El futuro digital es de todos | MinTIC

UNIVERSIDAD EL BOSQUE | MisiónTIC 2022

Ciclo 3

Semana 3

Desarrollo de páginas Web dinámicas en Java parte 2, arquitectura de software parte 1, metodología de desarrollo Scrum parte 1 y repositorio de código GitHub

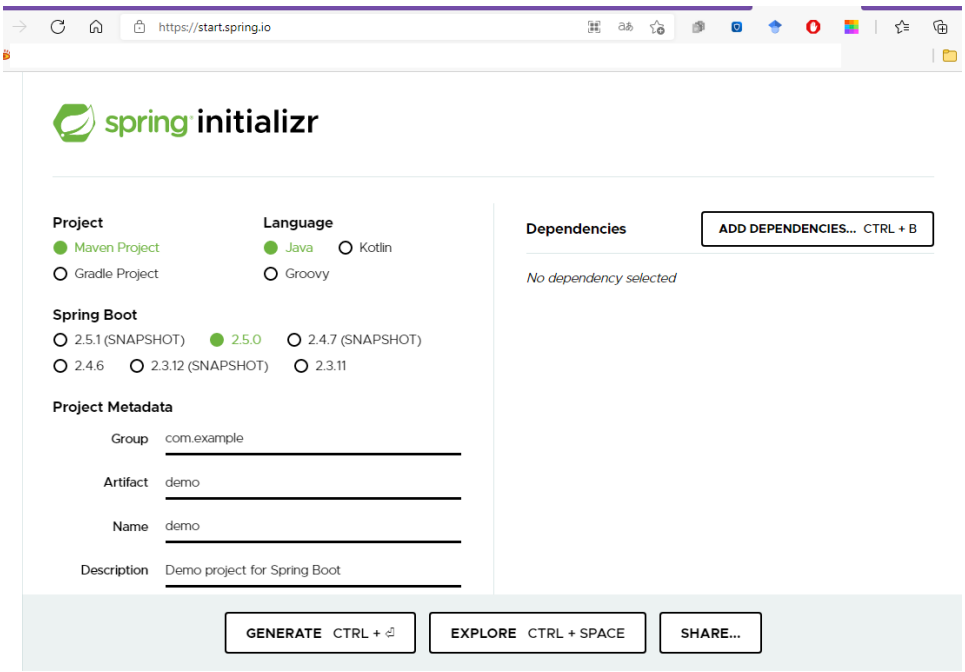
Lectura 2 - Desarrollo de proyectos Java Web con SpringBoot en Eclipse

Convenio de asociación no. 777 de 2021 suscrito entre el Fondo Único de Tecnologías de la Información y las Comunicaciones y la Universidad el Bosque.

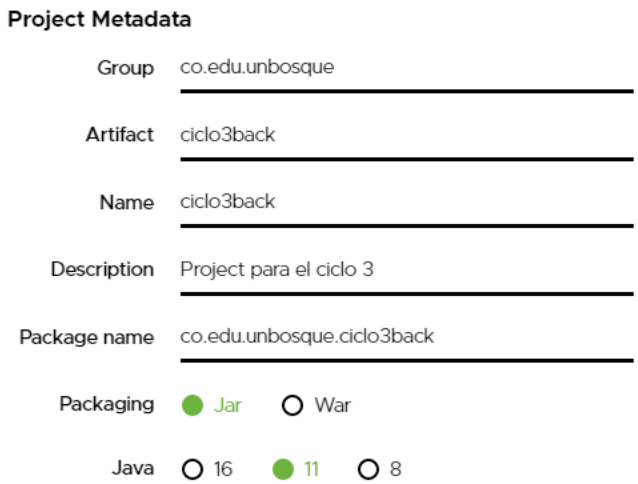
Desarrollo de proyectos Java Web con SpringBoot en Eclipse

Para desarrollar un proyecto Java Web con SpringBoot, debemos crear un archivo Project de SpringBoot compatible con Eclipse, el cual se define desde la página del creador de esta tecnología, que es: https://start.spring.io/ .

Esta página se debe ver como la siguiente:



En esta página, se ven diferentes opciones de configuración para nuestro proyecto SpringBoot, de las cuales vamos a dejar las opciones que están por defecto en **Project** (Maven Project), **Language** (Java), y la versión de **SpringBoot** (2.5.0). EN la sección Project Metadata vamos a escribir la siguiente información:



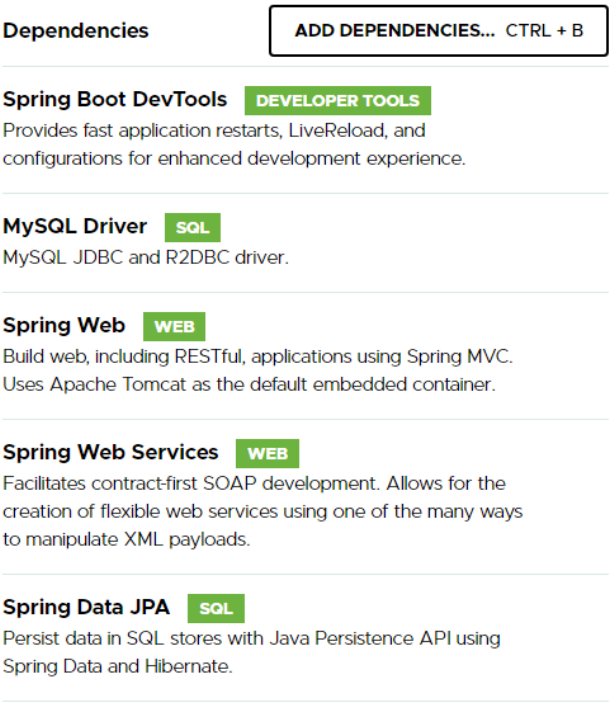
Lo indicado aquí es que vamos a utilizar una agrupación de archivos fuente Java (o **group**) que será la misma convención utilizada en la semana 2 para denominación de packages en Java como es `co.edu.unbosque`. El nombre del **artifact** (o aplicación) y el **Name** (nombre) será ciclo3back. Finalmente, se agrega una descripción para la aplicación, y el nombre particular del package, será co.edu.unbosque.ciclo3back.

Finalmente, el método de empaquetar la solución será una aplicación Jar (Java Runtime), y la versión de Java (JDK) a utilizar en nuestro caso, será el JDK 11.

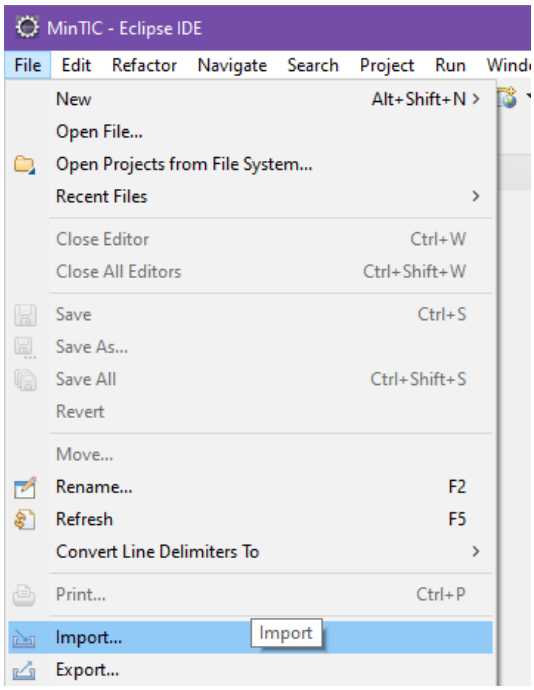
Al lado derecho de la página están las dependencias (o módulos de software que serán incorporados en la aplicación y que le darán las diferentes funcionalidades, las cuales serán manejadas por Maven). Hacemos clic en **Add Dependencies**, y seleccionaremos las siguientes:

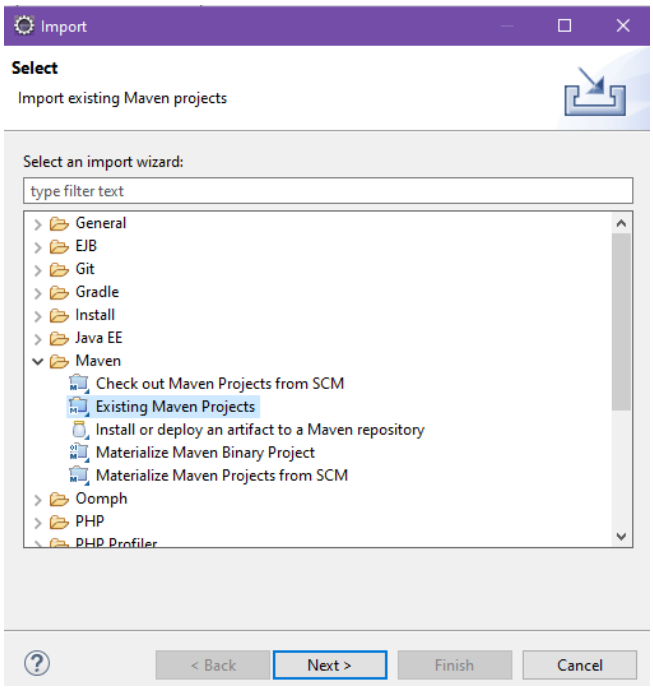
- **Spring Web** (Herramientas básicas para creación de aplicaciones Java Spring)
- **SpringBoot Dev Tools** (Herramientas para desarrollo y depuración en SpringBoot)
- **Spring Web Services** (Herramientas para creación de servicios Web de comunicación con capas superiores).
- **MySQL Driver** (controlador para conexión entre la aplicación SpringBoot, y La base de datos MySQL)
- **Spring Data JPA** (Herramientas para creación de repositorios de persistencia de datos en Java (Java Persistence API).

Finalmente, debería verse nuestra pantalla de selección de dependencias de la siguiente forma:

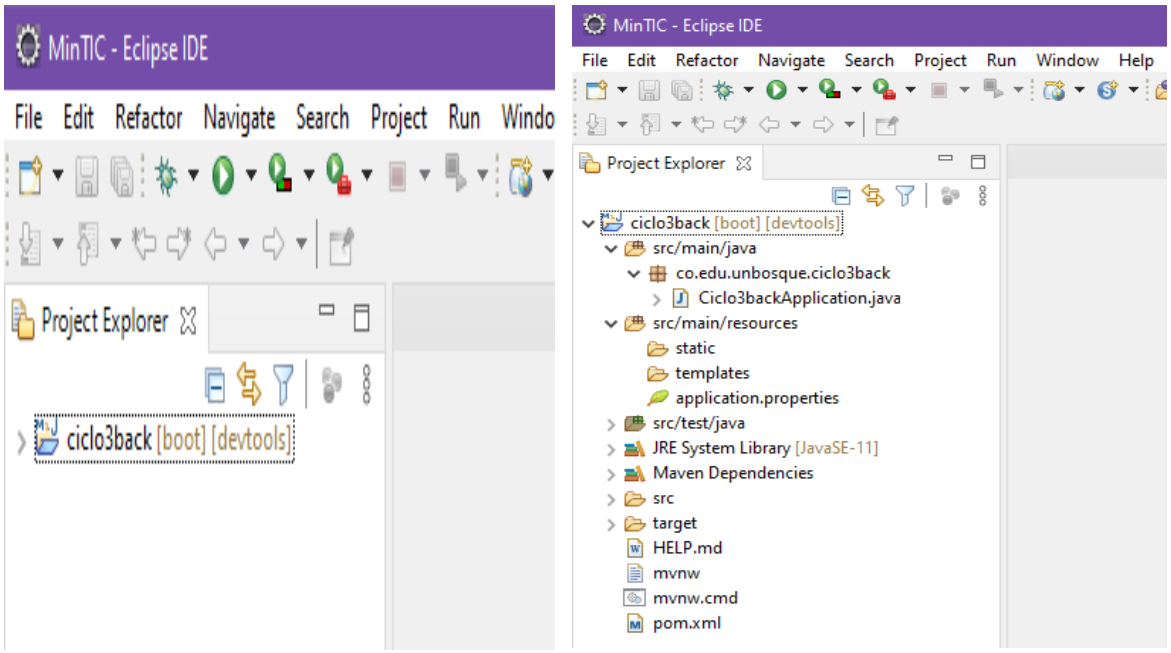


Hacemos clic en GENERATE, y nos crea un archivo empaquetado en ZIP con el nombre de la aplicación, en nuestro caso ciclo3back, y descomprimimos el Zip que nos quede una carpeta. En Eclipse, vamos a importar el proyecto que creamos, por medio de la opción File -> Import, y luego, buscamos la opción “Existing Maven Projects”, como se ve abajo:

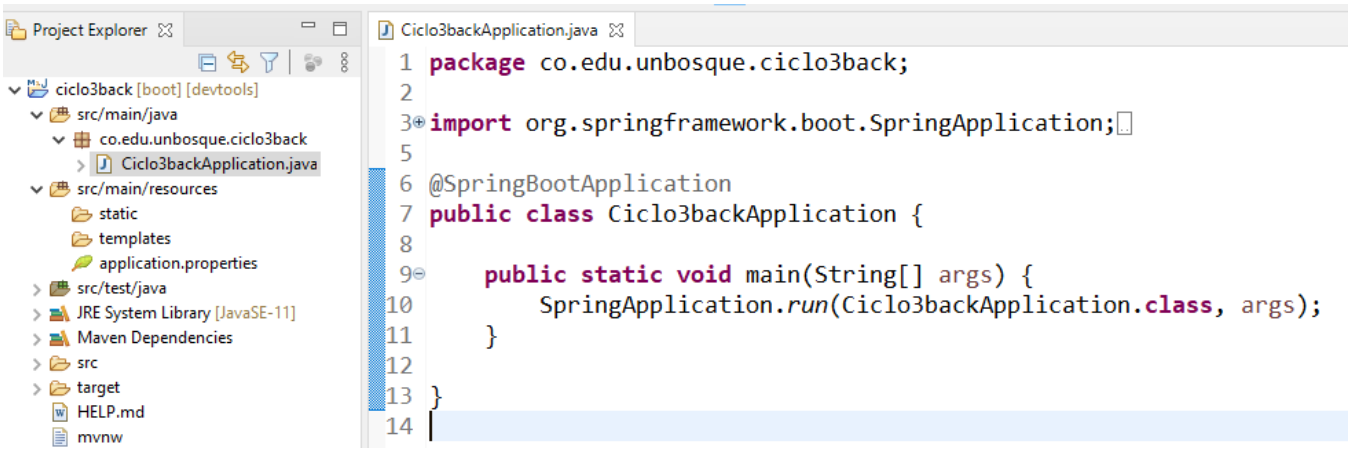




Hacemos clic en **Next**, y damos la opción **Browse**, buscamos la carpeta ciclo3back y damos la opción **Seleccionar Carpeta**. Finalmente, terminamos el proceso con **Finish**. Deberá verse el Project previamente importado en el Project Explorer de la siguiente forma, como se ve a la izquierda. A la derecha se ven los archivos que Eclipse configuró para el desarrollo de nuestra aplicación backend.



Dentro de la configuración, por lo pronto, ubicaremos dos archivos clave: 1) el archivo `Ciclo3backApplication.java`, el cual contiene el punto de ejecución de la aplicación, la cual deberá verse de esta forma.



Como se puede notar, la clase Ciclo3backApplication tiene una anotación (o declaración Java) con el nombre `@SpringBootApplication`, la cual le indica al compilador Java que va a ejecutar una aplicación SpringBoot.

El archivo clase 2) es `pom.xml`, el cual tiene toda la configuración de dependencias que administra Maven, y se ve de esta forma:

Ciclo3backApplication.java | ciclo3back/pom.xml

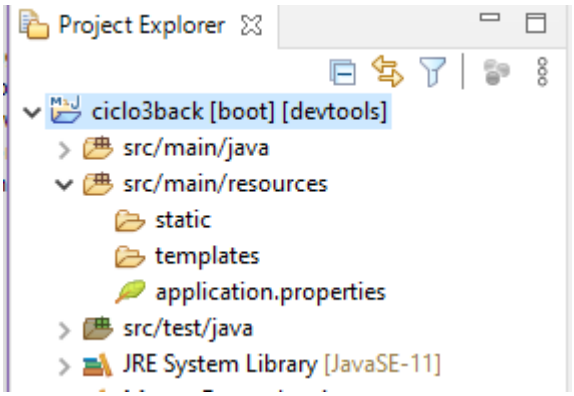
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.0</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>co.edu.unbosque</groupId>
    <artifactId>ciclo3back</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>ciclo3back</name>
    <description>Project para el ciclo 3 </description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <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-starter-web-services</artifactId>
```

```
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </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>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

En este archivo, de formato XML, se ven las diferentes dependencias marcadas con las etiquetas `<dependency></dependency>`. Posteriormente, describiremos con más detalle cada dependencia.

Dado que hicimos la instalación de la dependencia de la base de datos MySQL, Debemos a continuación realizar la realizar la configuración de la base de datos con la respectiva dependencia de tal forma que, al momento de ejecutarse la aplicación, se pueda realizar tal conexión y poder trabajarla en nuestro código.

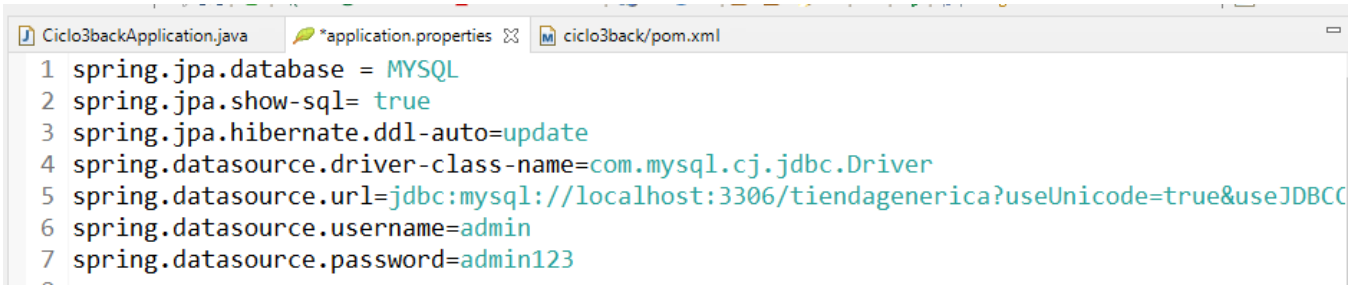
Vamos a ubicar el archivo `application.properties`, en el Project Explorer, y debe estar como se indica en la figura:



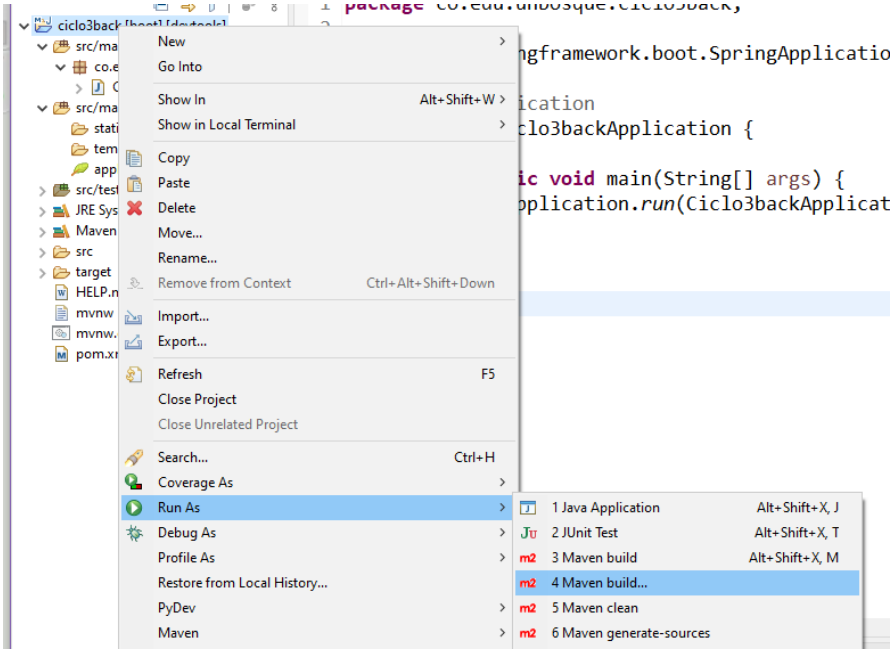
En este archivo, al hacer doble clic para editarlo, debe aparecer vacío, por lo que deberemos copiar la siguiente configuración para la base de datos MySQL para la base de datos **tiendagenerica** (que hicimos en la semana 2). Cuando se termine de copiar, deberá verse algo similar a esto:

```
spring.jpa.database = MYSQL
spring.jpa.show-sql= true
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/tiendagenerica?useUnico
de=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&se
rverTimezone=UTC
spring.datasource.username=admin
spring.datasource.password=admin123
```

Cuando se termine de copiar el archivo deberá verse así:



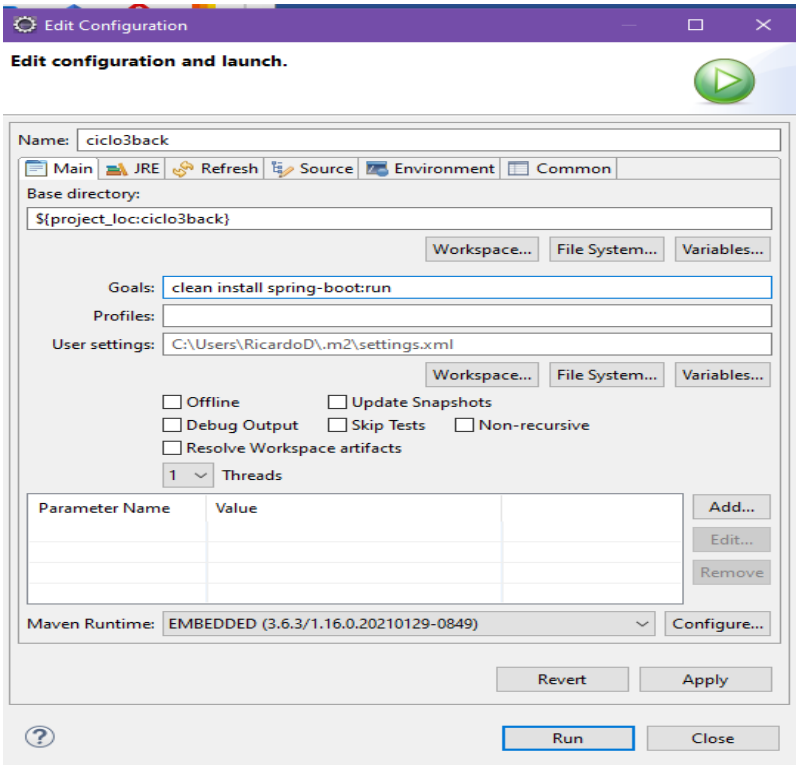
Con esto, ya estamos listos para ejecutar la aplicación por primera vez: Realizaremos la siguiente operación: hacemos clic sobre la carpeta de la aplicación ciclo3back en Project Explorer, clic sobre "**Run As..**", y luego, clic sobre "**Maven build..**" (con los dos puntos).



Nos lleva a una ventana donde debemos configurar las opciones de cómo queremos ejecutar la aplicación. Estas opciones se colocan en el campo **Goals**, y se escribe lo siguiente:

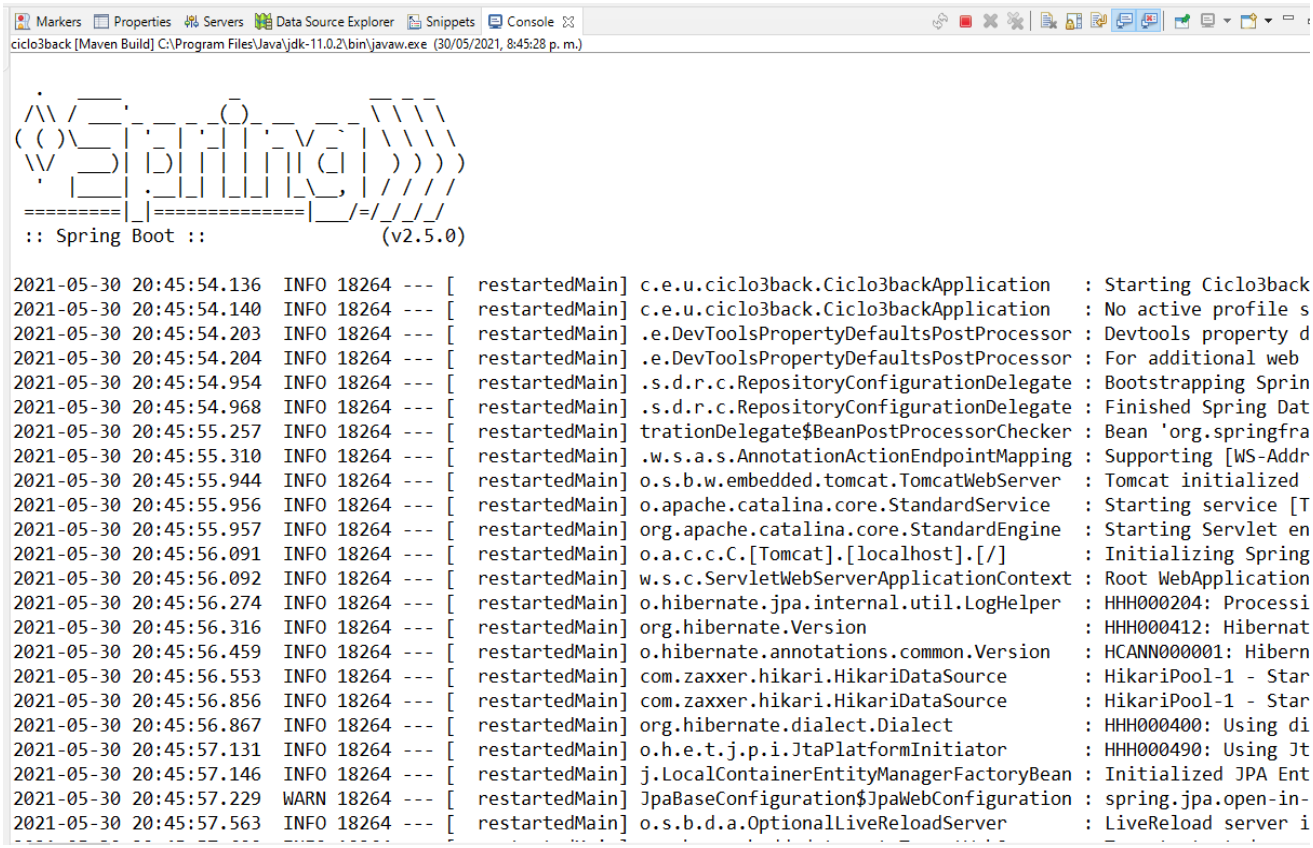
```
clean install spring-boot:run
```

Viéndose de la siguiente forma en la ventana:

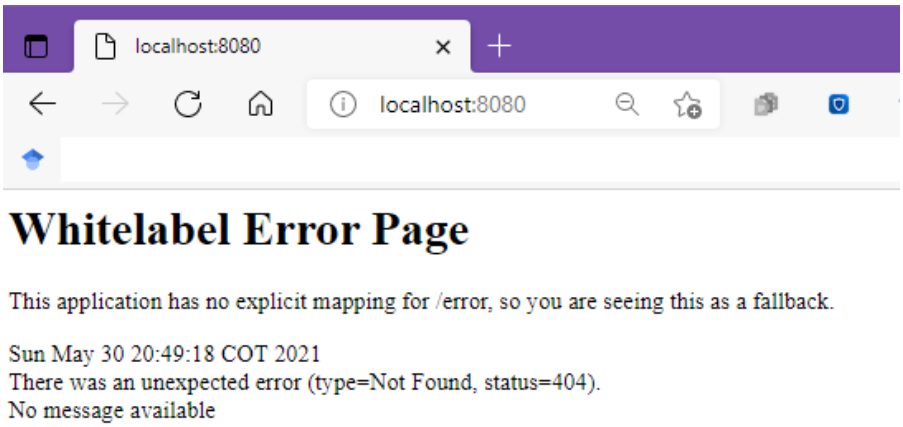


Hacemos clic en **Apply**, y luego en **Run**.

En la Ventana **Console**, deberá verse la ejecución similar a esto: (Nota: SpringBoot tiene embebido su propio servidor Web Tomcat, por lo que no es necesario correr ningún servidor).



En el navegador, podemos ejecutar http:/localhost:8080/ para constar la ejecución de la aplicación (nótese que es el mismo puerto que utilizamos en la semana 2 para la configuración del servidor Tomcat), de lo cual podemos ver la pantalla de abajo, en la que se indica que la aplicación corre correctamente, pero al momento no existe una página Web para mostrar como página de errores.



Posteriormente, procedemos a crear los siguientes sub packages dependientes del package principal, ubicado en `src/main/java` con nombre `co.edu.unbosque.ciclo3back`, con los siguientes nombres:

```
co.edu.unbosque.ciclo3back.model
co.edu.unbosque.ciclo3back.api
co.edu.unbosque.ciclo3back.dao
```

El propósito de los sub packages a construir es el siguiente:

a) **Model:** definir la estructura de datos de **usuarios** que estará en la aplicación, y que conectará con la tabla **usuarios**, previamente creada.
b) **Api (Application Program Interface):** es la definición de los métodos que harán las operaciones de inserción, borrado, listado y actualización de los datos (en adelante las llamaremos a estas operaciones CRUD: Create, Report, Update, Delete) de la tabla **usuarios**.
c) **DAO(Data Access Object):** corresponde a la creación y mantenimiento del repositorio JPA (*Java Persistence API*) de la tabla **usuarios**, que permitirá realizar las opciones CRUD, dentro de la aplicación., y que hará la interfaz con la base de datos MySQL.

Con base en lo anterior, podemos estructurar el diagrama de capas de la aplicación que llevamos al momento de la siguiente forma, basados en la gráfica inicial presentada en la semana 2, de la cual, resaltamos en rojo los componentes (packages de Java, bases de datos) que estamos trabajando:

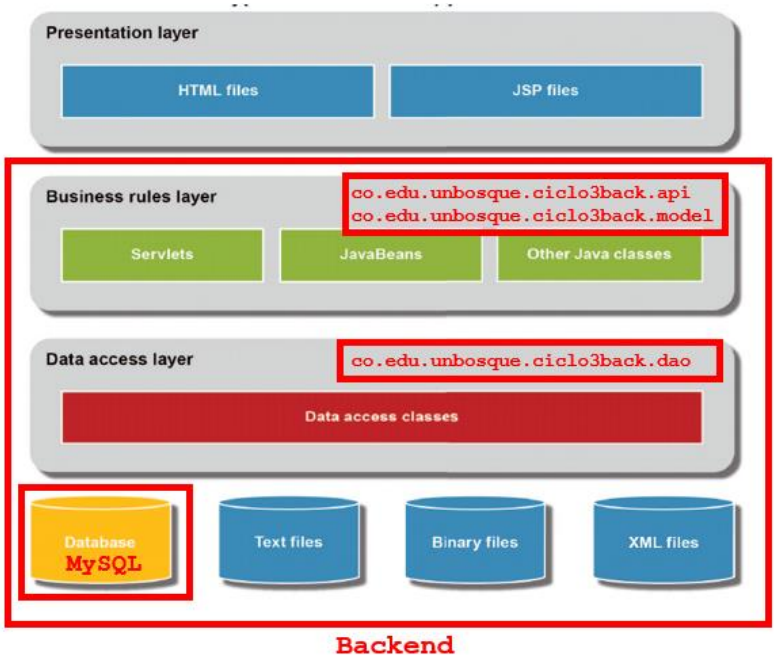


Gráfico 1: Tomado de https://ichi.pro/es/introduccion-al-desarrollo-web-de-servlet-jsp-14770007296516

Relación de Clases por crear en Java.

a) `Usuarios.java`:

En el sub package `model`, vamos a crear una nueva clase llamada Usuarios, la cual tendrá el siguiente código:

La anotación @Entity tiene como propósito informar a Java que la clase es una entidad, o estructura de datos que se conecta con una tabla equivalente en la base de datos, y, la anotación @Id indica que el atributo que le sigue es una llave primaria (PK), la cual igualmente fue definida como tal en la semana 2 en la tabla de MySQL.

```
package co.edu.unbosque.ciclo3back.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Usuarios {
	@Id
	private long cedula_usuario;
	private String nombre_usuario;
	private String email_usuario;
	private String usuario;
	private String password;
}
```

En la opción “Source -> Generate getters and setters”, aplicamos los getters y setters para los atributos.

b) UsuariosDAO.java

Vamos a crear en el sub package `co.edu.unbosque.ciclo3back.dao` , la clase UsuariosDAO.java, la cual tendrá el siguiente código:

```
package co.edu.unbosque.ciclo3back.dao;

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

import co.edu.unbosque.ciclo3back.model.Usuarios;

public interface UsuariosDAO extends JpaRepository<Usuarios, Integer>{

}
```

Esta es la clase que nos permitirá utilizar del JPA, los métodos del CRUD incrustados para la gestión de la tabla de **Usuarios** dentro del código java.

c) `UsuariosAPI.java`.

Esta clase tendrá como propósito realizar las operaciones CRUD dentro del repositorio JPA, la cual generará las APIs que estarán operando desde el backend, de tal forma que pueden ser llamadas desde la aplicación del frontend en JSP realizada la semana anterior.

```java
package co.edu.unbosque.ciclo3back.api;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import co.edu.unbosque.ciclo3back.dao.UsuariosDAO;
import co.edu.unbosque.ciclo3back.model.Usuarios;

@RestController //esta es una clase REST
@RequestMapping("usuarios")
public class UsuariosAPI {

	@Autowired //inyecta la dependencia de todos los métodos del JPA para usuarioDAO
	private UsuariosDAO usuariosDAO;

	@PostMapping("/guardar")//Request convierte en un objeto Java desde un JSon
	public void guardar(@RequestBody Usuarios usuarios) {
		usuariosDAO.save(usuarios);
	}

	@GetMapping("/listar")
	public List<Usuarios> listar(){
		return usuariosDAO.findAll();
	}

	@DeleteMapping("/eliminar/{id}")
	public void eliminar(@PathVariable("id") Integer id) {
		usuariosDAO.deleteById(id);
	}

	@PutMapping("/actualizar")
	public void actualizar(@RequestBody Usuarios usuarios) {
		usuariosDAO.save(usuarios);
	}


}
```

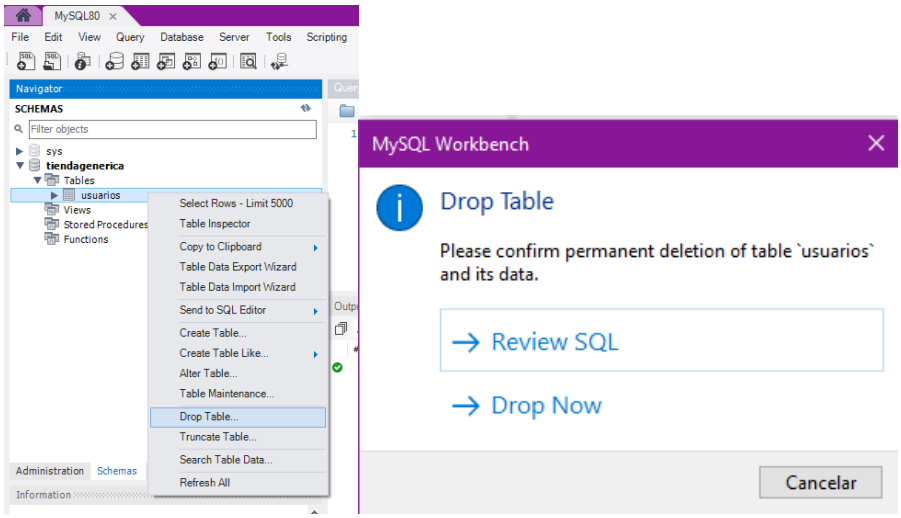
En el ejemplo que estamos trabajando, las APIs tendrían la siguiente estructura, basados en los métodos guardar(), listar(), eliminar(), y actualizar(), sus correspondientes operaciones GET, POST, PUT , y DELETE, y las URL que se invocaría para ser utilizada por el FrontEnd.

Operación CRUD	Método JAVA	URL conformada	Método Web
Agregar un nuevo registro	`public void guardar()`	http://localhost:8080/usuarios/guardar	POST
Listar todos los registros	`public List<Usuarios> listar(){`	http://localhost:8080/usuarios/listar	GET
Borrar un registro con un Id.	`public void eliminar(@PathVariable("id") Integer id)`	http://localhost:8080/usuarios/eliminar/1	DELETE
Actualizar los datos del registro identificado con un Id.	`public void actualizar(@RequestBody Usuarios usuarios)`	http://localhost:8080/usuarios/actualizar/1	PUT

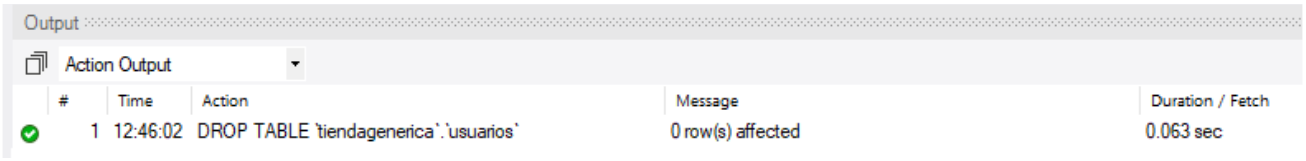
Para los dos últimos métodos – eliminar, y actualizar, se requiere agregar a la URL, el ID (o llave principal) del registro que se desea borrar.

Una de las características más útiles del marco SpringBoot, por su conexión con bases de datos es la capacidad de generar las actualizaciones a las tablas (crear, modificar, insertar, etc.), directamente desde la aplicación como tal, sin tener que realizar trabajos adicionales en MYSQL WorkBench. Como prueba de lo anterior, vamos a borrar la tabla previamente realizada en la semana 2 y dejaremos que la aplicación vuelva a crear la tabla con los cambios que indicamos en la clase Usuarios, para lo cual realizaremos lo siguiente:

En MySQL WorkBench, hacemos clic derecho en la tabla **usuarios**, y damos la opción “drop table”. Luego, saldrá una ventana de confirmación, en donde se nos indicará si queremos ver las instrucciones SQL para borrar la tabla (Review SQL), o ejecutaremos la acción (Drop Now).



Realizaremos la segunda opción, y en la ventana "Action Output" tendremos el siguiente mensaje:

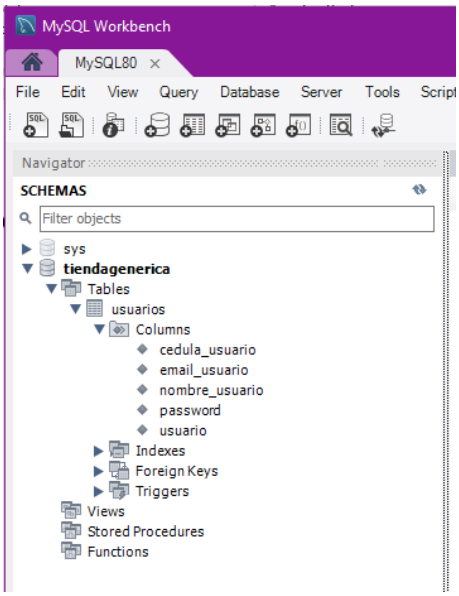


Finalmente, luego de introducir estos cambios, al ejecutar la aplicación, vamos a constatar la creación de la tabla de Usuarios desde SpringBoot – por medio de Hibernate, y en MYSQL, de la siguiente forma:

En la consola de SpringBoot, se ve se la siguiente forma:

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.5.0)

2021-06-27 12:49:47.742  INFO 1476 --- [           main] c.e.u.c.Ciclo3backApplicationTests       : Starting Ciclo3backApplicationTests using Java 15.0.2 on Lenovo-RDCL with PID 1476 (started by Ricard
2021-06-27 12:49:47.751  INFO 1476 --- [           main] c.e.u.c.Ciclo3backApplicationTests       : No active profile set, falling back to default profiles: default
2021-06-27 12:49:49.511  INFO 1476 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-06-27 12:49:49.702  INFO 1476 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 166 ms. Found 1 JPA repository interfaces.
2021-06-27 12:49:50.485  INFO 1476 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.ws.config.annotation.DelegatingWsConfiguration' of type [org.springframewor
2021-06-27 12:49:50.673  INFO 1476 --- [           main] .w.s.a.s.AnnotationActionEndpointMapping : Supporting [WS-Addressing August 2004, WS-Addressing 1.0]
2021-06-27 12:49:51.535  INFO 1476 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2021-06-27 12:49:51.637  INFO 1476 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.4.31.Final
2021-06-27 12:49:51.862  INFO 1476 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2021-06-27 12:49:52.151  INFO 1476 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2021-06-27 12:49:53.071  INFO 1476 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2021-06-27 12:49:53.100  INFO 1476 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.MySQL57Dialect
Hibernate: create table usuarios (cedula_usuario integer not null, email_usuario varchar(255), nombre_usuario varchar(255), password varchar(255), usuario varchar(255), primary key (cedula_usuario)) en
2021-06-27 12:49:54.763  INFO 1476 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.
2021-06-27 12:49:54.781  INFO 1476 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-06-27 12:49:56.228  WARN 1476 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during vi
2021-06-27 12:49:57.960  INFO 1476 --- [           main] c.e.u.c.Ciclo3backApplicationTests       : Started Ciclo3backApplicationTests in 11.371 seconds (JVM running for 13.927)
2021-06-27 12:49:57.963  INFO 1476 --- [           main] o.s.b.a.ApplicationAvailabilityBean      : Application availability state LivenessState changed to CORRECT
2021-06-27 12:49:57.971  INFO 1476 --- [           main] o.s.b.a.ApplicationAvailabilityBean      : Application availability state ReadinessState changed to ACCEPTING_TRAFFIC
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 13.186 s - in co.edu.unbosque.ciclo3back.Ciclo3backApplicationTests
2021-06-27 12:49:58.694  INFO 1476 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
2021-06-27 12:49:58.699  INFO 1476 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
```

En la línea destacada en rojo, se evidencia la creación de la tabla, con los campos que se definieron previamente en la Clase Usuarios, y en el MYSQL Workbench, se ve de la siguiente forma:



Estos pasos que acabamos de realizar serán el punto de partida para la creación del primer incremento de producto por realizar en el primer Sprint.