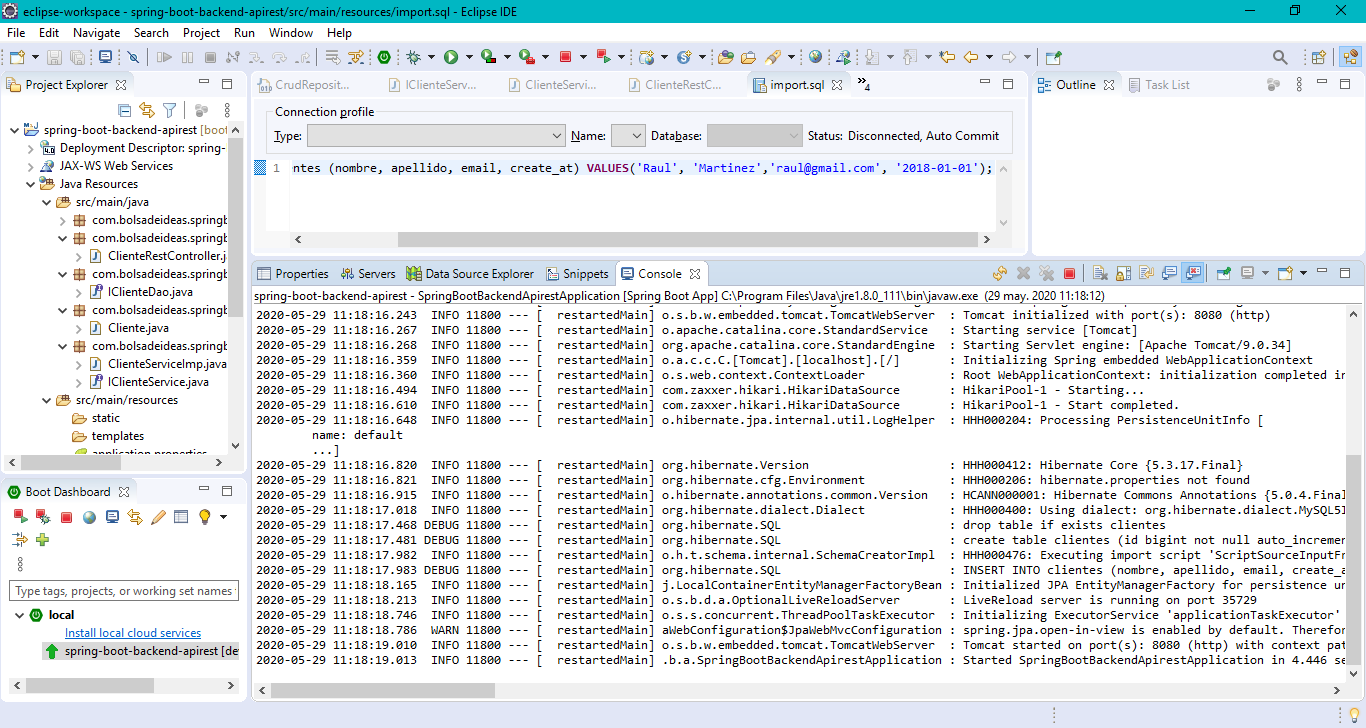
Se crea un paquete que almacena la clase(interface) Dao(IClienteDao) para crear métodos del crud

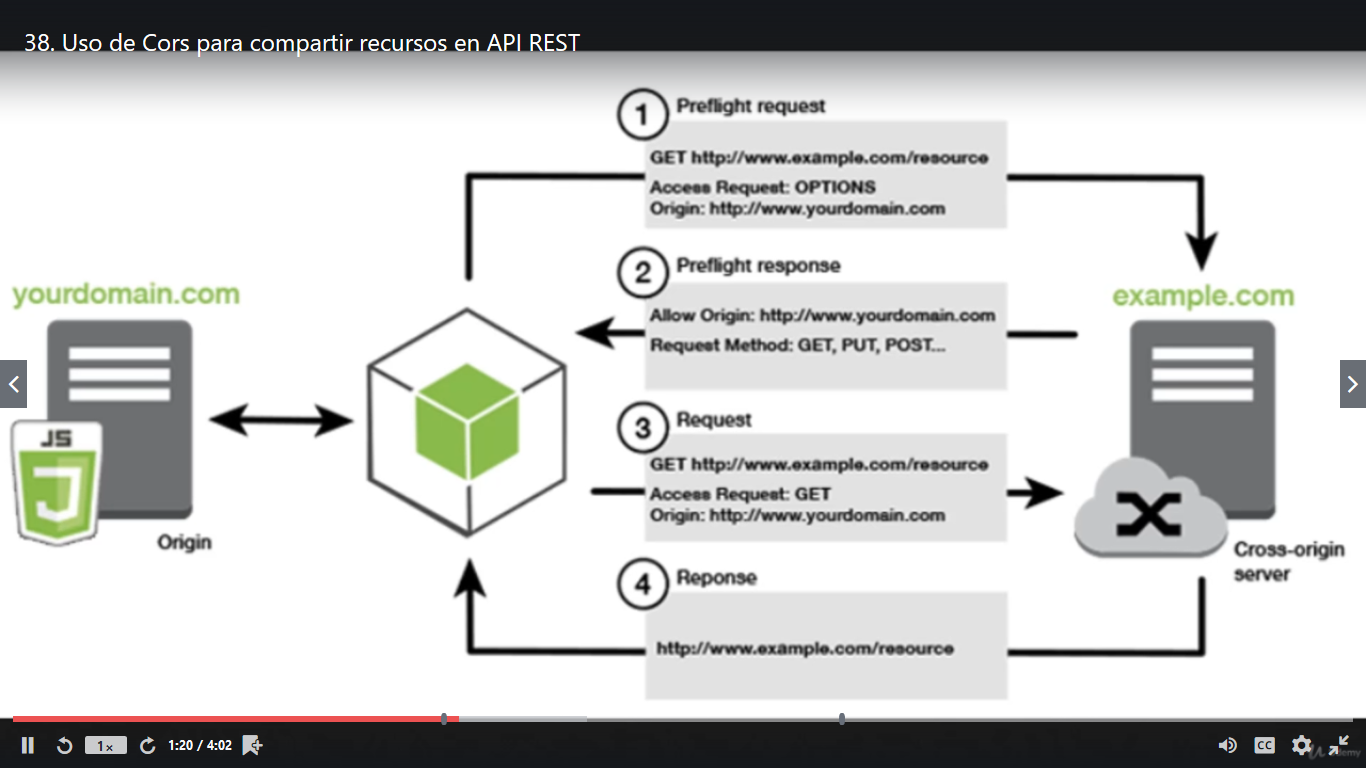
Se crea el IClienteService en el paquete Services, luego se crea ahí mismo la clase service (ClienteService)

**Creando controlador @RestController y EndPoint para listar**

**Añadiendo datos de prueba**

**en src/main/resources, se crea un archivo sql, con nombre import.sql donde se insertaran queries. Se inicia otra vez el servidor.**



**Uso del CORS para compartir api Rest**

**Implementación Servicio Angular con HttpClient**

**- Se importa httpClienteModule en app.module.ts, luego en cliente.service,ts se jalara la infomracion del API**

**- Se jala el único registro que tiene el api a la vista.**

**Escribiendo los métodos del CRUD en la clase ClienteService del Backend**

- Luego se comprubea el api en POSTMAN para verificar el buen funcionamiento del CRUD.

**CREANDO EL COMPONENTE FORM.COMPONENT Y LA VISTA DEL FORMULARIO.**

- ng g c clientes/form –flat (para que se cree e la carpeta clientes y no cree otra carpera form)

Clase 50

SweetAlert se debería poner la versión 7 o mas de 8, la 8 da errores

npm i sweetalert2 –save y se implementa en form.component

**56. MANEJO DE ERROR EN EL BACKEND EN METODO HANDLER SHOW (OBTENER POR ID)**

en vez de :

spring.jpa.hibernate.ddl-auto=create-drop

y se comenta el insert de import.sql

\* Se interpola incluso hasta string, si hay comilla simple o doble reemplazarla con la interporlacion `` y el código.

\* Para que el footer se baje hasta abajo y no divida la tabla se configura el style.css global para efectos globales.

html {

    position: relative;

    min-height: 100%;

}

body {

    margin-bottom: 60px;

}

el footer en su css también tiene

.footer {

    position: absolute;

    bottom: 0;

    height: 60px;

    width: 100%;

}

y app.component.html

<div class="container my-3 pb-5"> en pb el 5 es el maximo

**Clase 56: Manejo del error en el backend, en el método de la anotación:**

@GetMapping("/clientes/{id}") y

@PostMapping("/clientes")

Aedmas en el cliente se configura para que nombre y apellidos no sea nuleables e email sea unique.

Clase 62 se ve como enviar json desde el backend si sale undefined en frontend

63 en nagular poner validator para diferentes usos

**64 JAVA BEAN VALIDATION**

**Se valida los constraints en la clase Cliente. si usamos la 1.5 de spring boot se usa la de hibernate.validator.constraints sino la javax .validation.constraints y con código java se valida para que bote los mensajes en correspondientes**

**67. MANEJANDO LOS ERRORES DE VALIDACION EN ANGULAR.**

**Se crea un if en clienteservice para validar status:**

  if(e.status==400){

          return throwError(e);

        }

**Luego se crea un atributo que contenga un arreglo de mensajes de errores de tipo string en form.component.ts. Luego en danger se colocan en la vista.**

**69. Costumizar mensajes de error en español desde el backend**

**@NotEmpty(message=”No puede ser vacío”)**

**71.- Operador map formato uppercase en Observable:**

**Nombres en mayúsculas, fechas al reves**

'EEE dd/MM/yyyy' dia de la semana abreviado

'EEE dd/MM/yyyy' dia de la semana sin abreviar

'EEE dd/MMM/yyyy' triple M sale el nombre

'EEEE dd, MMM yyyy'

'fullDate'

**Se importa el idioma para que sea general**

**en la plantilla se puede formatear al uppercase en el html clientes.component.html**

import { LOCALE\_ID } from '@angular/core'; en app.module.ts

providers: [{provide: LOCALE\_ID, useValue: 'en-US' }]

      <td>{{ cliente.id }}</td>

                    <td>{{ cliente.nombre }}</td>

                    <td>{{ cliente.apellido | uppercase}}</td>

                    <td>{{ cliente.email }}</td>

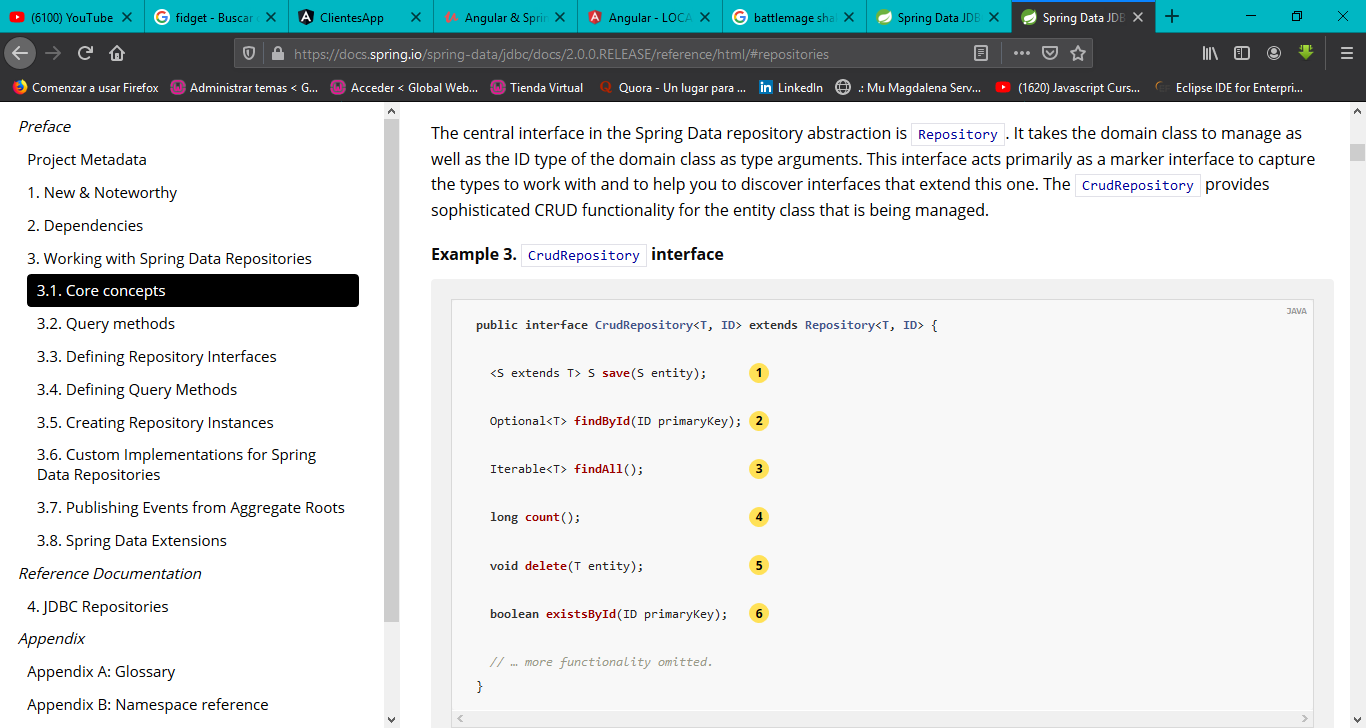
                    <td>{{ cliente.createAt | date: 'EEEE dd, MMM yyyy'

}}</td>

75. Uso del Operador Tap en el Observable.

**Sección 11: Paginación desde el Backend**

[**https://docs.spring.io/spring-data/jdbc/docs/2.0.0.RELEASE/reference/html/#reference**](https://docs.spring.io/spring-data/jdbc/docs/2.0.0.RELEASE/reference/html/#reference)

Se elige la current.

Se importa en clienteDao:

import org.springframework.data.jpa.repository.JpaRepository;

Luego en

IClienteService Se implementa método pageable

ClienteServiceImp luego se crea el método o se implementa desde el titulo ClienteServiceImp

Luego en el controlador rest, se implementa el método handler para paginar.

**78. Modificando el Observable de Clientes en el front-end.**

**79. se crea la ruta de la paginación**

**80. Creado archivo paginador**

**en consola ng g c paginator**

**81 Inyectar paginador usando decorador @Input en el componente Paginador**

**82.- Implementando los links de las páginas.**

si clickeas en pagina uno se desactiva cuando estés en esa pagina

85.-Paginador con muchas paginas OverFlow.

**86.- Implementando rangos de paginas en el componente Paginador**.

ngOnChanges():void{

//Aca se pone el calculo para que vaya actualizando mientras recorren las las paginas

}

**89. Añadiendo el campo para la fecha**

El null de la fecha sale un cuadro con x roja, en bd parece que no se puede modificar. Se modifica en la ases de datos pero sale no se pudo registrar por problemas en la bases de datos., se pone not null en el cliente en eclipse.

\*El datapikcer del formulario se registra un dia pero en la lista sale un dia menos, para eso nos vamos application.properties y colocar:

spring.jackson.time-zone =America/Lima

spring.jackson.locale=es\_PE

**90. Añadiendo DatePicker de Angular Material**

**Se instalará angular material con npm o en forma angular**

**Se copia un dataPicker para el formulario.**

<input [matDatepicker]="miDatepicker" class="form-control" [(ngModel)]="cliente.createAt" name="createAt">

                    <mat-datepicker-toggle [for]=" miDatepicker "></mat-datepicker-toggle>

                    <mat-datepicker #miDatepicker></mat-datepicker>

**se importa en app.module.ts, hay cosas que afinar**

import { MatNativeDateModule} from '@angular/material/core';

import { MatDatepickerModule } from '@angular/material/datepicker';

**Sección 13: Subida de imagen Upload.**

Se crea en el backend el método upload, se crea nuevo atributo foto en la

entidad, se crea carpeta uploads

**92.-**

Se sube imagen, se modifica máximo de imagen en plataforma en application.properties:

spring.servlet.multipart.max-file-size=10MB

spring.servlet.multipart.max-request-size=10MB

**94 me quede**

**95.- Añadiendo Logger en el back-end**

Este Logger que estará inicializado y estará en método upload y verFotos, sirve

para ver rutas en el backend de donde guardan las fotos desde el cliente, SE

GUARDAN EN LA CARPETA UPLOAD

+ Se genera esta imagen y se puede ver en el navegador:

<http://localhost:8080/api/uploads/img/0245e1d5-bcff-4ad5-ad48-cdbf96ced778_DIBUJO2.jpg>

96.- AÑADIENDO COMPONENTE DETALL DE LA IMAGEN PARA EL UPLOAD EN EL FRONT-END.

\* Se crea componente detalle del cliente

97.- IMPLEMENTANDO LA SUBIDA DE ARCHIVOS EN EL FRONT-END.

Se crea un nuevo atributo en clientes.ts, se implementa el método upload en clientes.service.ts

Se suba una foto haciendoclick en el id del usuario, se sube y borra la anterior.

98.-

const req = new HttpRequest(‘POST’, ‘/upload/file’, file,{

reportProgress: true;

});

101.- IMPLEMENTANDO MODAL COMO COMPONENTE ANIDADO (HIJO).

se borra: la ruta ver.

el selector : detalle-cliente se pone en clientes.component.html arriba.

Luego en clientes.components.ts se coloca la varible:

clienteSeleccionado

Luego en detalle.component.ts:

@Input() cliente: Cliente; se coloca en las variables arriba y se borra el routerilink en el html de clientes

102.- CONVIERTIENDO A MODAL DE BOOTSTRAP

\* Se crea model.services.ts y se implementa una plantilla de modelen detalle.component.html

103.- Añadiendo estilos y animación de difuminado al Modal

- Se crea su css en su componente, se hace efecto de somrba al model, se utiliza la técnica o modo fade In

104.- AÑADIENDO FOTO AL LISTADO

Se añade la foto al la lista, foto por default y cuando se borra la foto debe haber una predeterminada pero sale error pero para esto se modifica en el backend

105. Implementando EventEmitter para actualizar la foto en el listado

**Los Observables manejan eventos.**

106. Creando la clase Service UploadFileService en el Backend para optimizar.

-- Se hará una clase service nueva de upload, primero se creara un interface

-- Luego se crea una class UploadFileServiceImpl

**Sección 14: Agregando campo select en el formulario y relación de tablas**

**108.-**

1.- Se crea el cliente Region luego en Cliente se enlaza Region,

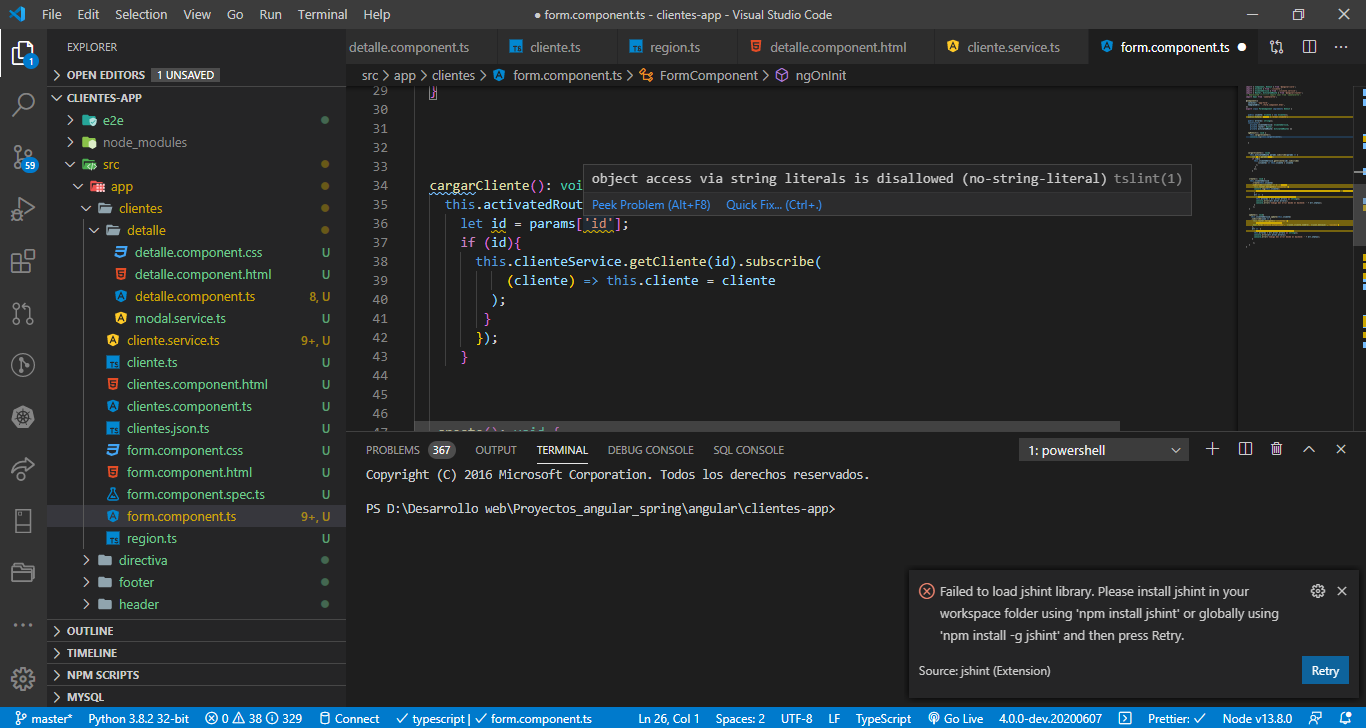
2.- Se crea el método de Region en ClienteDao para listar las regiones.

109.-

Se incrusta el método de región en Cliente Service, luego en ClienteServiceImp se importa el nuevo método para luego ir al ClienteControlador para crear un método de regiones.

Se crean registros de las regiones en mysql

111. ANGULAR Y REGIONES



112.- CREANDO EL SELECT EN EL FORMULARIO

113. COMPARANDO LA REGION EN EL SELECT

[compareWith]

\* Al poner not empty en región al crear el registro en el formulario, sale caution! caso contario con not null, REVISAR EN EL FUTURO.

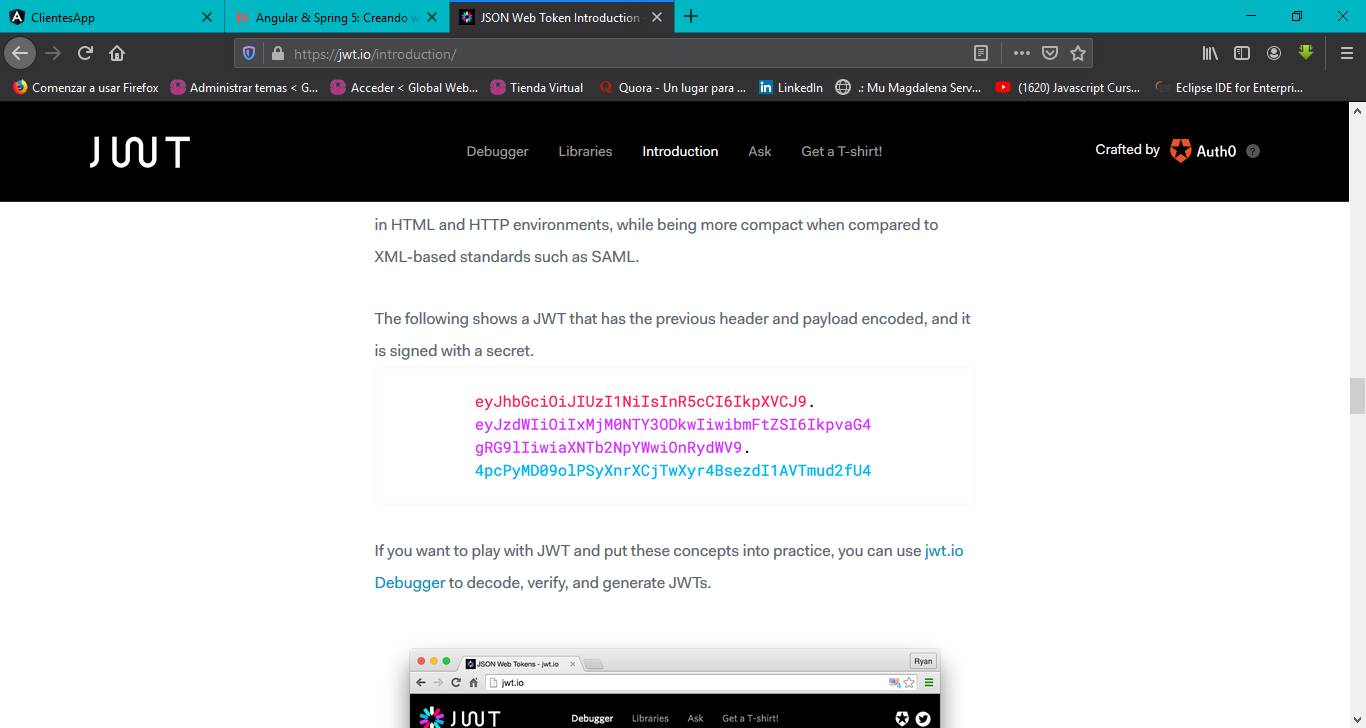
**114.- AÑADIENDO LA OPCION POR DEFECTO CON EL TEXTO SELECCIONAR.**

**Sección 15: Autenticación OAuth2 con JWT:**

**Backend Spring**

**116. Introducción a JSON Web Token (JWT)**

**JWT esta dividida en tres partes**

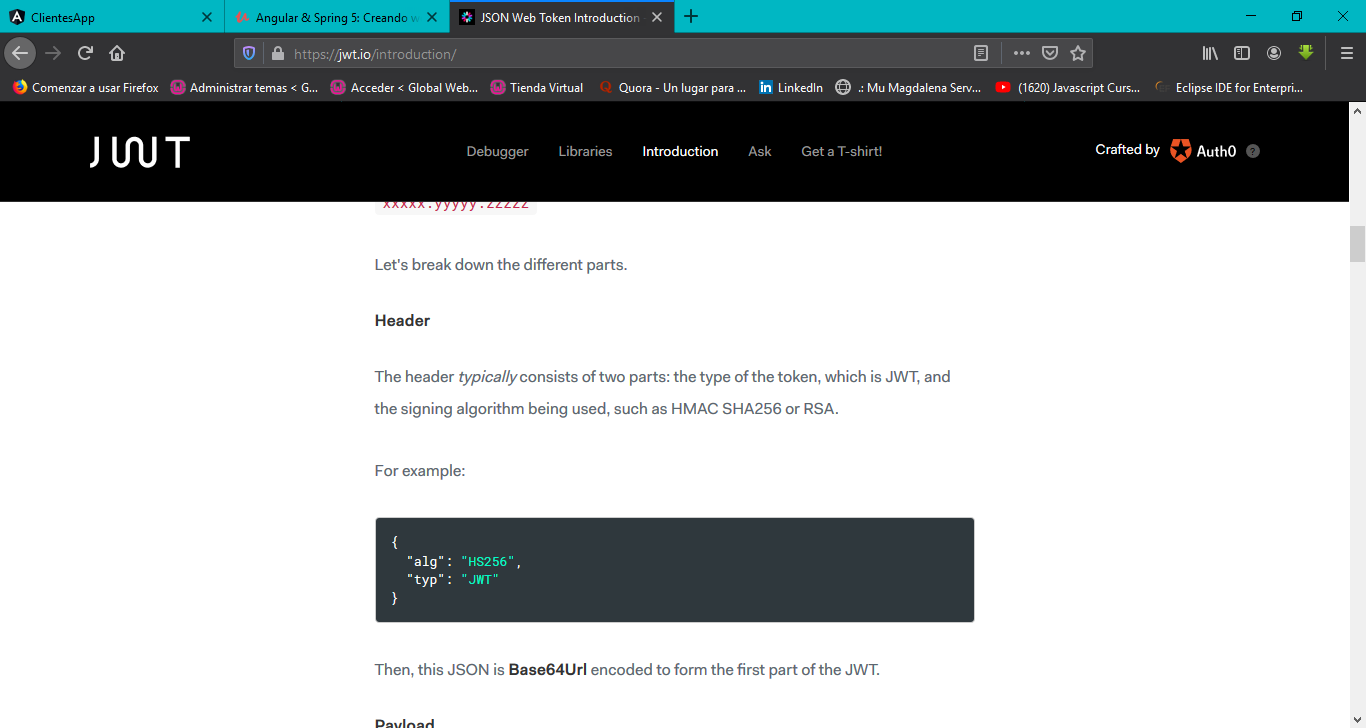


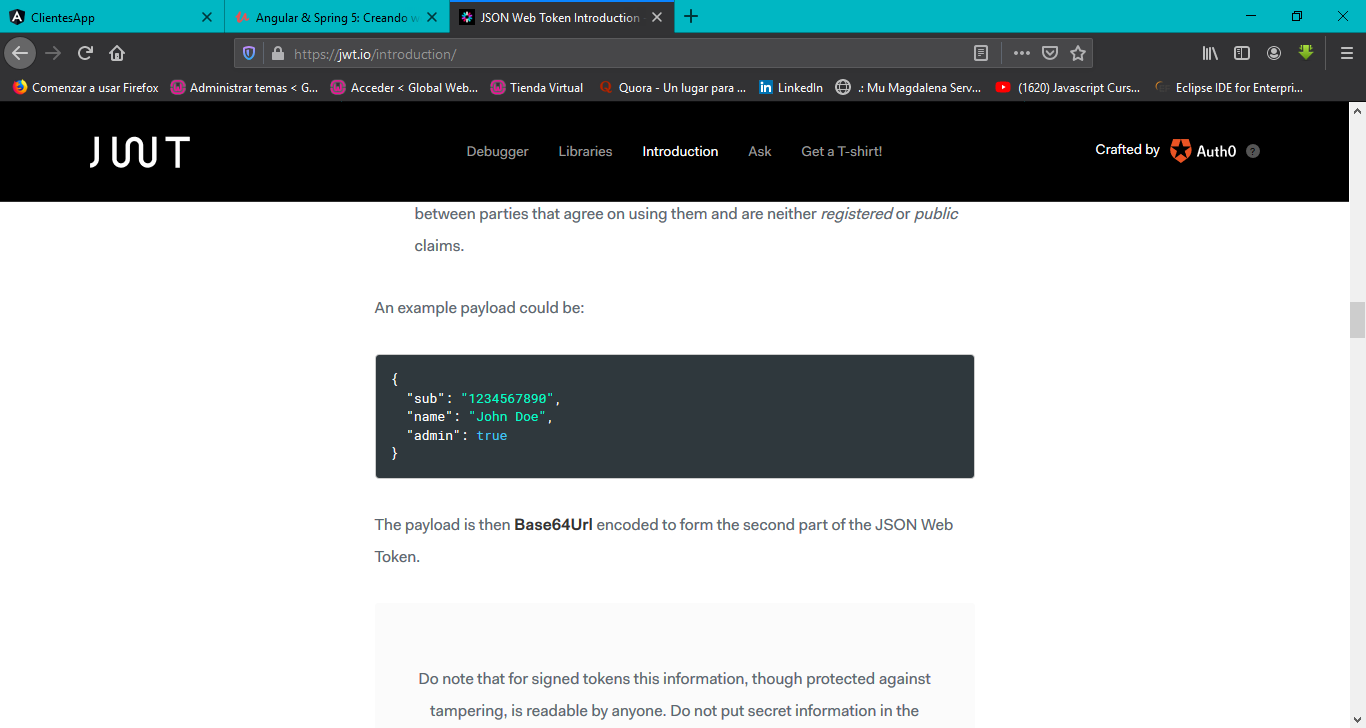
HEADER: ALGORITHM & TOKEN TYPE

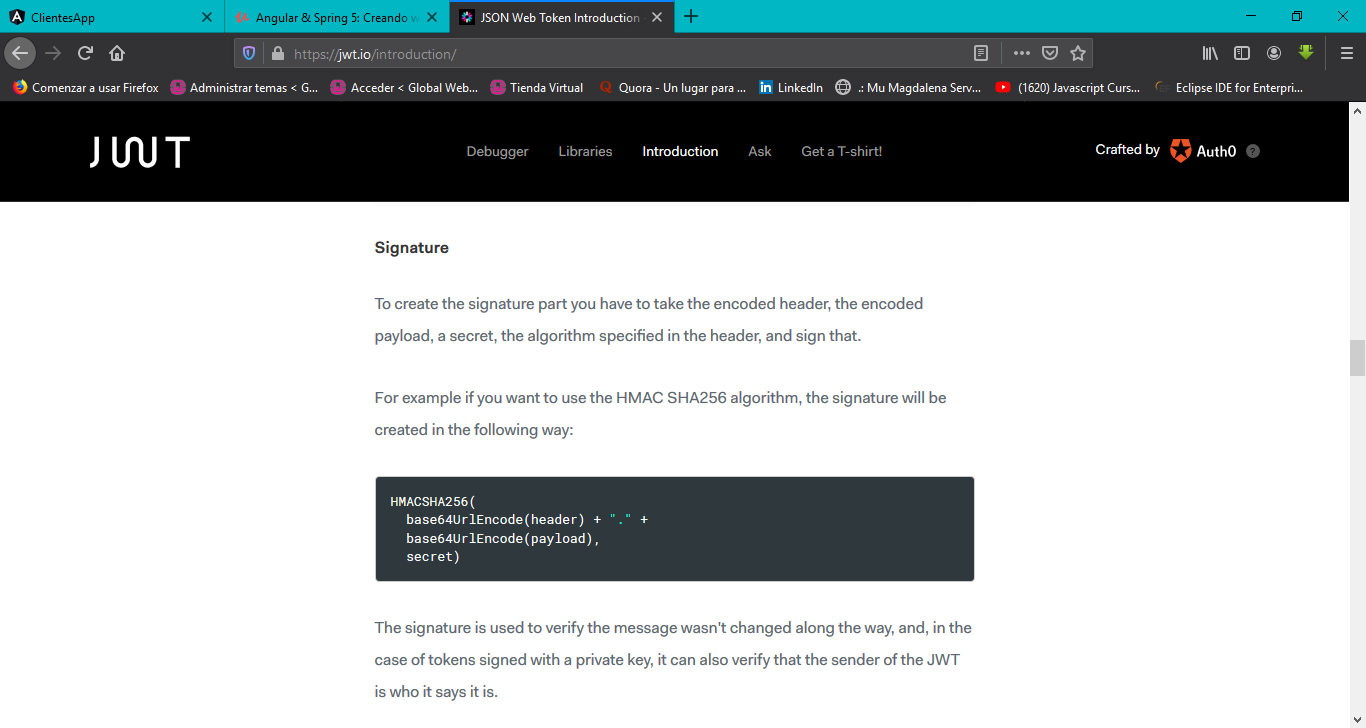
PAYLOAD: DATA

VERIFY SIGNATURE:

La encriptación es BASE64, es fácilmente reversible, decodificable. No se debe guardar información sensible en el Token.







118. Introducción a OAuth2 y añadiendo las dependencias

Spring Security

en mvnrepository se consigue el:

## [OAuth2 For Spring Security](https://mvnrepository.com/artifact/org.springframework.security.oauth/spring-security-oauth2) » [2.5.0.RELEASE](https://mvnrepository.com/artifact/org.springframework.security.oauth/spring-security-oauth2/2.5.0.RELEASE)

con su dependencia:

<!-- https://mvnrepository.com/artifact/org.springframework.security.oauth/spring-security-oauth2 -->

<dependency>

<groupId>org.springframework.security.oauth</groupId>

<artifactId>spring-security-oauth2</artifactId>

<version>2.5.0.RELEASE</version>

</dependency>

Y se coloca en el pom.xml de spring

## Luego

## [Spring Security JWT Library](https://mvnrepository.com/artifact/org.springframework.security/spring-security-jwt) [1.1.1.RELEASE](https://mvnrepository.com/artifact/org.springframework.security/spring-security-jwt/1.1.1.RELEASE)

## <!-- https://mvnrepository.com/artifact/org.springframework.security/spring-security-jwt -->

## <dependency>

## <groupId>org.springframework.security</groupId>

## <artifactId>spring-security-jwt</artifactId>

## <version>1.1.1.RELEASE</version>

## </dependency>

## \* Si tienes Jdk 10 o 9 o más, instalar:

## JAXBException en el cual se instala dos dependencias: JAXB API QUE SE LE DEBE QUITAR LA VERSION, ya que spring boot se la coloca.

## LA IMPLEMENTACION:

## JAXB RUNTIME, se copia su dependencia y se coloca en el pom.xml, se le quita la versión también.

119.-

Se crea la clase Entity Usuario y Role.

120.- Creando el repositorio JPA IUsuarioDao

Se crea el UsuarioDao

121. Creando

**126. Añadiendo la configuración para el servidor de recurso**

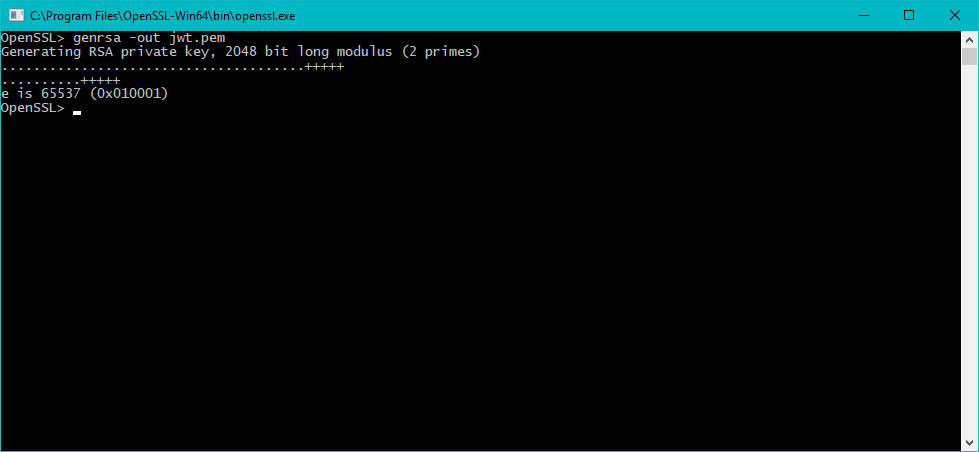
Se quita autenticación a ciertos sectores y quita un error de CORS en la consola.

129. Creando y asignando certificado RSA para firmar el token JWT

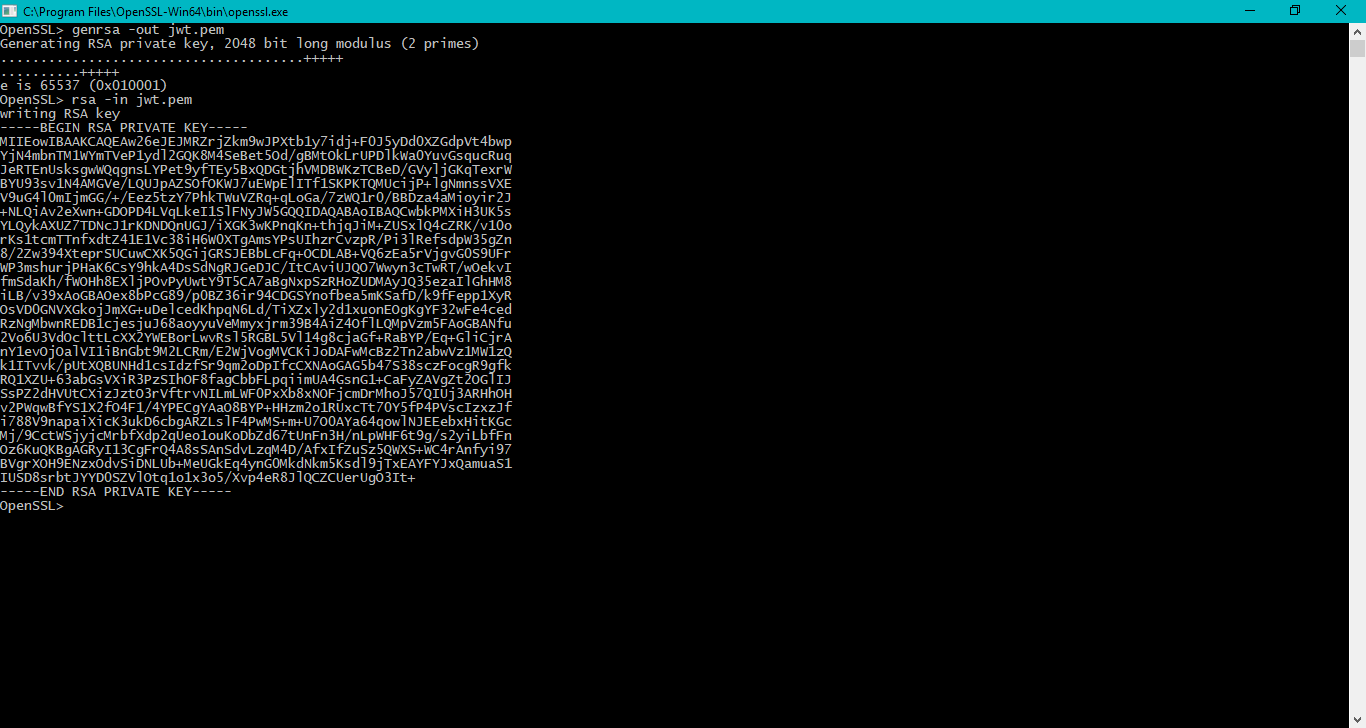
<https://slproweb.com/products/Win32OpenSSL.html>

Se descargar la versión de OpenSLL - Win64OpenSSL-1\_1\_1g,

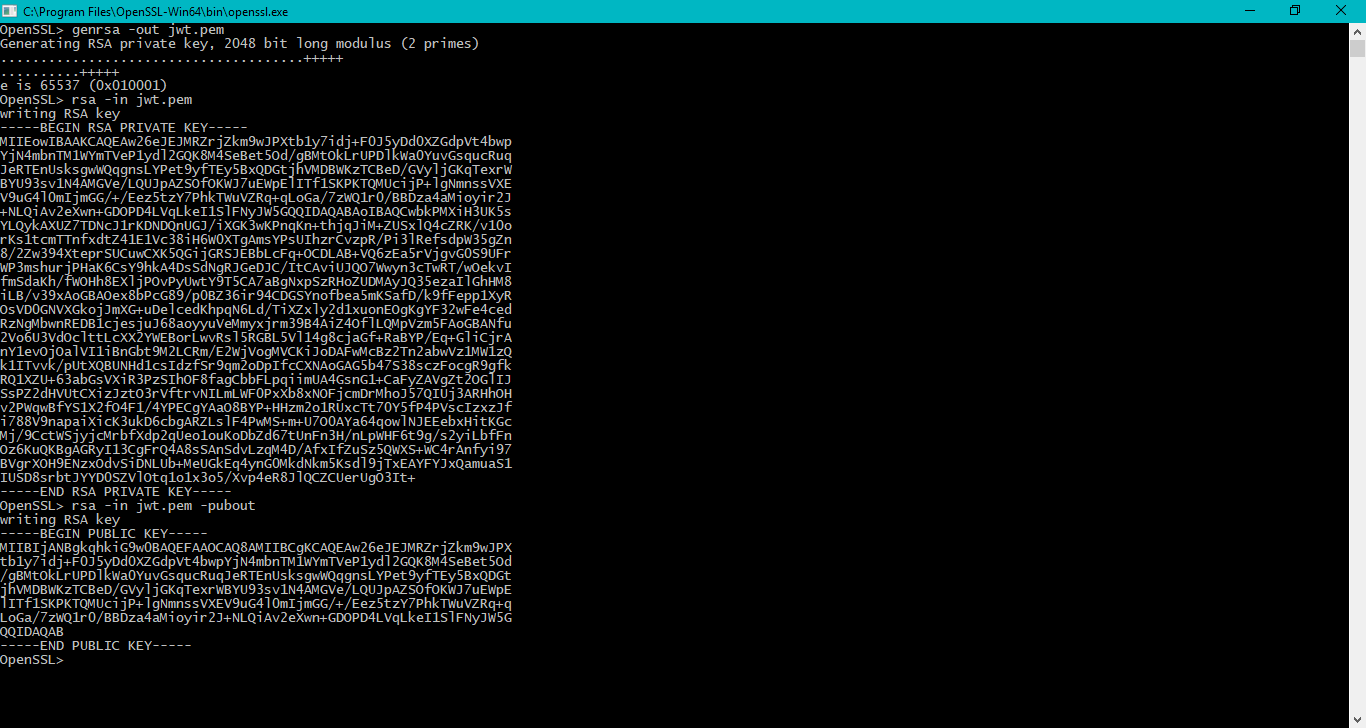
luego en los comandos en la carpeta del OPENSSL en su BIN:



Luego nos muestra la llave privada:

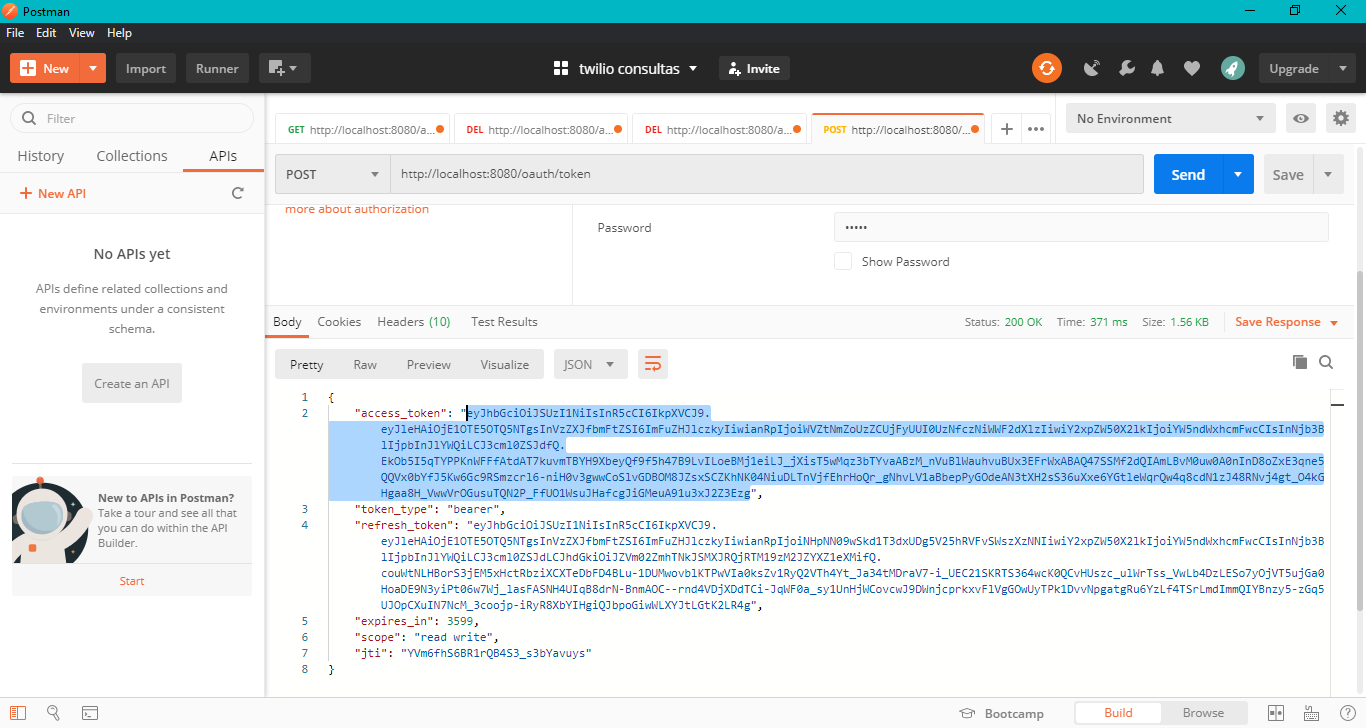


Luego nos muestra la llave pública.

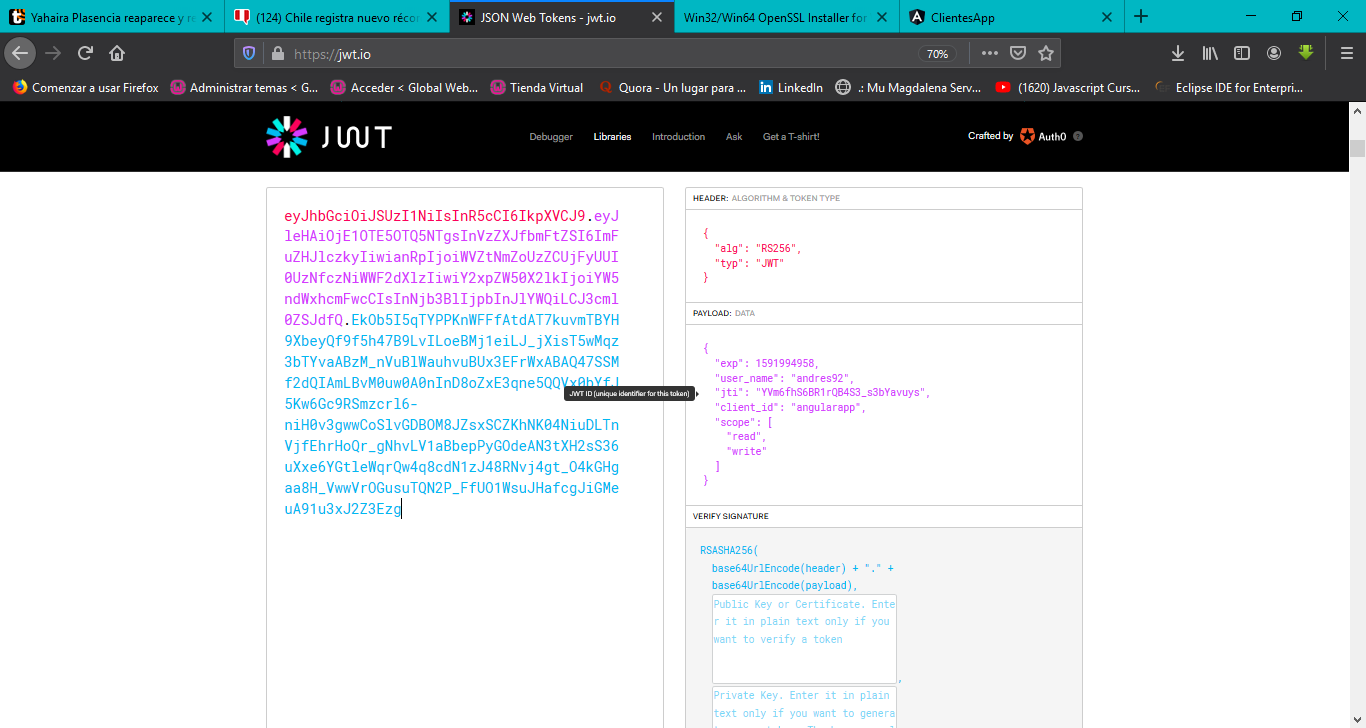


Se crea una variable public static final y se copia la publica y privada en JWTCOnfig

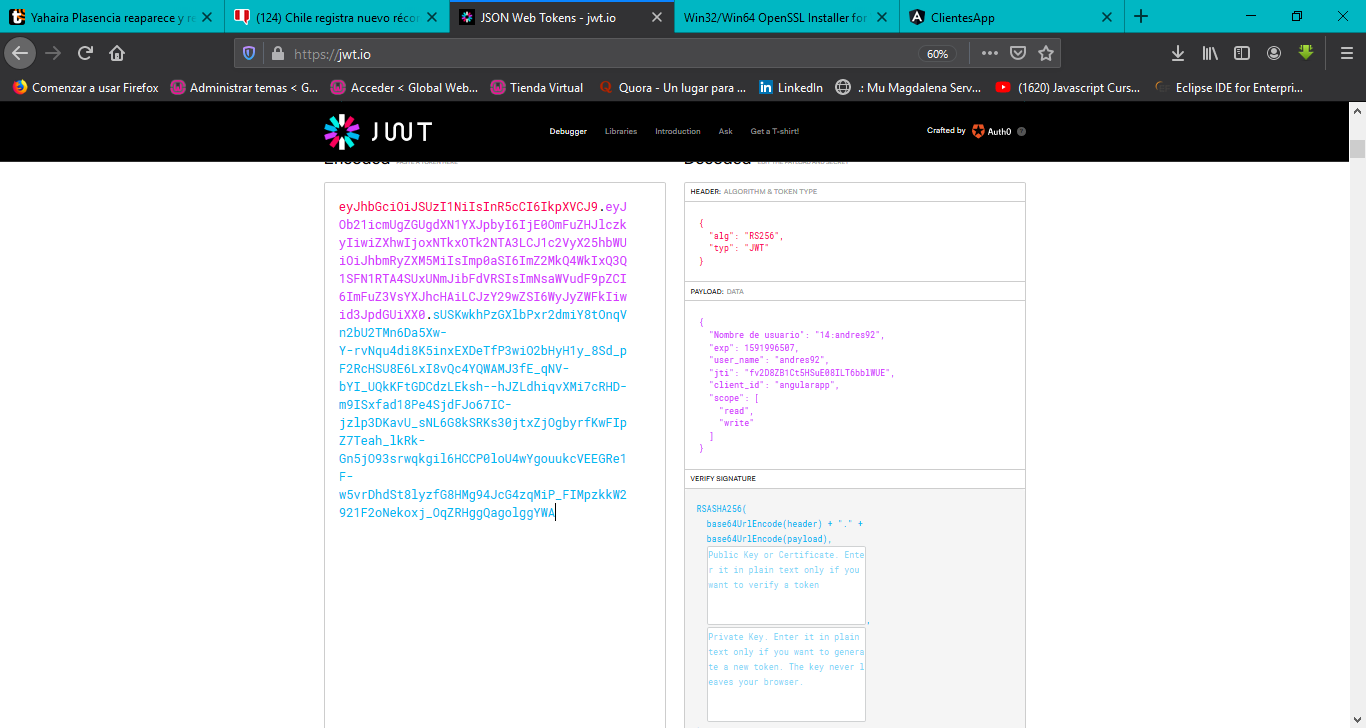
Se copia desde aquí :



Para ver sus características de encode del token:



130. AÑADIENDO MAS DATOS DEL TOKEN AL JWT.



Se le añadió un mensaje: “Nombre de usuario y su iD” en payload: DATA.

131.-

No se emite el token cuando se testea/loguea en Postman, para que funcione borro los atributos nombre, apellido, email, luego:

info.put("Nombre de usuario", usuario.getId() + ":" + usuario.getUsername());

en el importSQL poner la línea de código:

sin los tres anteriores atributos

**SE SOLUCIONÓ BORRANDO LOS FORAIGN Y LAS TABLAS PARA LUEGO VOLVERLAS A CREAR CON CREATE EN EL APPLICATION.PROPERTIES**

132.- AGREGANDO SEGURIDAD A NUESTRAS RUTAS DE FORMA PROGRAMATICA CON HTTP SECURITY.

Se crean las rutas con sus restricciones y permisos , pero de mejor forma se harán con anotaciones.

134. Configurando Cors en Spring Security

\*\* SE PERMITE LA IMAGEN POR DEFECTO EN LAS RUTAS DE allPermit()

135.- CONFIGURANDO EL CORS

\*\*\* SE EXTRAE DE SPRiNG SECURITY > LEARN

**SESION 16: AUTENTICACION CON OAUTH2 CON JWT: FRONTEND ANGULAR.**

ng g c usuarios/login --flat=true (Para que cree un componente login y no un directorio).

**139. Añadiendo el método login() en el componente y nueva clase Usuario.**

**se crea:**

**comando tipo ubicación nombre**

**ng g class usuarios / usuario**

VALIDA SI ESTA NULL/ INGRESA AL SISTEMA CON USERNAME Y CONYRASEÑA

**Deployment de la aplicación.**

**en el video 188 se deploy la aplicación para produccion**

**\*\* ERRORES**

capitulo 57 EL POST, CREA CON NULL, Y CREA EMAIL DUPLICAODS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\* Capitulo 58: EL PUT DE UPDATE CON NULO IGUAL EDITA, NO HAY EMAIL UNIQUE PERO SI ES UN USUARIO INEXISTENTE NO ACEPTA.

**\*\*\*( @NotNull puede ser sustituto de @Column(nullable=false) y arregla el error de los nulos) o mejor modificar el null desde la db**

**\*\* @Column(nullable=false) son validos cuando se crean los entities, mas no despues de modificar la db**

\*\*\*alter table users add constraint UK\_some\_id unique (email) en bases de datos para que sea único el campo de email.

\*\*\*\*Podria ser que se borre la tabla de nuevo y las anotaciones funcones

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

\* Al actualizar en el front-end me lanza en el model de alerta que: **‘cliente undefined fue actualizado’ igual para delete.**

**Solucion: en el cliente.service.ts los Obsevables en vez de Cliente poner any, luego en form.component.ts**

\* Al poner not empty en región al crear el registro en el formulario, sale caution! caso contario con not null, REVISAR EN EL FUTURO.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MORALEJA: Respetar espacios en blanco en cada termino.

  const httpHeaders = new HttpHeaders({

      'Content-Type': 'application/x-www-form-urlencoded',

      'Authorization': 'Basic ' + credenciales

    });

**CURIOSIDADES**

/\*# sourceMappingURL=bootstrap.min.css.map \*/

\* SI SALE ESTO EN BOOSTRAP.MIN.CSS EL BOOTSTRAP MINIFICADO, BORRARLO!

\* El contenido estatico me lanzaba que tenia una rul andefined, lo borre y se fue el error en el navegador

**50. Cargando los datos en el formulario  
para actualizar \*\* NO CARGA LOS DATOS EN EL FORMULARIO PARA EDIATARLOS\*\* Era solo interpolar bien**

\* Al poner el Update en Oninit(){} me botaba indefinido.

\* Al Parar y luego comenzar el server en eclipse, se borran mis registros

\* En application.properties se pone:

spring.jpa.hibernate.ddl-auto=update

**MANUAL PARA EL BACKEND CHAT DE SPRING Y ANGULAR**

Se instalan las dependencias:

Web: web

Messaging: Web Socket

Core: dev tolos

MongoDB: mongodb

**ERROR, SALE QUE NO ESTA DEFINIDO GLOBAL EN CONSOLA.**

La solución es colocar en polyfiles línea 58:

(window as any).global = window;