UD 3: Bases de datos y servicios REST
2.- Visual Studio Code

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Objetivos de la sesión:

- *Aprender a **crear, configurar, ejecutar y depurar proyectos Spring Boot en Visual Studio**.*
- ***Acceder a Bases de Datos** desde nuestro proyecto Spring Boot.*
- *Configurar una **Base de Datos H2**.*
- *Acceder al **gestor de base de datos H2** a través del navegador.*
- *Crear **entidades JPA** asociadas a una tabla de una Base de Datos.*
- ***Inicializar** las tablas de una Base de datos H2 al arrancar la webapp.*
- *Implementar **repositorios** para cada entidad JPA.*
- *Aprender a crear una **aplicación que funcione por linea de comandos en Spring Boot**.*

UD 3: Bases de datos y servicios REST
2.- Visual Studio Code

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

¿Como que Visual Studio Code?

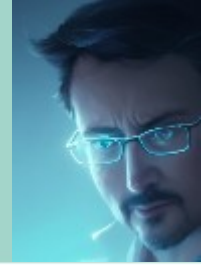
No es un error, en las siguientes páginas vamos a **configurar e importar el proyecto Spring Boot de la UD2 de Eclipse a VSCode**.

¡¡Ahora que ya nos habíamos encariñado /desesperado con el Eclipse!!

PD: Esta práctica puede parece larga, pero es muy sencilla...

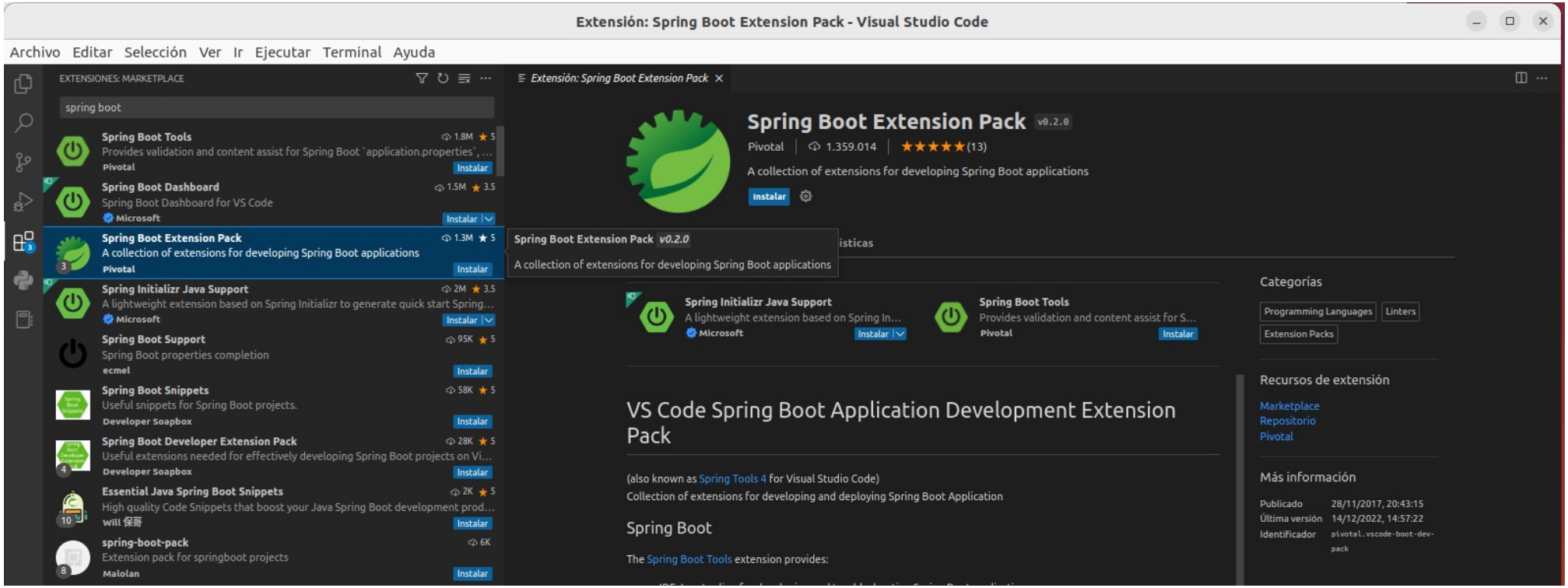
UD 3: Bases de datos y servicios REST
2.- Visual Studio Code

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



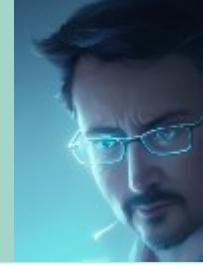
Configuración de Visual Studio Code:

Para trabajar desde VSCode primero debemos instalar el paquete “Spring Boot Extension Pack”. Para ello en el menú superior (Ctrl+Shift+X) y buscamos spring boot. Instalar “Spring Boot Extension Pack”:

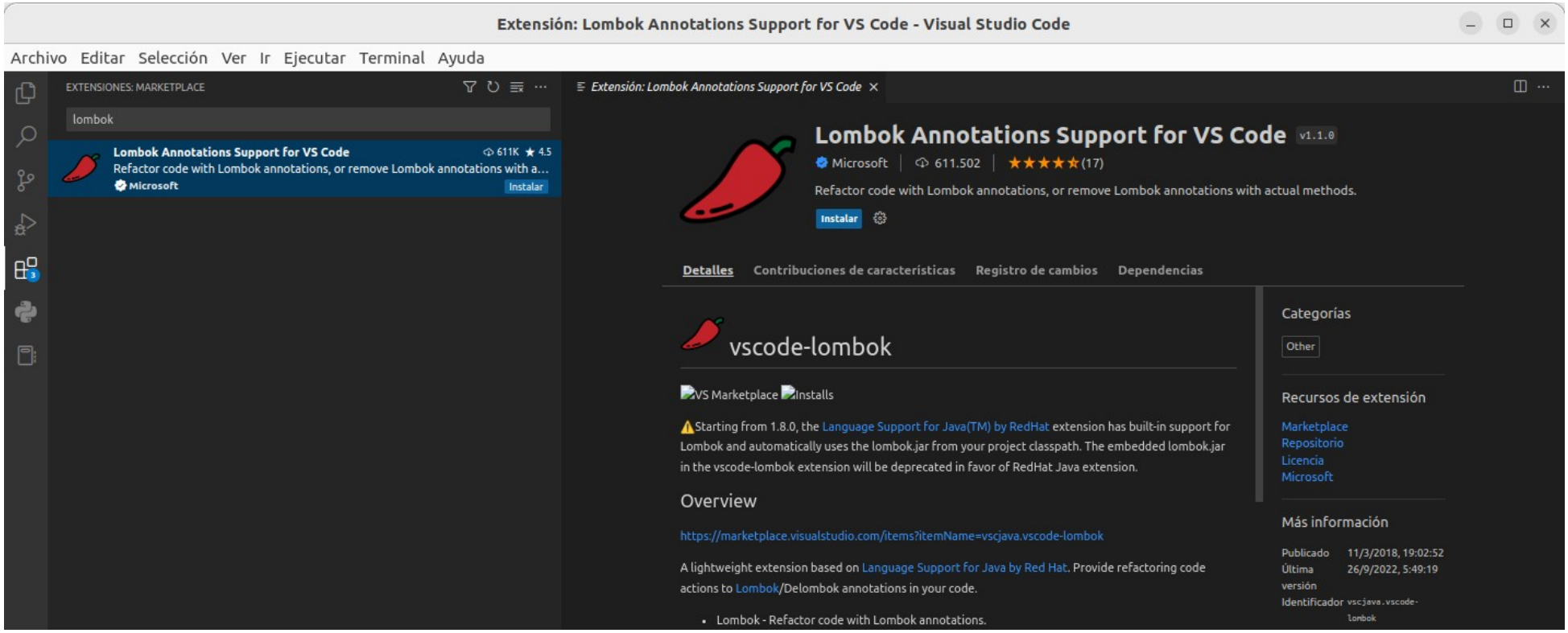


UD 3: Bases de datos y servicios REST
2.- Visual Studio Code

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

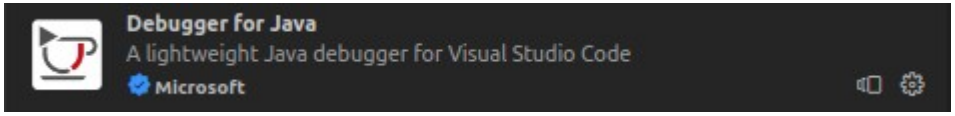


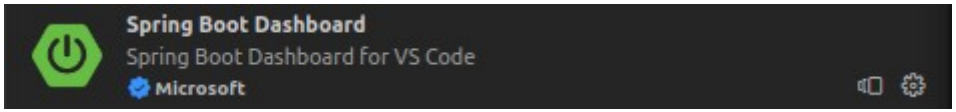
También debemos instalar la extensión “Lombok”:

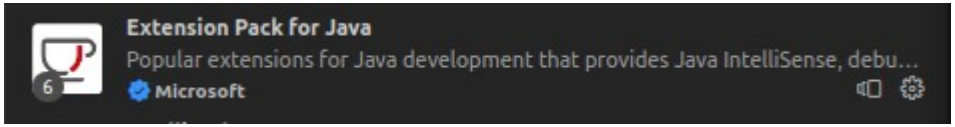


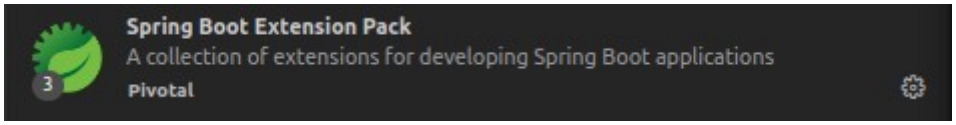
Si no tenemos las siguientes extensiones añadirlas por orden: **“Extension Pack for Java”, “Test runner for Java”, “Maven for Java”, “Tomcat” y “Project Manager for Java”.**

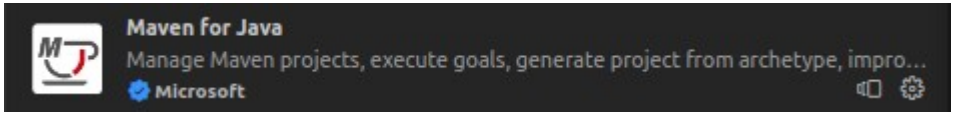
Al final como mínimo debemos tener instaladas las siguientes extensiones:
(Algunas se han autoinstalado al instalar las anteriores)

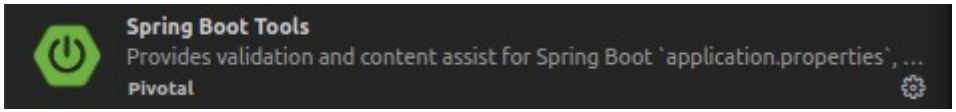


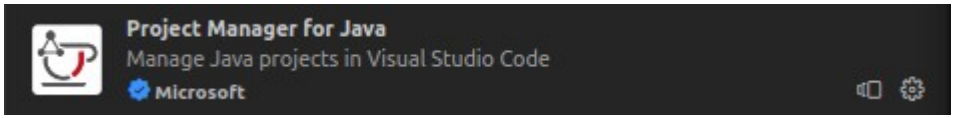


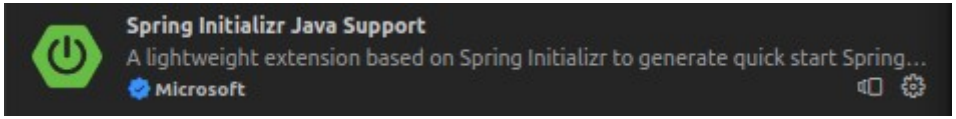












Puede que necesitemos salir del VSCode y entrar para que la lista de extensiones que acabamos de instalar en las páginas anteriores se vean.

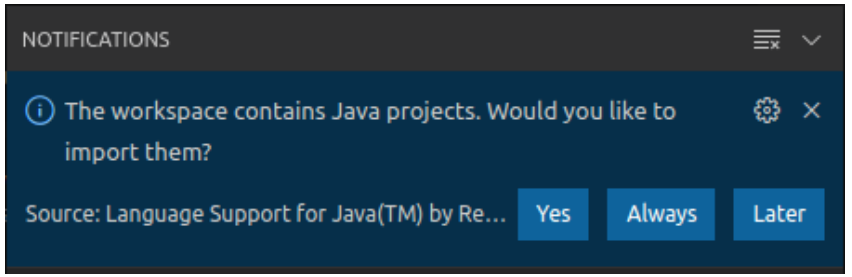
UD 3: Bases de datos y servicios REST
2.- Visual Studio Code

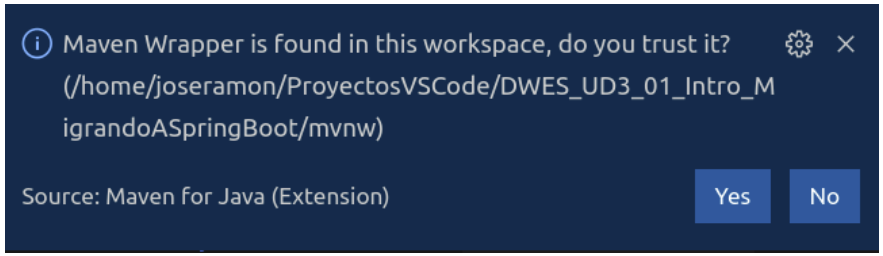
Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Importar un proyecto a Visual Studio Code:

Antes de crear un nuevo proyecto vamos a importar el proyecto de la Ud2 convertido a Spring Boot. Creamos una **carpeta nueva "ProyectosVSCode"** y copiamos el proyecto entero a esa carpeta para no estropear nada…

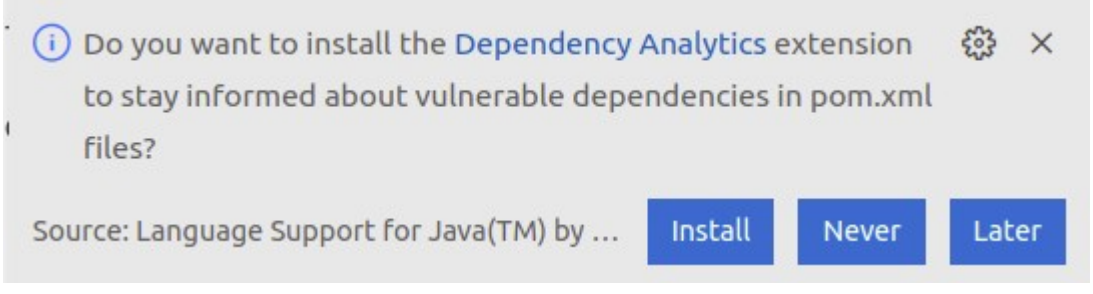
Desde VSCode abrimos la carpeta (File\Open Folder\) y nos preguntará si queremos importar el proyecto e Incorporarlo a "Java Projects" : **YES**





Si nos preguntará si confiamos en el maven detectado: **YES**

Abrimos el fichero pom.xml y dependiendo de la versión del VSCode puede que nos pregunte si queremos instalar el analizador de dependencias para detectar vulnerabilidades. Le confirmamos que queremos instalarlo:

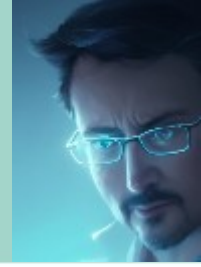


pom.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://
4      <modelVersion>4.0.0</modelVersion>
5      <parent>
6          <groupId>org.springframework.boot</groupId>
7          <artifactId>spring-boot-starter-parent</artifactId>
8          <version>2.6.2</version>
9          <relativePath/> <!-- lookup parent from repository -->
10     </parent>
11     <groupId>edu.profesor.joseramon</groupId>
12     <artifactId>dwesUd3WebAppSpringBoot</artifactId>
13     <version>0.0.1-SNAPSHOT</version>
14     <packaging>war</packaging>
15     <name>dwesUd3WebAppSpringBoot</name>
```

UD 3: Bases de datos y servicios REST
2.- Visual Studio Code

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



Antes de ejecutar nuestra aplicación, si en la práctica anterior no hemos añadido las 2 dependencias para que se ejecute la aplicación de manera independiente (sin IDE) deberemos hacerlo, porque en VSCode hace falta siempre:

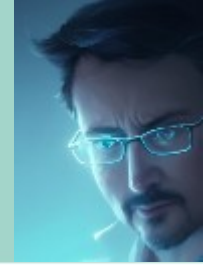
pom.xml

```
104         <groupId>org.mapstruct</groupId>
105         <artifactId>mapstruct</artifactId>
106         <version>${org.mapstruct.version}</version>
107         <scope>compile</scope>
108       </dependency>
109       <!-- Arrancar la webapp con Tomcat embebido -->
110       <!-- Need this to compile JSP -->
111       <dependency>
112           <groupId>org.apache.tomcat.embed</groupId>
113           <artifactId>tomcat-embed-jasper</artifactId>
114       </dependency>
115
116       <dependency>
117           <groupId>org.eclipse.jdt.core.compiler</groupId>
118           <artifactId>ecj</artifactId>
119           <version>4.6.1</version>
120       </dependency>
121   </dependencies>
```

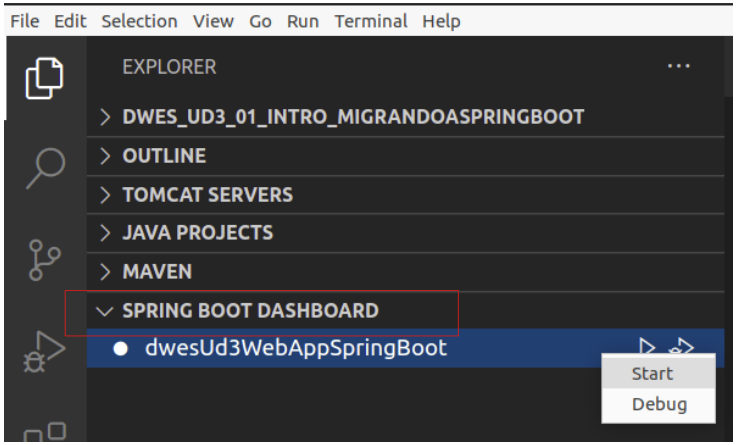
Si intentamos realizar alguna modificación, vemos que **VSCode autocompleta incluso el fichero pom.xml.** Echabamos de menos el autocompletado de VSCode...

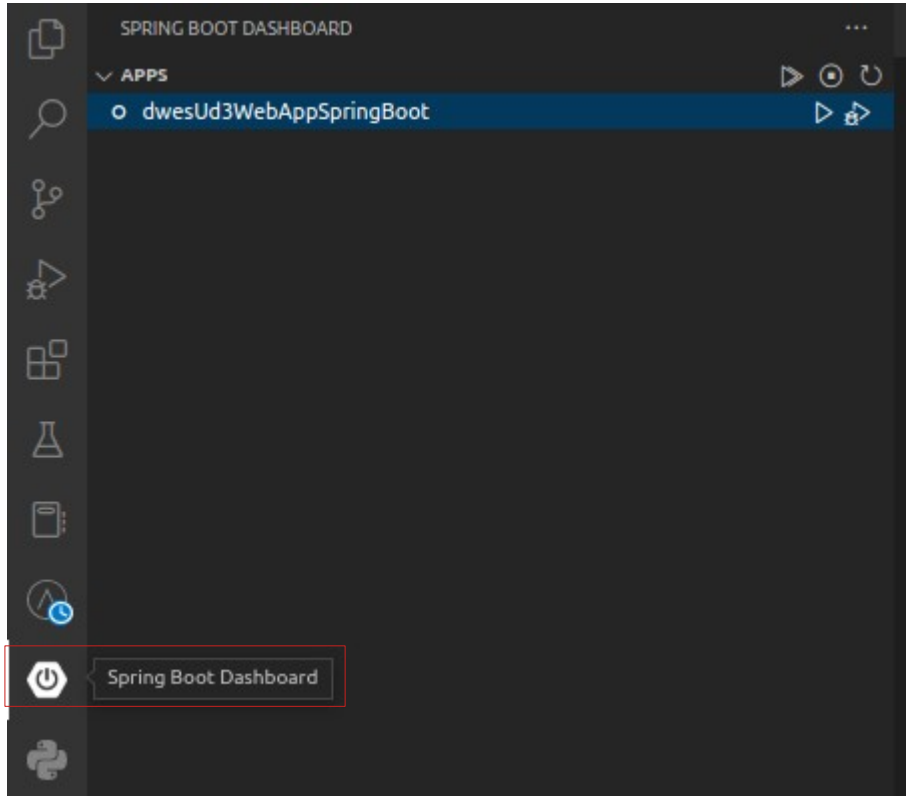
UD 3: Bases de datos y servicios REST
2.- Visual Studio Code

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

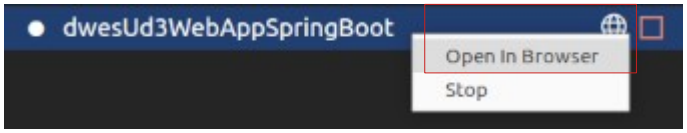


Para lanzar nuestra aplicación **1º** deberemos acceder al **Spring-Boot Dashboard** y dependiendo de nuestra versión de plugin lo tendremos como una subopción debajo de Maven o con una sección aparte en el menú lateral:



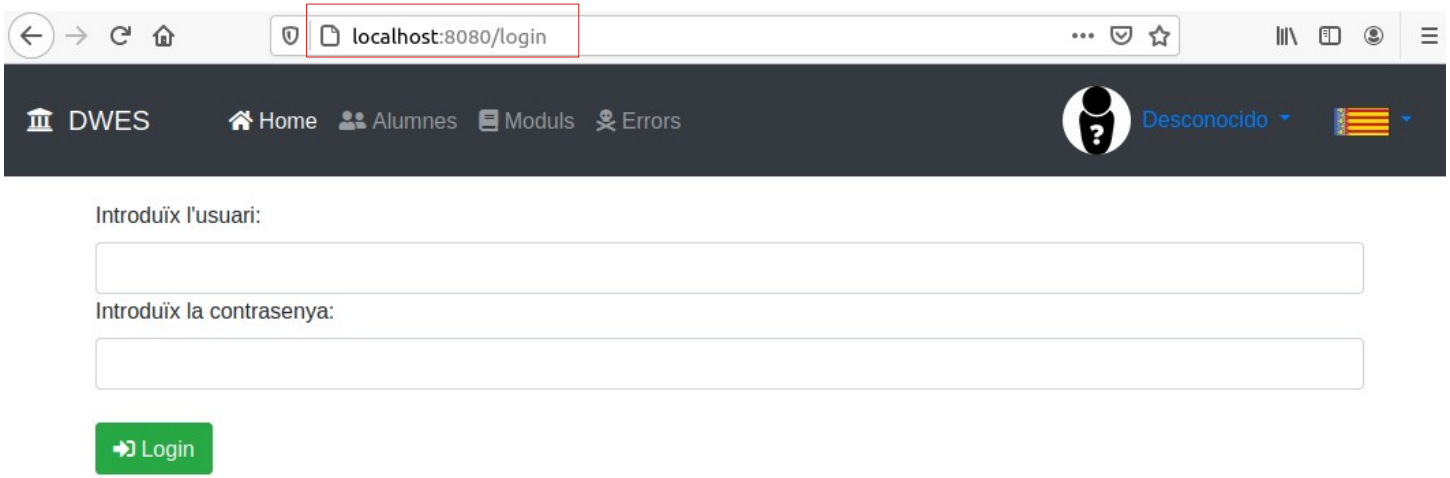


2º Pulsamos en **Start o Debug** dependiendo de lo que queramos hacer y ya tenemos nuestra web en marcha.
3º Podemos verla al pulsar botón derecho y “Open in Browser” :



Debemos darnos cuenta de **3 detalles**:

1º Al igual que pasaría si ejecutásemos nuestra aplicación en solitario **la ruta ya no contiene el nombre de la aplicación porque el Tomcat esta embebido** y solo hay una aplicación, esto significa que tenemos nuestra aplicación corriendo en la raiz de dicha url:



2º Podemos **depurar como en Eclipse** viendo los valores de las variables y expresiones (en VSCode sección “Watch”).

3º Si estamos realizando cambios y modificamos alguna clase **no hace falta recompilar la aplicación (maven clean, update e install) para ver los cambios porqué la dependencia “spring-boot-devtools” declarada en el pom.xml nos recarga la aplicación automáticamente cuando hay cambios**.

UD 3: Bases de datos y servicios REST
3.- JPA

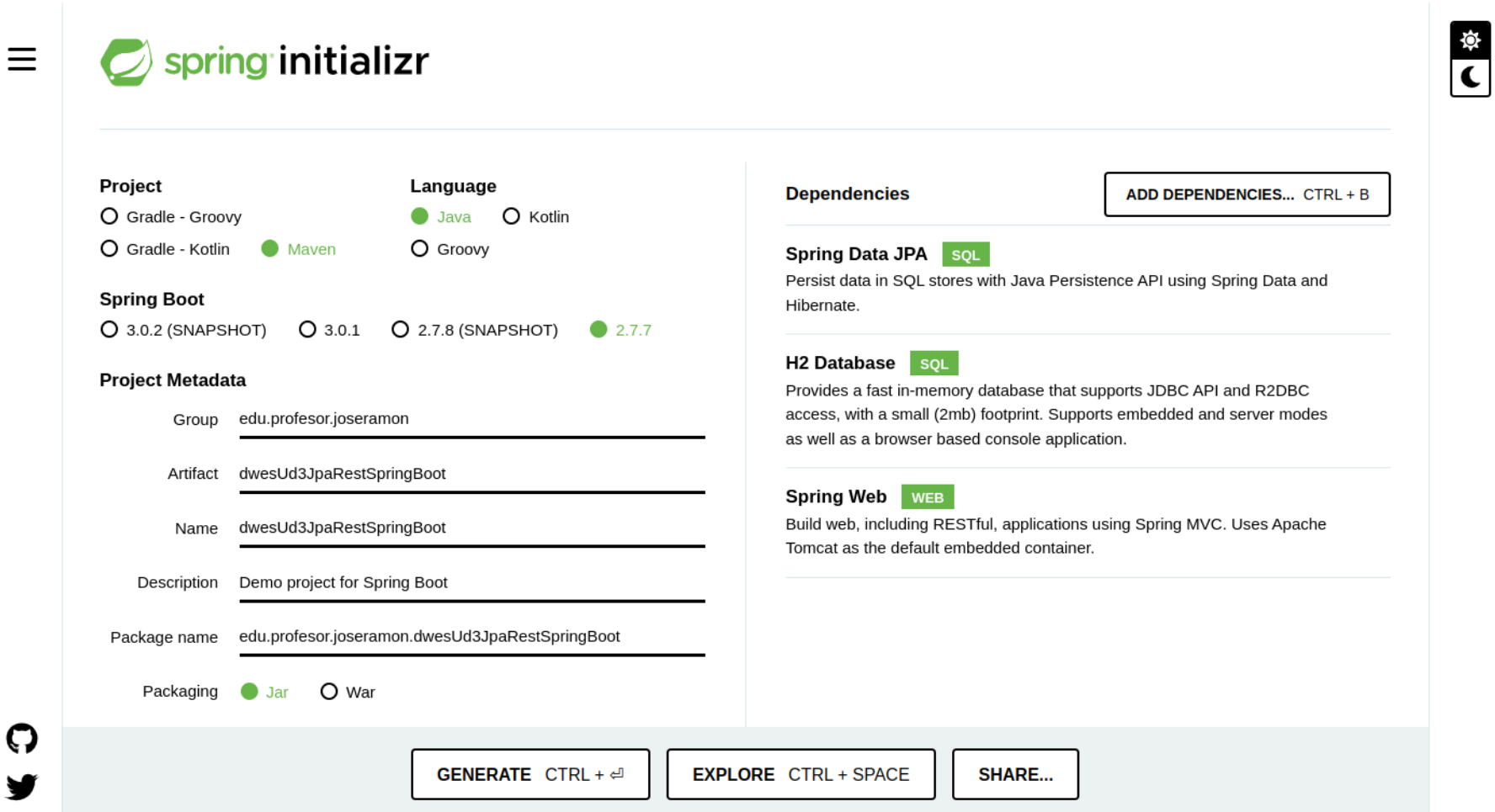
Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Nuestra primera aplicación Spring Boot con acceso a datos con JPA:

Ahora vamos a ver como podemos acceder a una base de datos desde nuestra aplicación. Por claridad, vamos a *crear una aplicación nueva Spring Boot desde Visual Studio* para la explicación.

¿ Por donde empezamos ?

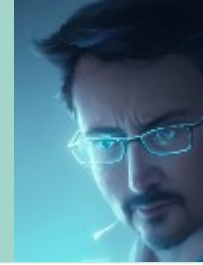
1º opción: Acceder a http://start.spring.io para crear el proyecto maven de una manera rápida escogiendo SOLO la dependencia JPA, H2 y Web. ***No vamos a escoger esta, sino la opción 2.***



2º opción : Esta es la opción que escogeremos para crear nuestra primera App "dwes_ud3_primer_jpa" directamente desde **VSCode**:

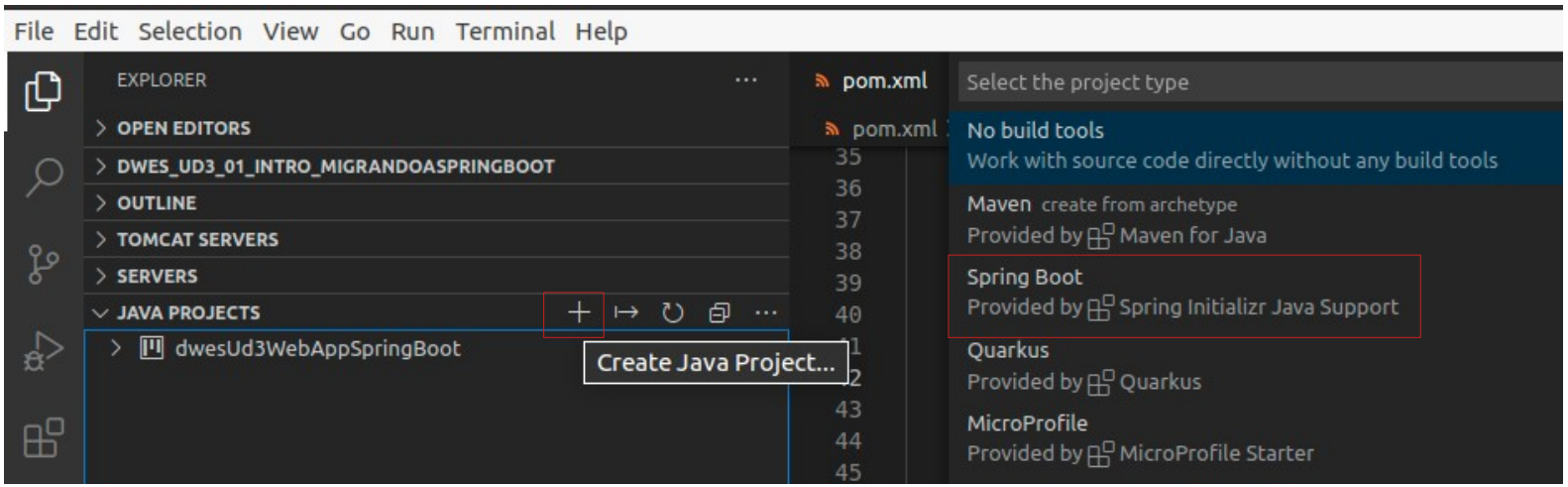
Para ello seguiremos estos **pasos** en las siguientes páginas:

1º Crear el proyecto Spring Boot
2º Configurar la Base de Datos
3º Crear las entidades JPA
4º Inicializar con datos la BD
5º Crear el repositorio para cada entidad JPA
6º Configurar la App para que se pueda acceder por linea de comandos
7º Pruebas

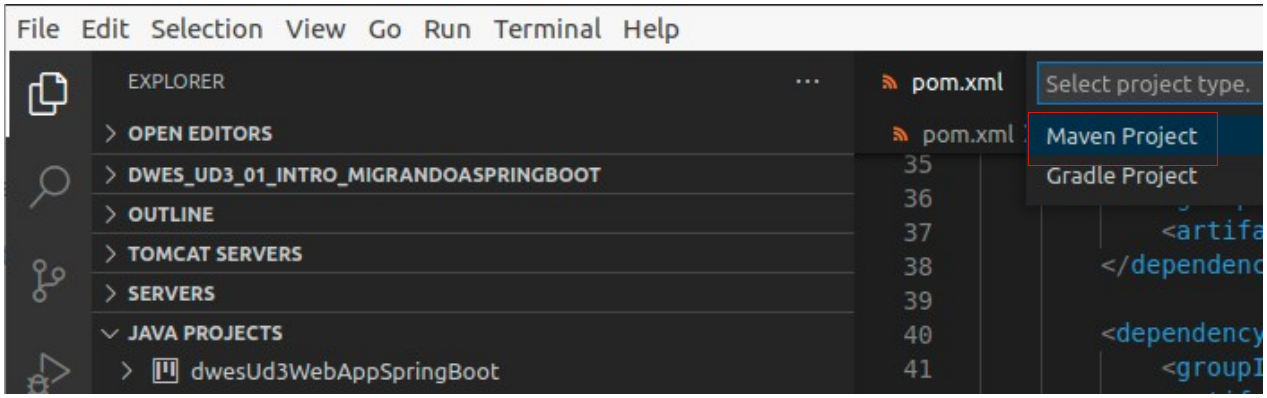


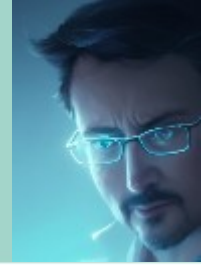
1º Crear el proyecto Spring Boot:

Para crear nuestro primer proyecto Spring Boot desde VSCode nos situamos en la pestaña “Java Projects” , pulsamos el “+” y luego “Spring Boot” :

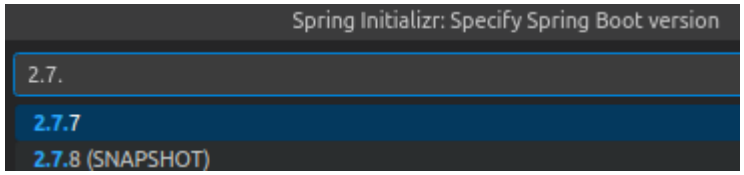


A continuación debemos indicar que es un proyecto “Maven”:

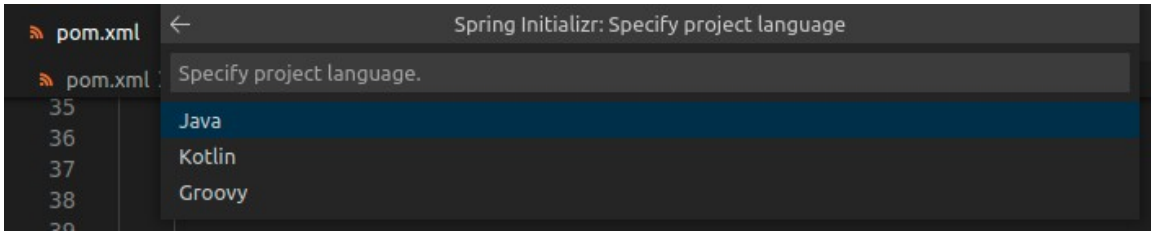




Nos pide la versión y le indicamos **2.7.7**:



Nos pide el lenguaje y le decimos Java:

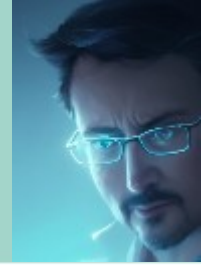


Y de igual manera le introducimos :

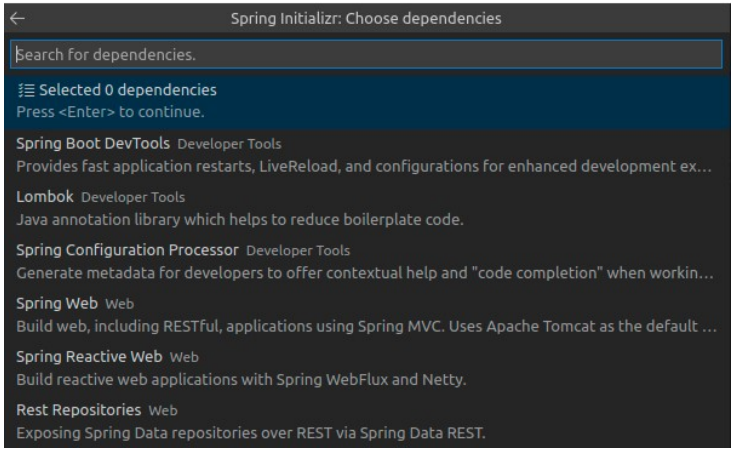
Group Id: edu.alumno.NombreAlumno
Artifact Id: dwes_ud3_primer_jpa
Package: jar
Versión de Java: 11

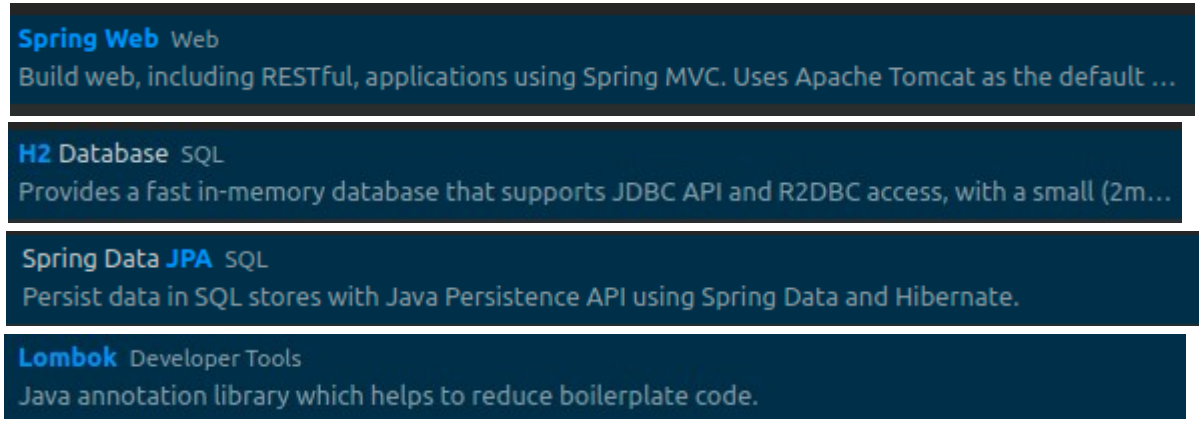
UD 3: Bases de datos y servicios REST
3.- JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



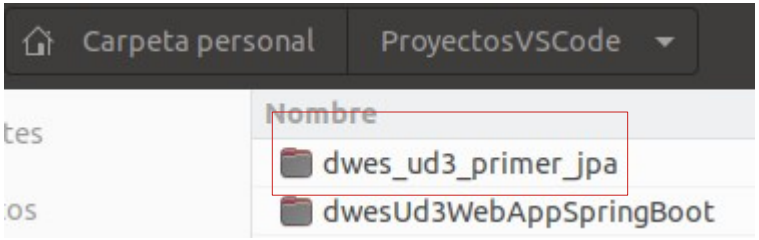
Ahora hay que introducir las dependencias



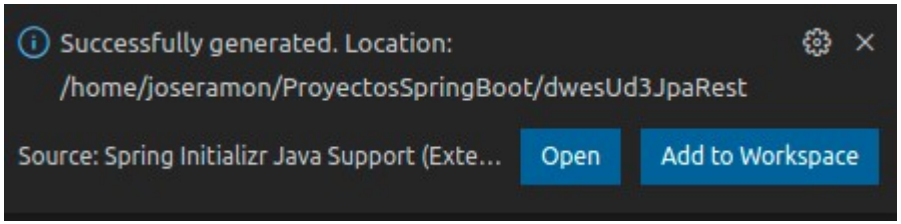


Deberemos seleccionar solo “Spring Web”, “H2”, JPA” y “Lombok”.

Una vez seleccionados le damos INTRO y decimos que lo cree en la carpeta “ProyectosVSCode” porque VSCode creará automáticamente una carpeta dentro cuyo nombre sea el nombre el “artifact id”.



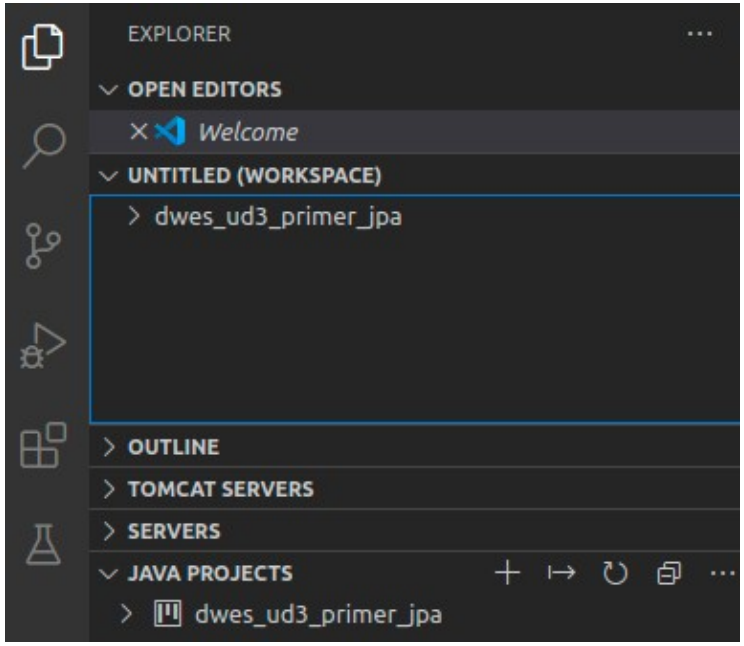
Tras generar el proyecto nos pregunta si queremos añadirlo al workspace y le decimos "Add to Workspace":



El workspace detecta un nuevo proyecto y para que no nos lo pida siempre le deicmos que siempre importe los proyectos (Always):

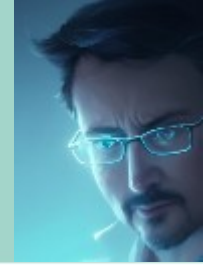
Y ya podemos ver nuestro proyecto en el workspace, podemos añadir más proyectos y lanzar cualquiera de ellos :

Ayuda: Si en la pestaña "Spring Boot DashBoard" no vemos los proyectos nuevos hará falta salir del VSCode y volver a entrar para que actualice la lista.

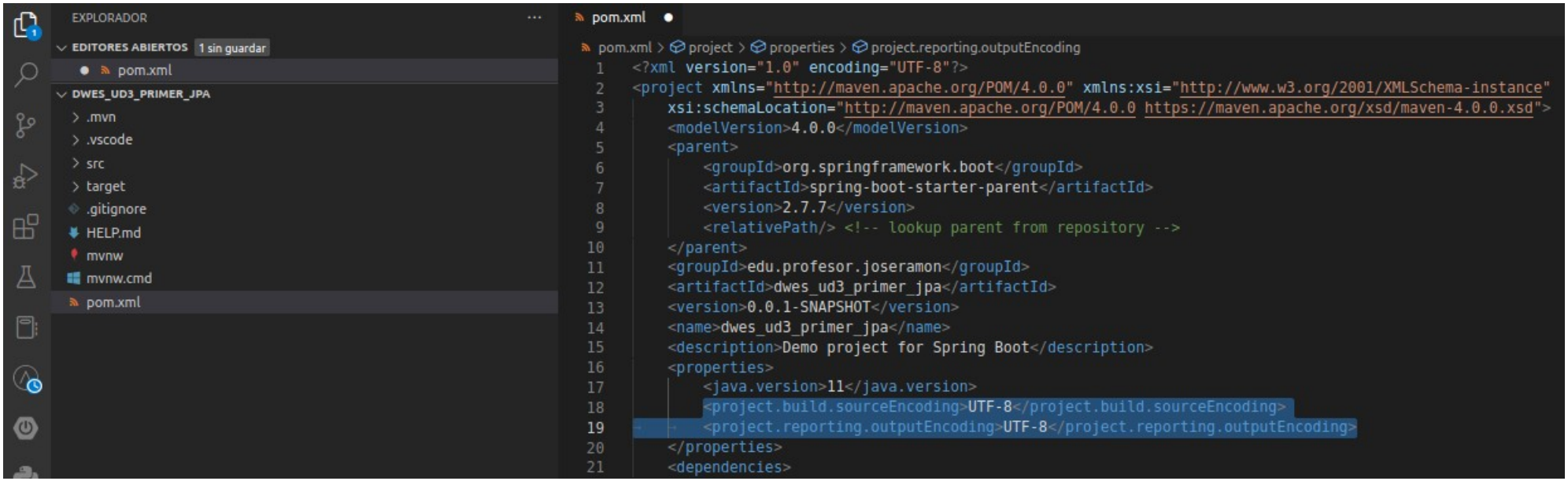


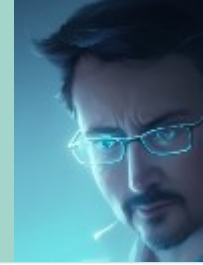
UD 3: Bases de datos y servicios REST
3.- JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

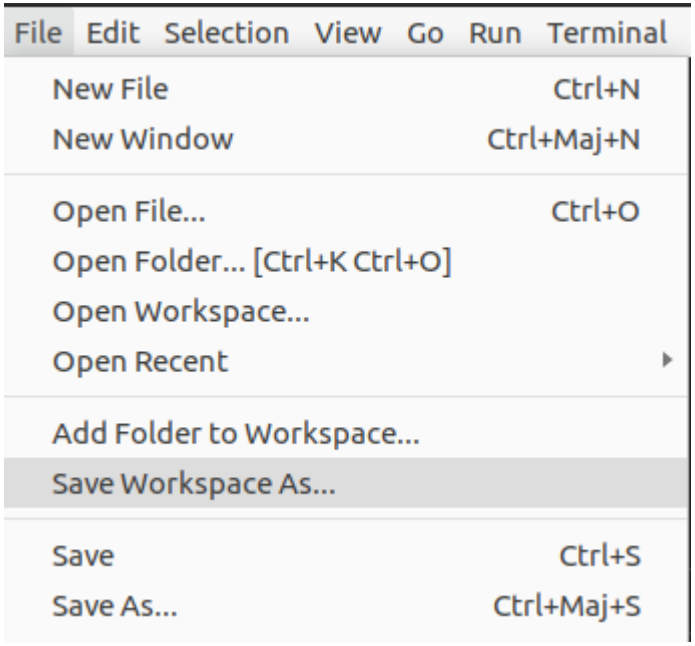


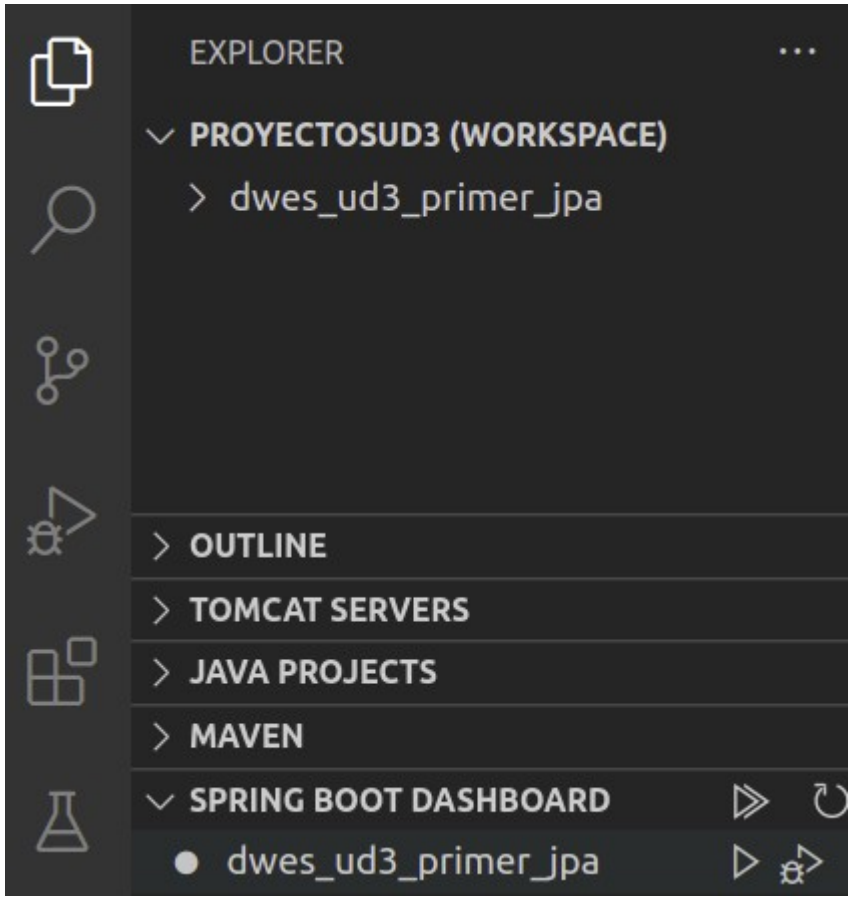
De manera opcional es una buena costumbre añadir al fichero pom del proyecto el hecho de que utilizamos encoding UTF-8:





Por último, no esta de más que procedamos a cambiar el nombre al WorkSpace de Visual Studio Code:

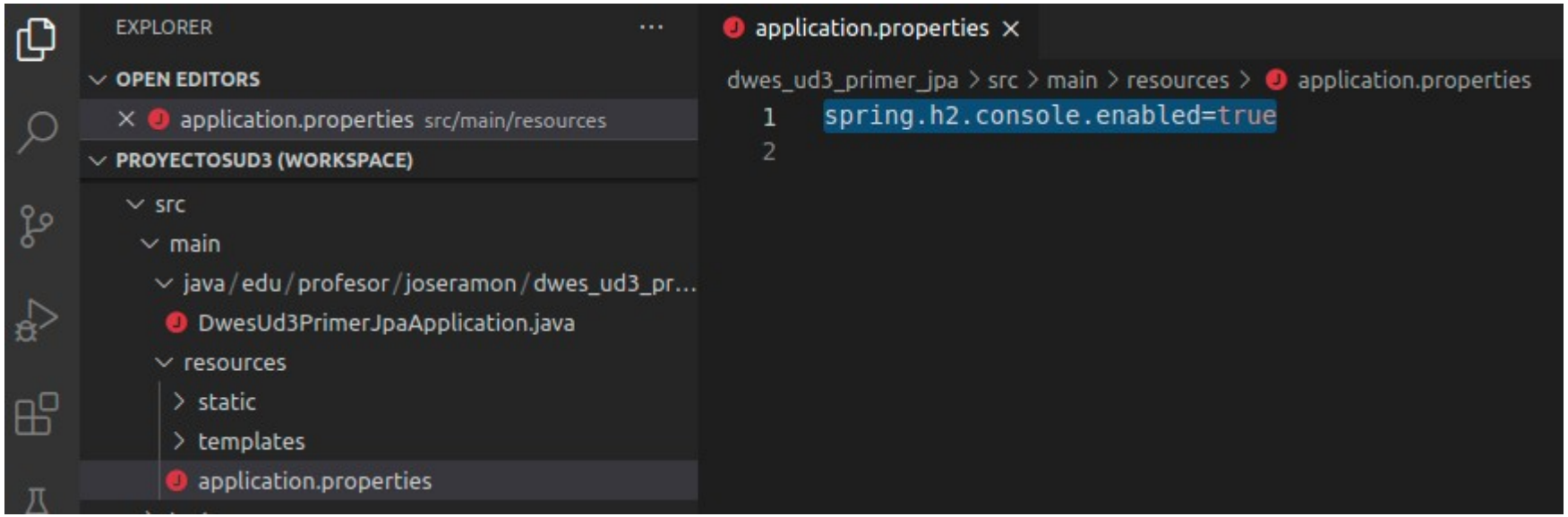




2º Configurar la Base de Datos:

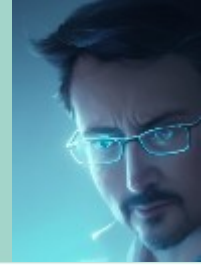
H2 es una Base de Datos en memoria. Esto quiere decir que cuando paremos la aplicación los datos desaparecerán, lo que lo convierte en la base de datos ideal durante las pruebas que realizamos en desarrollo.

Si queremos tener **acceso vía web a la base de datos H2** aparte de haber configurado la dependencia H2 en el fichero pom.xml debemos configurar en el fichero de propiedades de la aplicacación que queremos activar la consola h2. Para ello debemos añadir la linea que activa la consola en “src\main\resources”:



UD 3: Bases de datos y servicios REST
3.- JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



Sin embargo, antes de acceder vía web debemos **configurar la Base de Datos H2**.

Añadimos los parámetros de configuración de la Base de Datos H2 y habilitamos las estadísticas de Hibernate copiando el código del DRIVE:

Acordaros de desactivar las estadísticas de Hibernate si ponemos la aplicación en producción.

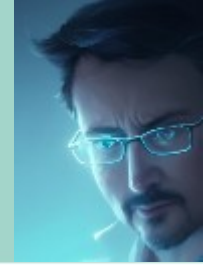
application.properties

src > main > resources > application.properties

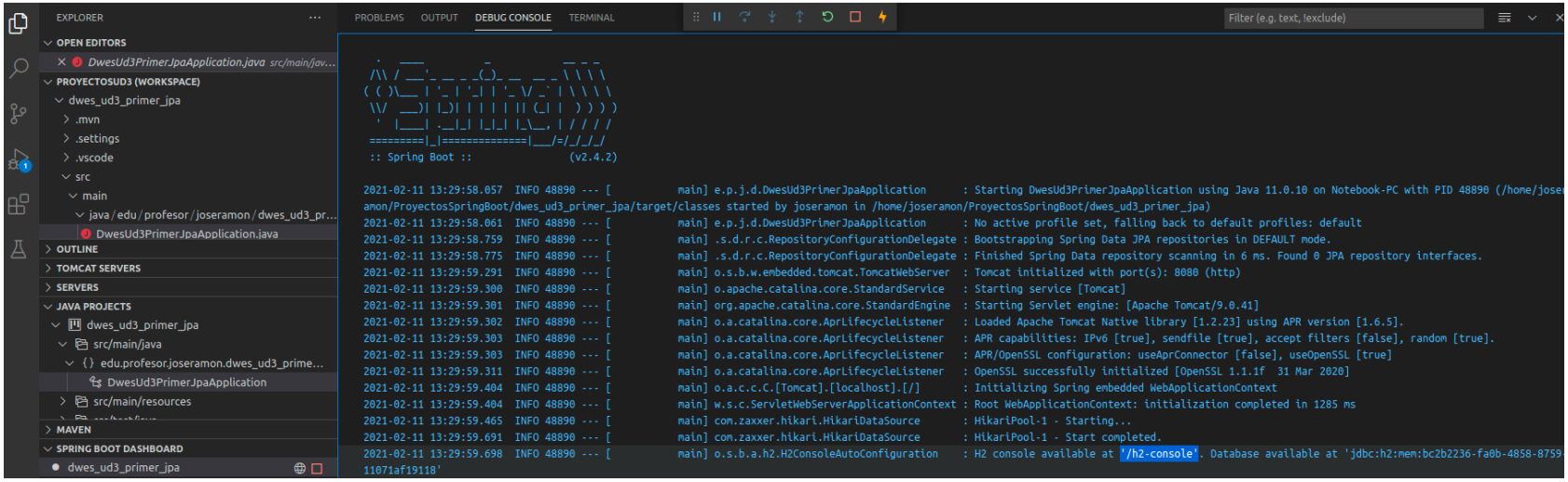
```
#Activar consola para acceder a la BD H2 via navegador
# localhost:puertoConfigurado/h2-console
spring.h2.console.enabled=true
#Configuración de la BD H2
spring.datasource.url=jdbc:h2:mem:testDwesDb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=dwes
spring.datasource.password=Simarro@1
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

#En la versión 2.4.2 no hace falta, pero en la
# 2.6.2 hace falta para poder hacer inserts en data.sql
spring.jpa.hibernate.ddl-auto=none

#CONFIGURACIÓN SOLO durante las pruebas:
# Habilitar estadísticas hibernate
spring.jpa.properties.hibernate.generate_statistics=true
# Habilitar LOGGER de las estadísticas de hibernate
logging.level.org.hibernate.stat=
# Mostrar que consultas esta realizando Hibernate
spring.jpa.show-sql=true
# Formatear las consultas
spring.jpa.properties.hibernate.format_sql=true
# Mostrar los parametros que estan enviandose a las consultas
logging.level.org.hibernate.type=debug
#FIN CONFIGURACIÓN SOLO durante las pruebas
```

UD 3: Bases de datos y servicios REST
3.- JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



Si arrancamos el proyecto podemos ver que Spring Boot ha activado la ruta /h2.console para acceder a la Base de datos H2:

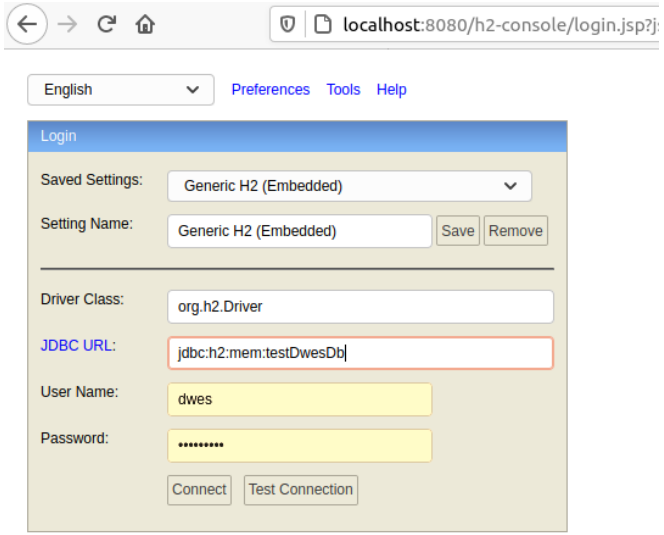


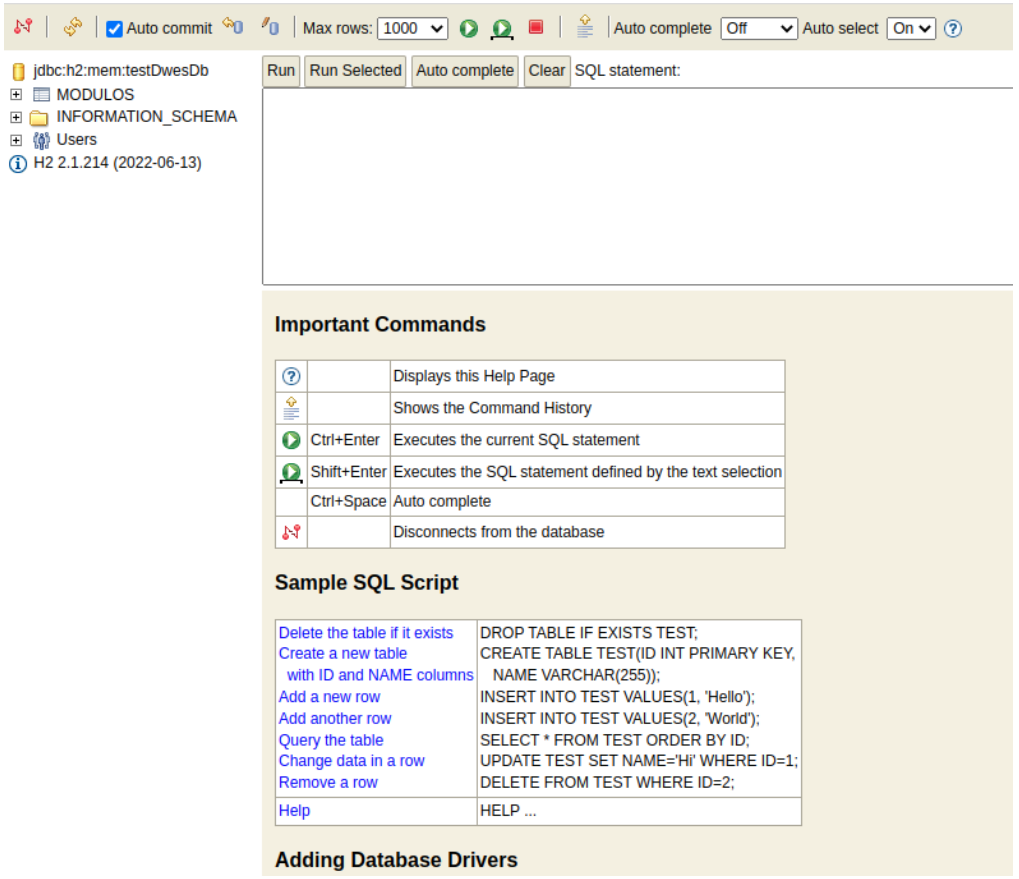
Si no tenemos posibilidad de tener una Base de Datos como MySql, Sql Server, Oracle o PostGres podemos optar por persistir los datos a fichero, en vez de en memoria (para que se guarden los datos) y para ello cambiaríamos la configuración de la url en application.properties:

spring.datasource.url=jdbc:h2:file:/rutaDatos/NombreFicheroDatos

Si introducimos en el navegador la ruta localhost:8080/h2-console podemos ver que tenemos acceso a la Base de Datos H2.

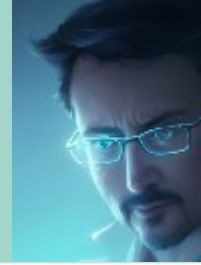
Deberemos introducir la url JDBC, el usuario y password configurados en application.properties :

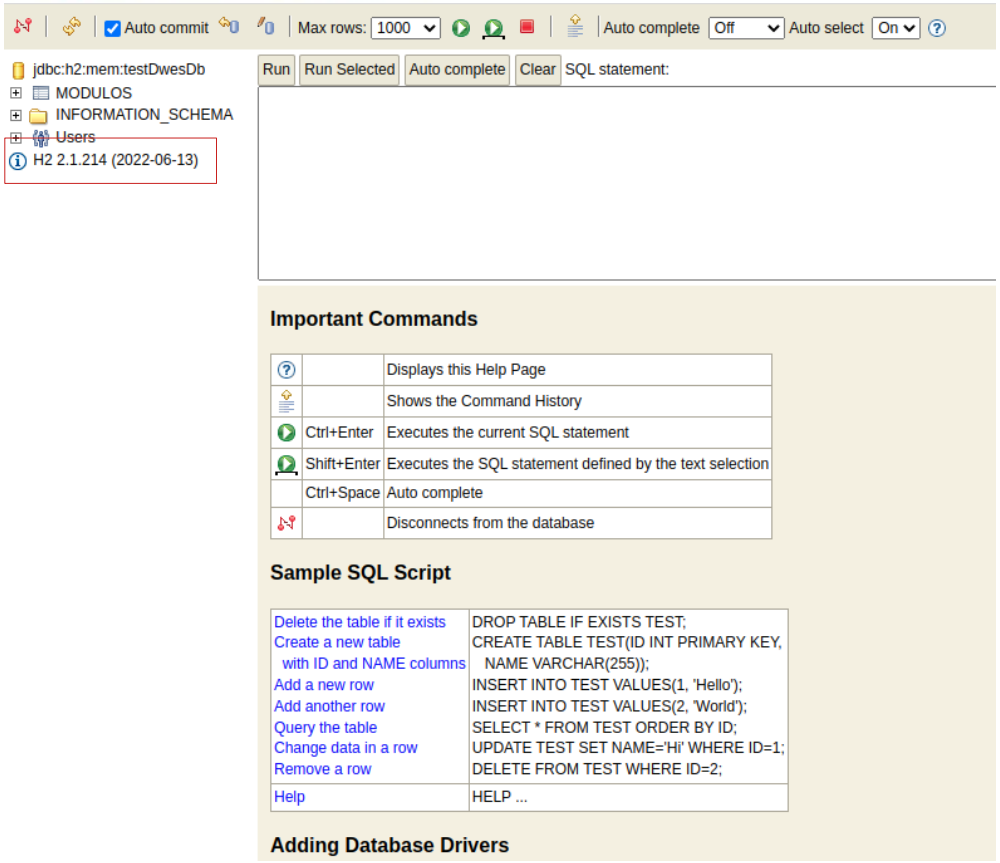




UD 3: Bases de datos y servicios REST
3.- JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



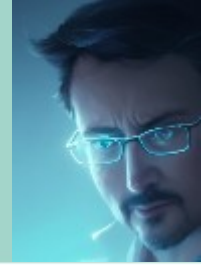


Por defecto, al crear el proyecto e indicarle que queremos la dependencia H2Database no se incluye el número de versión en el pom.

Debemos asegurarnos que la versión H2 es la 2.1.214 o superior o puede que no inserte correctamente los registros que se autoincremente la clave primaria.

Si no tenemos una versión igual o superior tendremos que modificar el fichero pom.xml para añadir la versión.

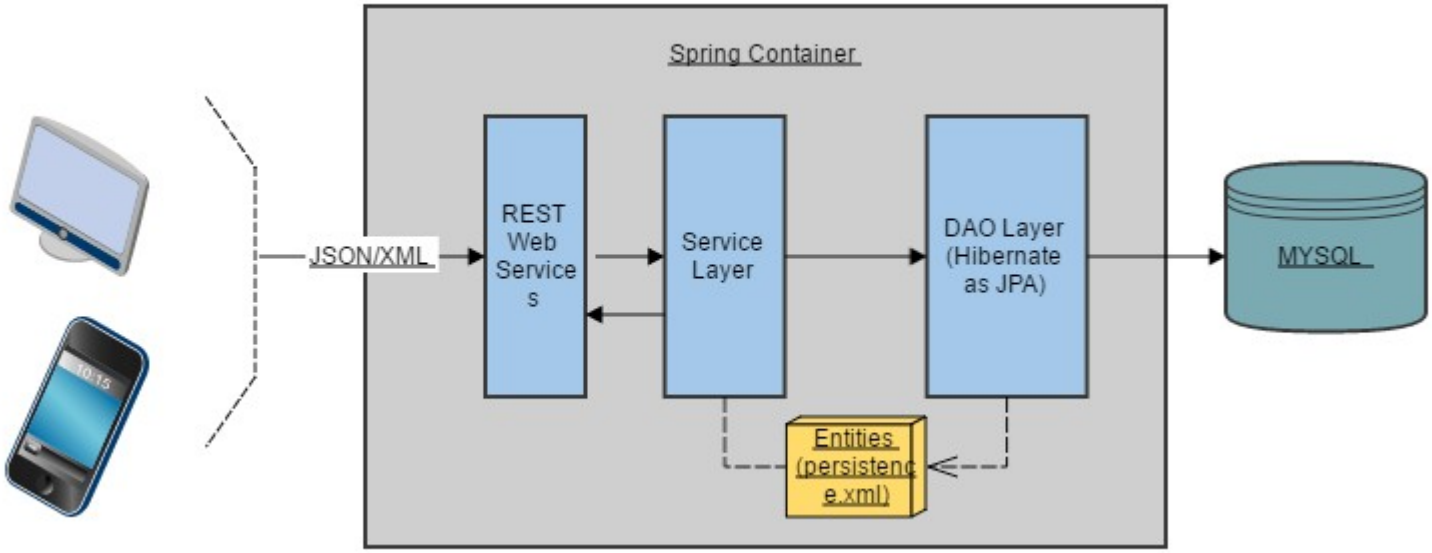
```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>2.1.214</version>
    <scope>test</scope>
</dependency>
```



3º Crear las entidades JPA:

JPA es el acrónimo de **Java Persistence Api**. JPA nos permite olvidarnos de como acceder a la Base de datos y como gestionarla porque Spring lo hace por nosotros.

Esta abstracción se consigue gracias al **DAO** (Data Access Object) que nos permite el Mapeo Objeto Relacional **ORM** (Object Relational Maping).

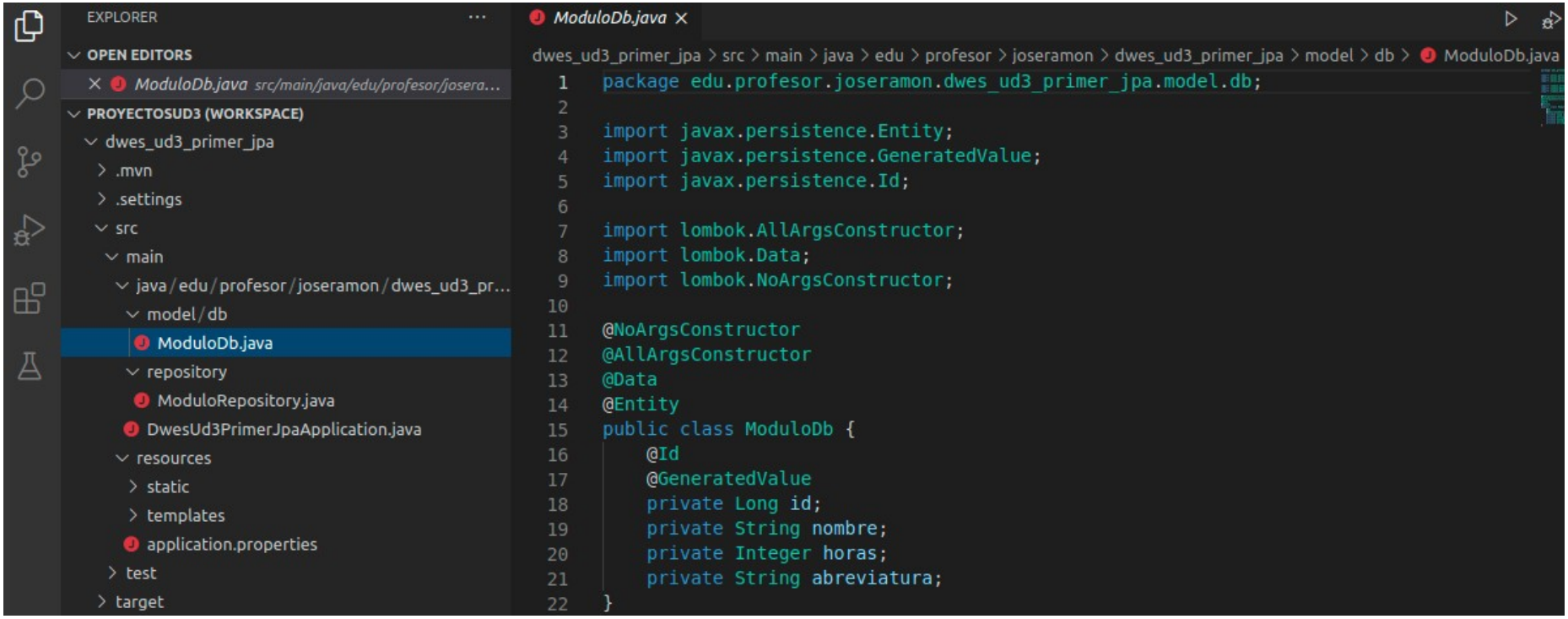


UD 3: Bases de datos y servicios REST
3.- JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Empecemos por **crear la entidad ModuloDb** en el paquete **“model.db”**. Para ello copiaremos SOLO los siguientes atributos de la clase “Modulo.java” del proyecto de la Ud2 y le añadiremos los atributos y anotaciones para que tenga el siguiente código:

Fijate que hemos tenido que cambiar **id** para que sea de **tipo Long** y permitir más valores que un tipo Integer.



¿ Que significado tienen las anotaciones nuevas ?

Lombok:

```
@NoArgsConstructor
@AllArgsConstructor
@Data
@Entity
public class ModuloDb {
    @Id
    @GeneratedValue
    private Long id;
    private String nombre;
    private Integer horas;
    private String abreviatura;
}
```

@**NoArgsConstructor:** Crea un constructor sin parámetros.

@**AllArgsConstructor:** Crea un constructor con todos los parámetros.

@**Data:** Equivale a poner TODAS las anotaciones siguientes:
@Getter, @Setter, @ToString, @EqualsAndHashCode y @RequiredArgsConstructor(constructor solo con parámetros que no pueden ser nulos).

Persistencia (Acceso a datos):

@**Entity:** Permite definir el bean ModuloDb como una entidad que está en la BD.

@**Id:** Permite definir la clave primaria de la tabla relacional.

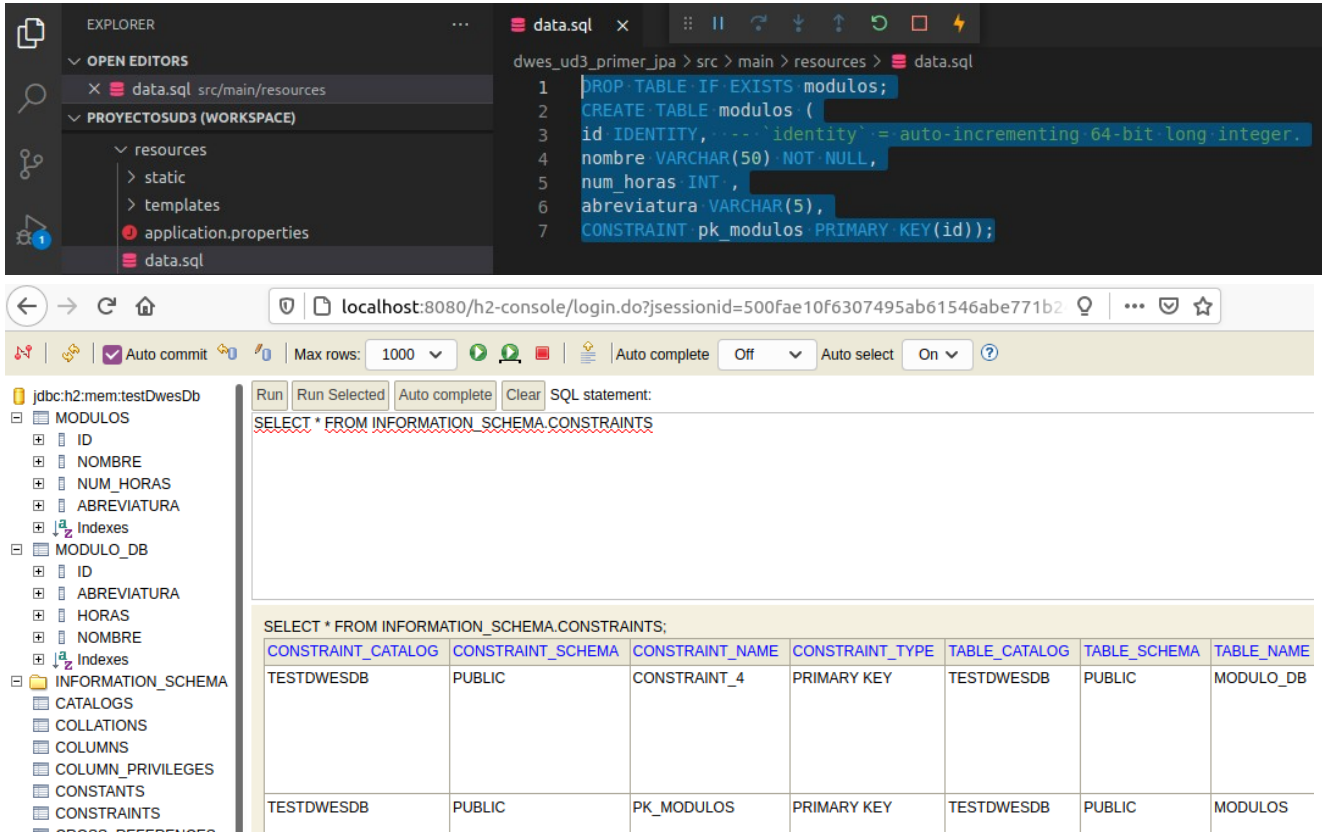
@**GeneratedValue:** Permite indicar que el id se generará automáticamente desde la Bd.

UD 3: Bases de datos y servicios REST
3.- JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

4º Inicializar con datos la BD:

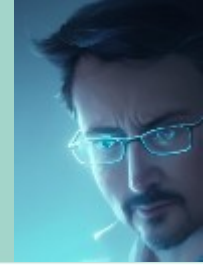
Para **inicializar la Base de Datos H2** hay que crear el fichero “data.sql” en la carpeta **“src\main\resources”** y escribir la sentencia para crear la tabla. Arranca la webapp para comprobar el funcionamiento:



¿Por tenemos “MODULOS” y “MODULOS_DB” si arrancamos la webapp?

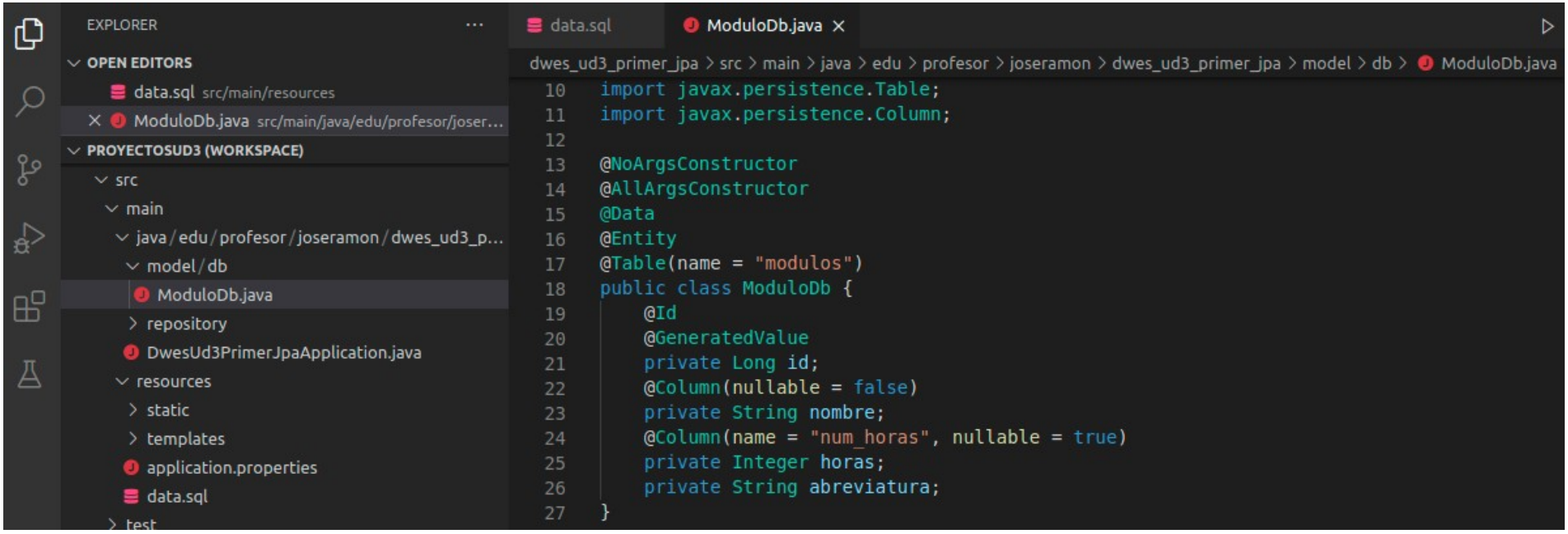
UD 3: Bases de datos y servicios REST
3.- JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

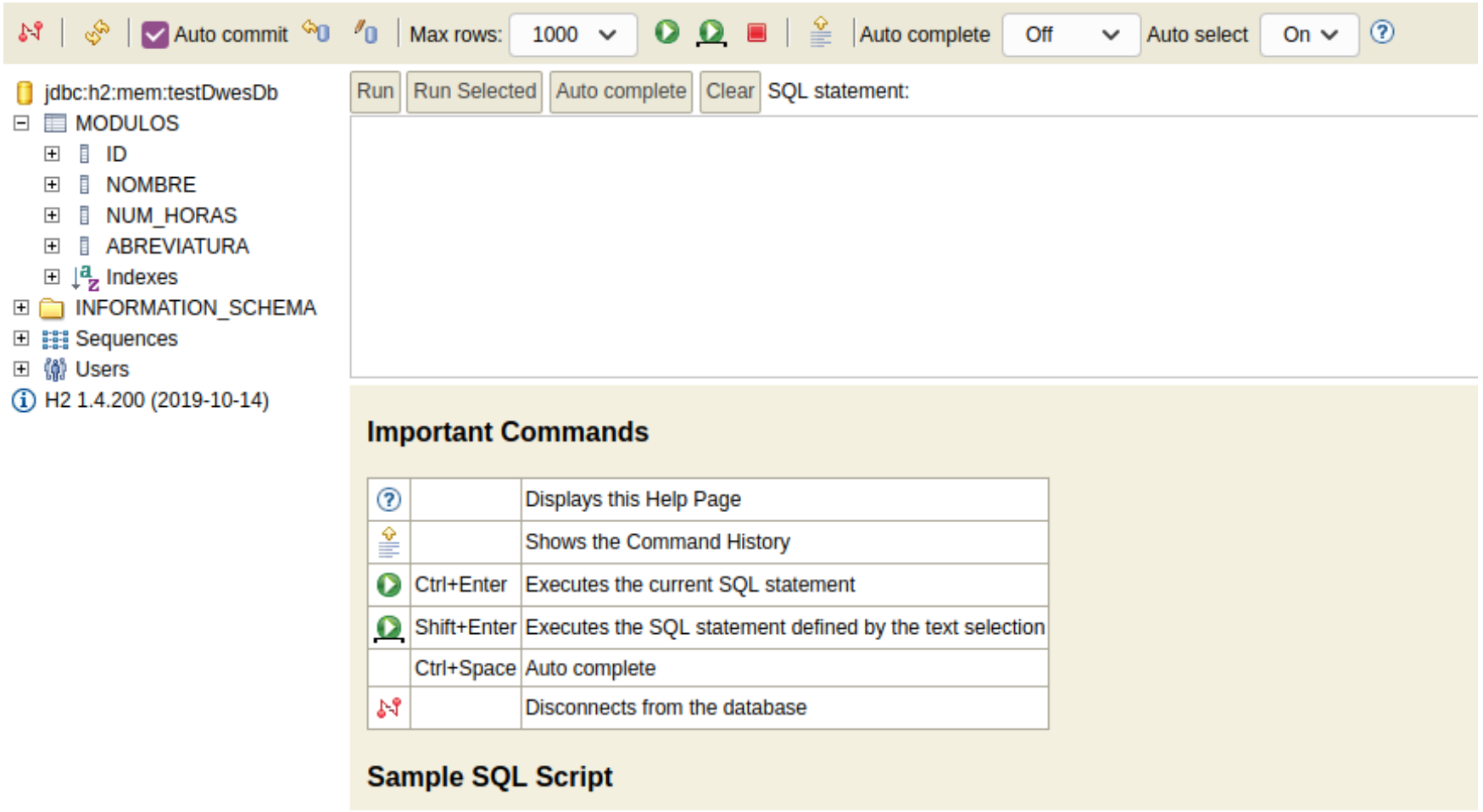


La Base de datos H2 es tán “lista” que cuando hemos configurado ModuloDb.java, al decirle que es una entidad ha creado la tabla por nosotros. Si la clase se hubiera llamado igual que la tabla no se habría duplicado.

Si queremos **tener más control** sobre la creación o si queremos hacer referencia a una tabla existente en una Base de Datos deberemos de añadir la notación **@Table** para indicar la tabla de la Base de Datos a la que se está haciendo referencia. Adicionalmente podemos también decirle el nombre del atributo en la tabla (si varia del que tenemos en la clase) con la notación **@Column** o incluso indicarle si permite valores nulos o no. Realiza los siguientes cambios:



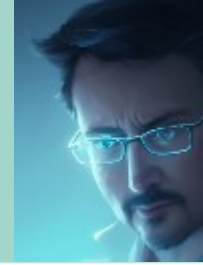
Si arrancamos solo veremos ya una tabla:



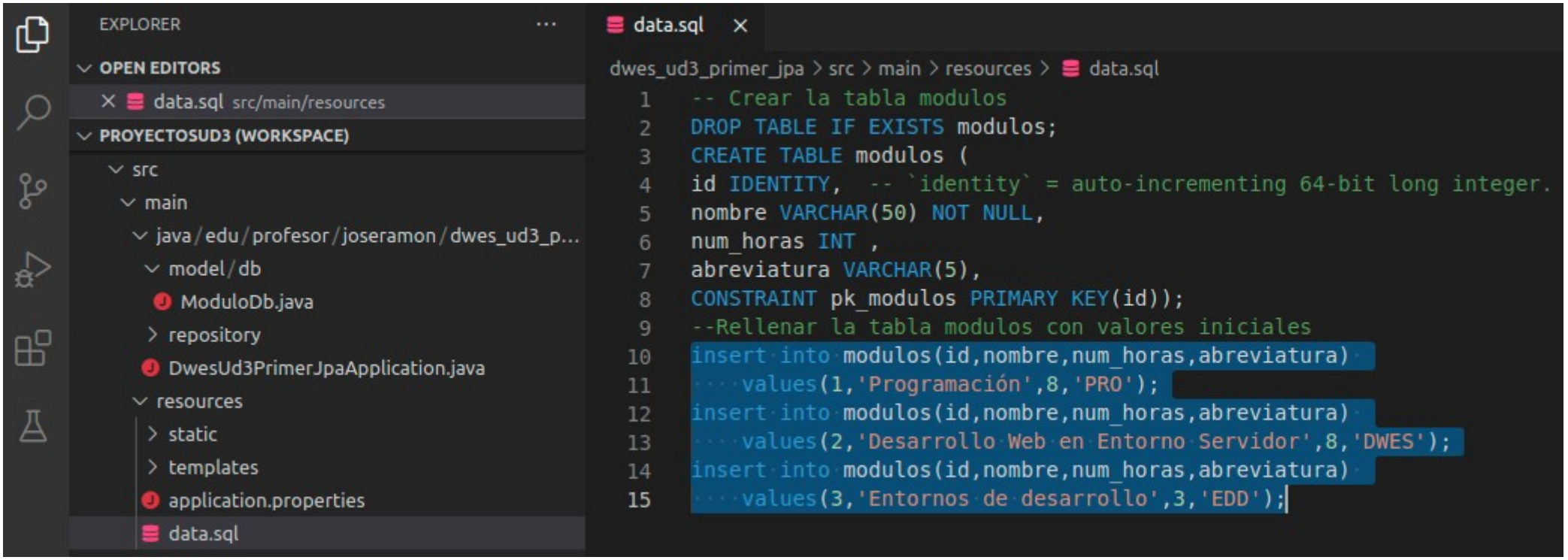
Los datos se borran al apagar la aplicación al estar configurado como una BD en memoria pero, ¿Como inicializamos las tablas para que tengan los mismos datos cada vez que arranque?

UD 3: Bases de datos y servicios REST
3.- JPA

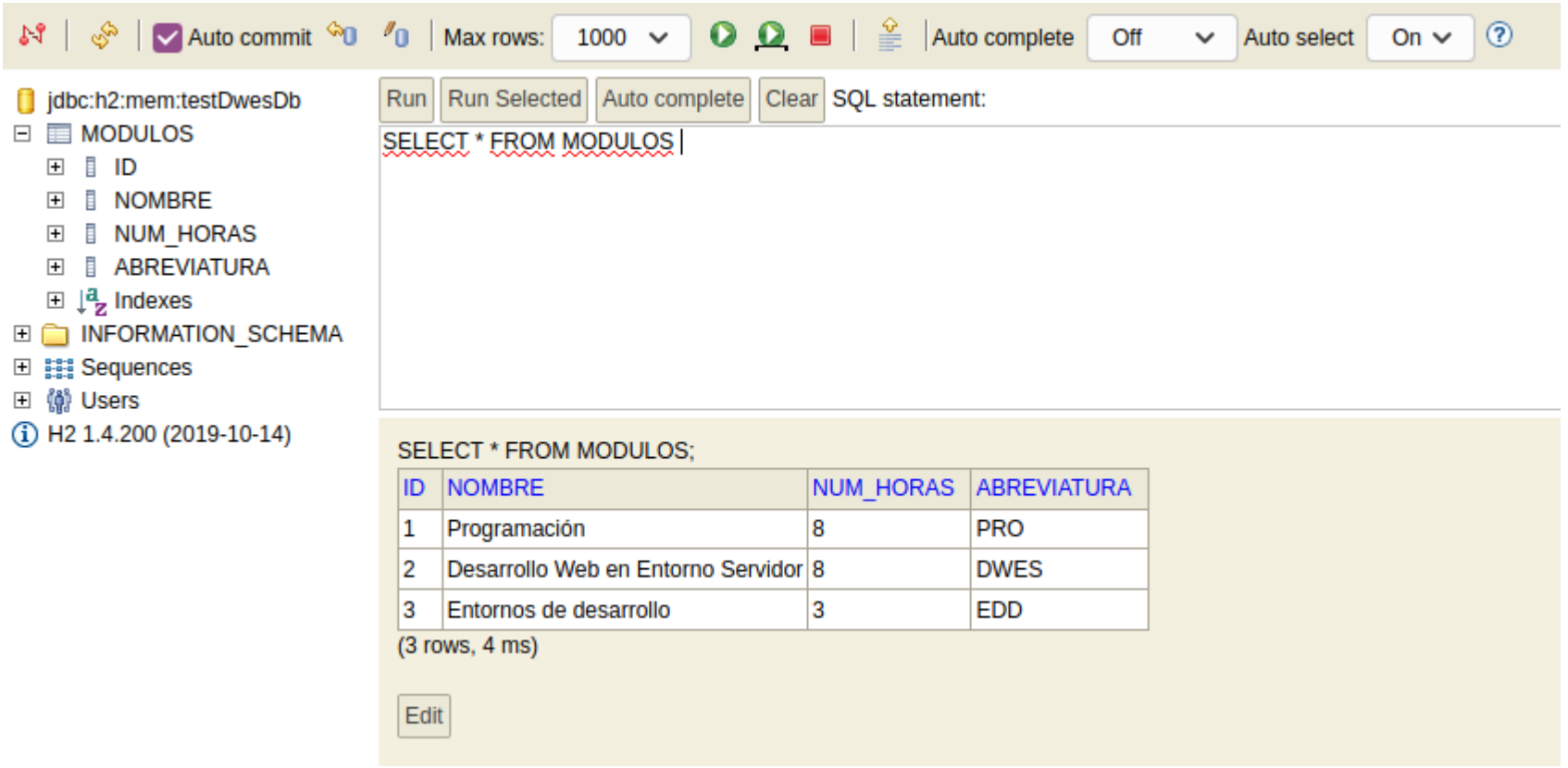
Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

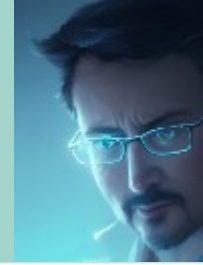


Para **inicializar con datos la BD H2** hay que rellenar el fichero “data.sql” en la carpeta “src\main\resources” y además de crear las tablas crear las instrucciones con los inserts:



Podemos **comprobar los datos que tiene la BD H2** . Para ello rellenar la sentencia SQL necesaria:





5º Crear el repositorio para cada entidad JPA:

Para hacer **CRUD (Create Read Update Delete)** sobre la **tabla "Modulo"** Spring pone a nuestra disposición la interface **JpaRepository**.

Crear la interface ModuloRepository en el package respository:

UD 3: Bases de datos y servicios REST
3.- JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

La **utilidad** de la interfaz **JpaRepository** es doble:

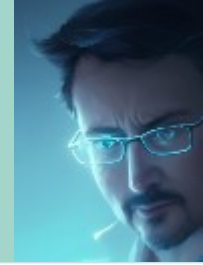
1º Por el hecho de heredar de JpaRepository ya tenemos unos métodos que nos dan un *método sencillo para acceder y manipular los datos* de cualquier tabla.

2º Los métodos que proporciona esta interface se convierten en un *standard de acceso y manipulación de registros en una tabla*.

Al acceder a cualquier tabla desde su repositorio que hereda de JpaRepository se utilizarán los mismos nombres para hacer las mismas operaciones.

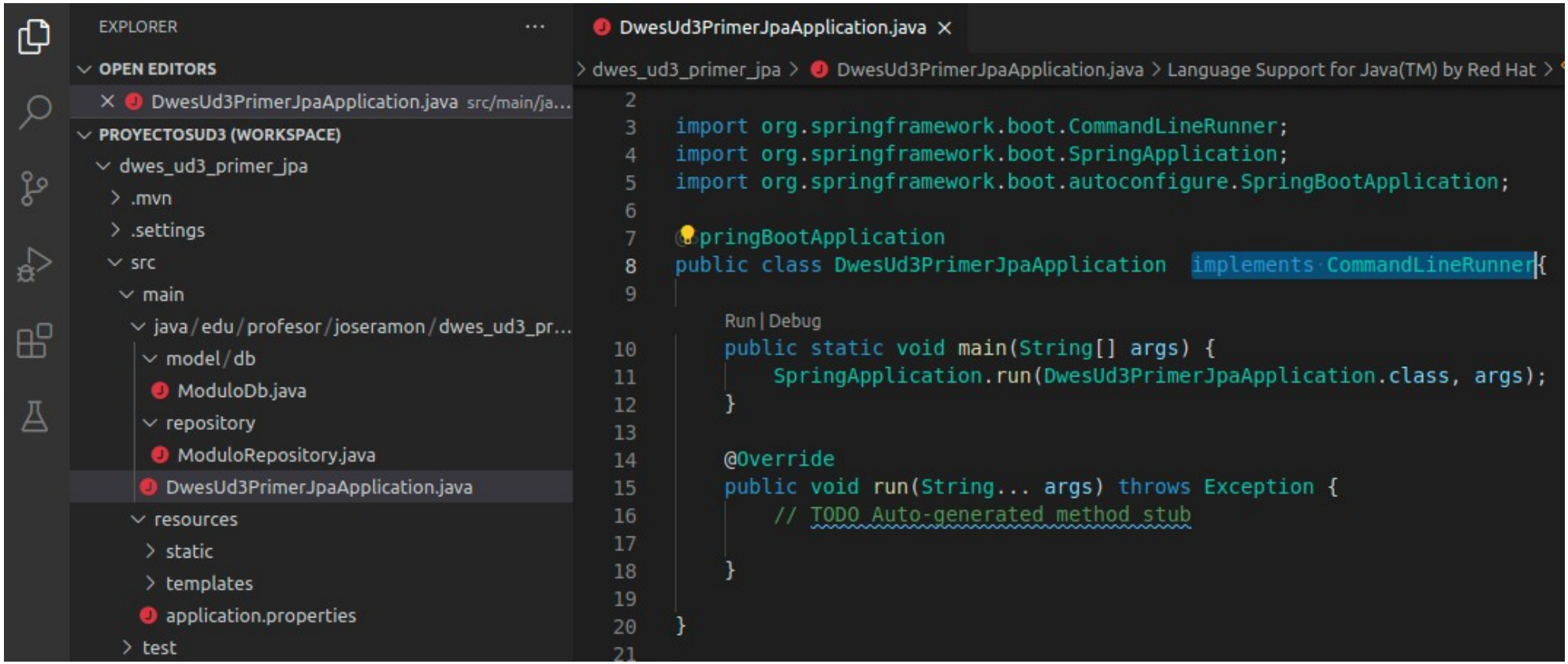
En el paso 6 donde se hacen pruebas se entenderá como utilizar los métodos que más se gastan.

```
getOne(Long id) : Modu…  JpaRepository.getOne…
save(S entity) : S
saveAndFlush(S entity) : S
count() : long
count(Example<S> example) : long
equals(Object obj) : boolean
exists(Example<S> example) : boolean
existsById(Long id) : boolean
findAll() : List<ModuloDb>
findAll(Example<S> example) : List<S>
findAll(Example<S> example, Pageable pageable…
findAll(Example<S> example, Sort sort) : List…
findAll(Pageable pageable) : Page<ModuloDb>
findAll(Sort sort) : List<ModuloDb>
findAllById(Iterable<Long> ids) : List<Modulo…
findById(Long id) : Optional<ModuloDb>
findOne(Example<S> example) : Optional<S>
getClass() : Class<?>
hashCode() : int
saveAll(Iterable<S> entities) : List<S>
toString() : String
delete(ModuloDb entity) : void
deleteAll() : void
deleteAll(Iterable<? extends ModuloDb> entiti…
deleteAllInBatch() : void
deleteById(Long id) : void
deleteInBatch(Iterable<ModuloDb> entities) : …
flush() : void
notify() : void
notifyAll() : void
wait() : void
wait(long arg0) : void
wait(long timeoutMillis, int nanos) : void
```



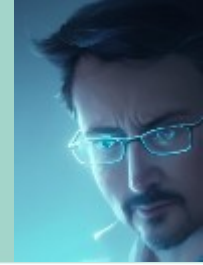
6º Configurar la App para que se pueda acceder por linea de comandos

Vamos a permitir que la aplicación pueda mostrar datos por linea de comandos (implementando CommandLineRunner y su método run) para probar los métodos que nos ha proporcionado la interfaz JpaRepository.

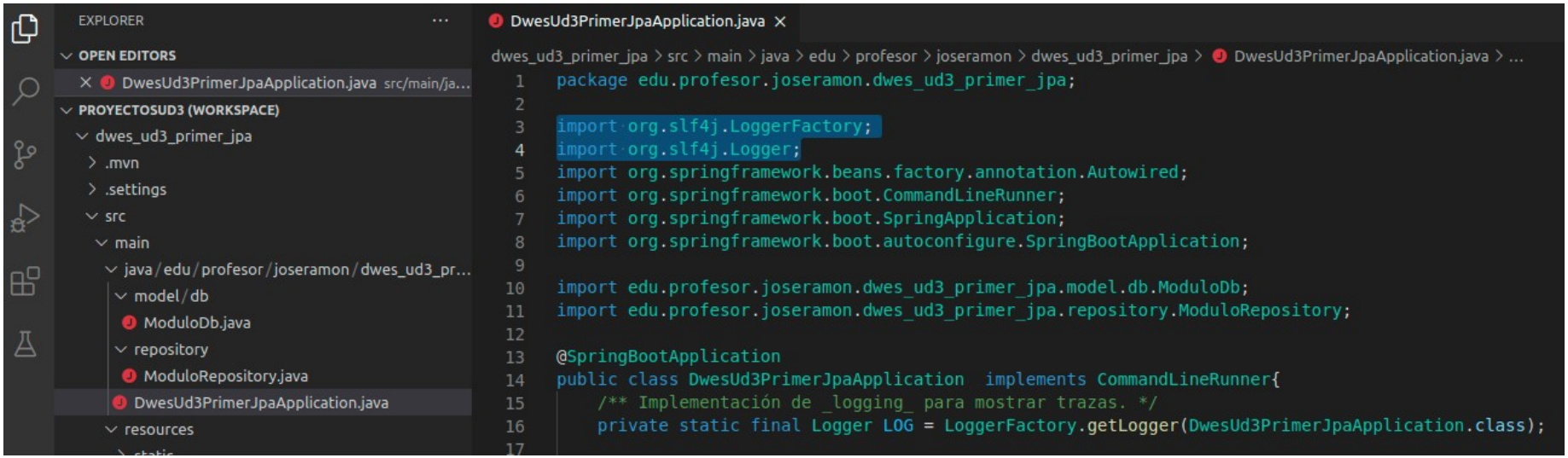


UD 3: Bases de datos y servicios REST
3.- JPA

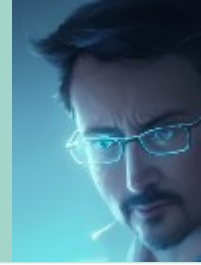
Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



Antes de continuar vamos a configurar en nuestra aplicación un Logger. En las aplicaciones reales ya no se utiliza las instrucciones System.out, estas han sido sustituidas por los **Loggers** ya que son más configurables y flexibles:



’Slf4j’ es una librería que ha sido objeto de muchos comentarios debido a un fallo de seguridad. Por suerte Spring Boot ya lo tiene controlado como podemos ver en : https://spring.io/blog/2021/12/10/log4j2-vulnerability-and-spring-boot



7º Pruebas:

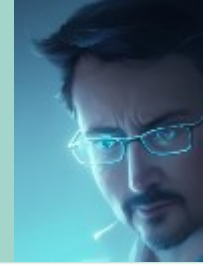
Ahora vamos a probar todas las **operaciones CRUD** (Create Read Update y Delete) sobre la **tabla "modulos"** con el siguiente código en el método "run()" que puedes copiar del DRIVE y pegar en DwesUd3PrimerJpaApplication.java.

Si lo ponemos en marcha con un punto de interrupción en la creación del primer módulo parece que toda va bien

DwesUd3PrimerJpaApplication.java

edu > profesor > joseramon > dwes_ud3_primer_jpa > DwesUd3PrimerJpaApplication.java > Language Support for Java(TM) by R

```
27
28      @Override
29      public void run(String... args) throws Exception {
30          //Crear MODULOS
31          ModuloDb nuevoModuloSinIdDb=  crearModulo_SinId();
32          LOG.info("Nuevo Modulo sin indicar el ID -> {}",nuevoModuloSinIdDb);
33          ModuloDb nuevoModuloConIdDb=  crearModulo_ConId();
34          LOG.info("Nuevo Modulo indicando el ID -> {}",nuevoModuloConIdDb);
35
36          //Leer MODULOS
37          //Como el 'id' es Long es 2L
38          Optional<ModuloDb> modulo2Db= moduloRepository.findById(2L);
39          if (modulo2Db.isPresent())
40              LOG.info("Mostrar Modulo si existe -> {}",modulo2Db.get());
41
42          //Modificar MODULOS
43          ModuloDb moduloModificadoDb=modificarModulo(nuevoModuloSinIdDb);
44          LOG.info("Modificar Modulo -> {}",moduloModificadoDb);
45
46          //Listar MODULOS
47          List<ModuloDb> listaModulos=moduloRepository.findAll();
48          LOG.info("Listar Modulos -> {}",listaModulos);
49
50          //Borrar MODULOS utilizando registro
51          borrarModuloUtilizandoRegistro(moduloModificadoDb);//Borrar módulo 4
52          //Tras borrar listaModulos contiene el módulo borrado porque no se actualiza
53          //por lo que debemos de volver a consultar los módulos directamente de la BD
54          LOG.info("Lista tras borrar módulo 4 -> {}",moduloRepository.findAll());
55      }
56
57      private ModuloDb crearModulo_SinId(){
58          ModuloDb nuevoModulo2Db= new ModuloDb(); //Crear modulo nuevo
59          nuevoModulo2Db.setNombre("Redes de Area Local");
60          nuevoModulo2Db.setAbreviatura("RAL");
61          nuevoModulo2Db.setHoras(3);
62          return moduloRepository.save(nuevoModulo2Db);
63      }
```



Sin embargo falla al intentar guardar el nuevo módulo:

DwesUd3PrimerJpaApplication.java

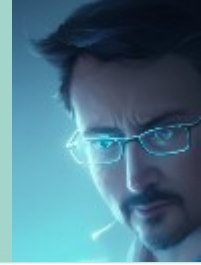
```
798                 throw new IllegalStateException("Failed to execute ApplicationRunner", ex);
799             }
800         }
801
802         private void callRunner(CommandLineRunner runner, ApplicationArguments args) {
803             try {
804                 (runner).run(args.getSourceArgs());
805             }
806             catch (Exception ex) {
807                 throw new IllegalStateException("Failed to execute CommandLineRunner", ex);
808             }
809         }
810
811         private void handleRunFailure(ConfigurableApplicationContext context, Throwable exception,
812                 SpringApplicationRunListeners listeners) {
813             try {
814                 try {
815                     handleExitCode(context, exception);
816                     if (listeners != null) {
817                         listeners.failed(context, exception);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Filter (e.g. text, !exclude)

```
    values
        (?, ?, ?, ?)
2021-02-12 13:06:33.458  WARN 30228 --- [           main] o.h.engine.jdbc.spi.SqlExceptionHelper   : SQL Error: 23505, SQLState: 23505
2021-02-12 13:06:33.466 ERROR 30228 --- [           main] o.h.engine.jdbc.spi.SqlExceptionHelper   : Violación de indice de Unicidad ó Clave primaria:
KEY ON PUBLIC.MODULOS(ID) [1, STRINGDECODE('Programaci\u00f3n'), 8, 'PRO']"
Unique index or primary key violation: "PRIMARY KEY ON PUBLIC.MODULOS(ID) [1, STRINGDECODE('Programaci\u00f3n'), 8, 'PRO']"; SQL statement:
insert into modulos (abreviatura, num_horas, nombre, id) values (?, ?, ?, ?) [23505-200]
```

¿ Por qué falla ? ¿ Porque se queja de la clave primaria ?



Cuidado con la generación automática de claves primarias. Falla porqué intenta poner como clave primaria el valor 1. Podemos ver en detalle las diversas opciones de configuración del generador de claves primarias (**@GeneratedValue**) consultando : https://www.baeldung.com/hibernate-identifiers.

ModuloDb.java

dwes_ud3_primer_jpa > src > main > java > edu > profesor > joseramon > dwes_ud3_primer_jpa > model > db > ModuloDb.java

```
16   **/
17   @NoArgsConstructor
18   @AllArgsConstructor
19   @Data
20   @Entity
21   @Table(name = "modulos")
22   public class ModuloDb {
23       @Id
24       @GeneratedValue
```

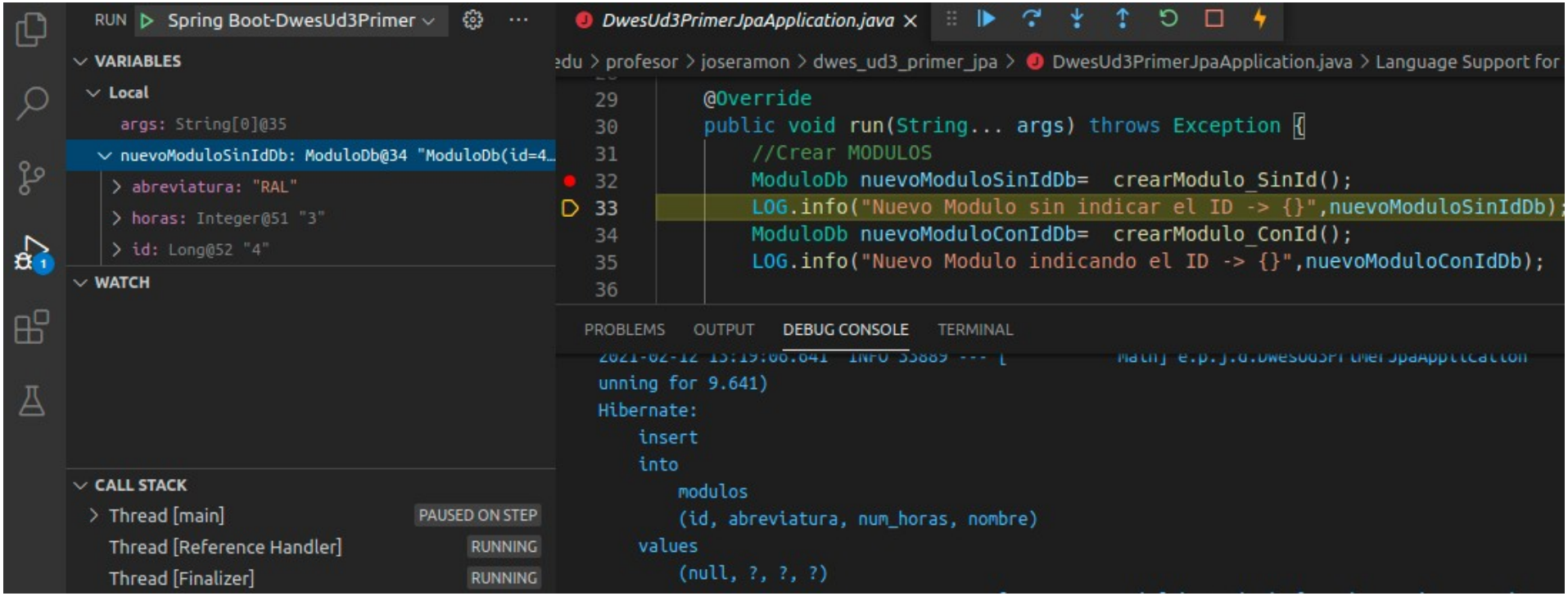
Sin embargo, vamos a **proponer directamente un método**, que sea la BD quien proporcione un número de 'Id' mediante GenerationType.IDENTITY. Si le damos un Id no fallará si no existe el registro en la BD y si no le damos valor al id Hibernate le asignará un valor al id cuando se guarde el registro :

ModuloDb.java

dwes_ud3_primer_jpa > src > main > java > edu > profesor > joseramon > dwes_ud3_primer_jpa > model > db > ModuloDb.java

```
5  > import lombok.AllArgsConstructor;…
15
16   @NoArgsConstructor
17   @AllArgsConstructor
18   @Data
19   @Entity
20   @Table(name = "modulos")
21   public class ModuloDb {
22       @Id
23       @GeneratedValue(strategy = GenerationType.IDENTITY)
```

Volvemos a ejecutar la aplicación en modo depuración y esta vez parece que funciona correctamente tanto si le damos id como si no:

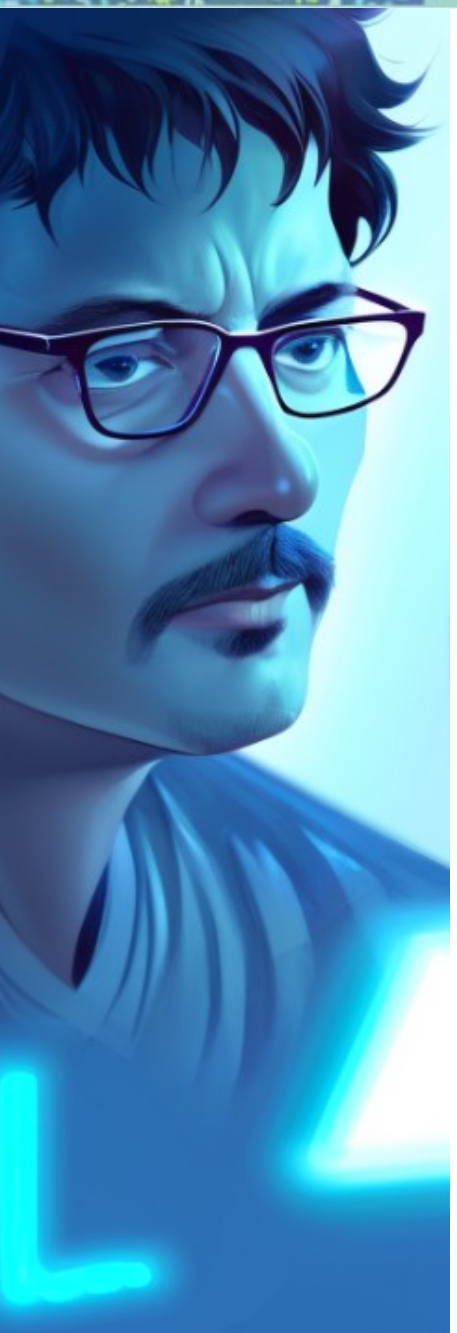


Si le das a ejecutar hasta el final paso a paso puedes comprobar que ha creado los módulos correctamente, los ha leído, los ha modificado y los ha borrado.

Ya hemos acabado nuestra primera aplicación CRUD con JPA y Spring Boot !!

UD 3: Bases de datos y servicios REST
3.- JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



EJERCICIO:

Sube la aplicación final al moodle.
Para ello:

1º Haz un "Run As \Maven Clean" para dejar solo los fichero fuentes y quitar momentaneamente los necesarios para ejecutar la aplicación (dependencias).

2º Comprime la carpeta de tu aplicación y ponle como nombre UD3_practica2_nombreAlumno.tar.gz al fichero comprimido donde nombreAlumno es el nombre del alumno que entrega la práctica.

3º Súbela al moodle.

IMPORTANTE: No comprimir en RAR, porque Ubuntu no lo lee bien y en clase tenemos Ubuntu. Si tuviesemos Windows, podemos comprimir en ZIP.