	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador



**IES LOS SAUCES
BENAVENTE (ZAMORA)**

**TÍTULO GRADO SUPERIOR DE
DESARROLLO DE APLICACIONES WEB**

DEPARTAMENTO: INFORMÁTICA

TÍTULO PROYECTO: SPRING BOOT FRAMEWORK

AUTOR: Ismael Heras Salvador

CURSO: 2019/2020

PERIODO: JUNIO

TIPO DE PROYECTO: Proyecto de investigación y desarrollo

CÓDIGO DE PROYECTO:

TUTORÍA COLECTIVA: Amor Rodríguez Navarro

TUTORÍA INDIVIDUAL: María Criado Domínguez



CÓDIGO DEL PROYECTO: código


TÍTULO: SPRING BOOT FRAMEWORK

AUTOR: Ismael Heras Salvador

SPRING BOOT FRAMEWORK

Presentación del famoso framework de java, spring boot, todas las herramientas necesarias para desarrollar en él y demostración práctica con un CRUD.



	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

El presente proyecto de investigación y desarrollo cuyo título es SPRING BOOT FRAMEWORK del módulo PROYECTO ha sido realizado por Ismael Heras Salvador del IES LOS SAUCES del Ciclo Formativo Grado Superior Desarrollo de aplicaciones web con el fin de la obtención del Título de Técnico Superior en Desarrollo de Aplicaciones Web.

La tutoría individual de dicho proyecto ha sido llevada a cabo por D./D^a. María Criado Domínguez profesor de segundo curso de CFGS Desarrollo de Aplicaciones web.


En Benavente, __ de mayo de 2020

Fdo. NOMBRE Y
APELLIDOS

Fdo. Amor Rodríguez
Navarro

Fdo.: María Criado
Domínguez

Fdo.: Ismael Heras
Salvador

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

1Resumen del proyecto.....	5
2Conceptualización del problema.....	5
3Objetivos.....	6
4Que es spring boot y sus características.....	6
5Tecnologías de Spring Boot.....	8
5.1Dependencias.....	8
5.2Base de datos MYSQL.....	9
5.3Método de conexión a la base de datos JPA.....	10
5.4Anotaciones y beans.....	10
5.5Contenedores y beans.....	13
5.6Patrones de diseño en el scope de SPRING BOOT.....	14
6Implementación.....	16
6.1Modelo de red, equipos y sistemas operativos.....	16
6.2IDE y herramientas de desarrollo utilizados.....	17
6.3Lenguajes de programación y lenguajes de marcas utilizados.....	17
6.4SGBD o sistema de almacenamiento utilizado.....	17
6.5Repositorio de software.....	17
6.6Otras tecnologías utilizadas de Spring Boot.....	18
6.7Análisis y diseño del supuesto práctico:.....	18
6.8Modelo de clases.....	19
6.9Modelo de casos de uso.....	20
6.10Árbol de navegación.....	21
6.11Modelo físico de datos.....	22
7Conclusiones y líneas futuras.....	24
8Anexos.....	25
8.1Instalación de spring boot.....	25



1 Resumen del proyecto

El proyecto que queremos desarrollar es un simple CRUD con **Spring Boot** para ver lo sencillo que es realizarlo y ver todas las tecnologías utilizadas para desarrollarlo y como utilizarlas en nuestro en un futuro en un proyecto de verdad.

Antes de desarrollar el proyecto haremos una investigación teórica de que es **Spring Boot** que tecnologías implementa (que son los beans dependencias anotaciones) en que nos ayuda y como trabajar con éste **framework de java**.


También estudiaremos otras tecnologías necesarias para el desarrollo de aplicaciones tales como bases de datos, conectores para la **base de datos, IDE** que vamos a utilizar y por último documentaremos la instalación y los requisitos necesarios y como hemos mencionado antes realizaremos la aplicación con **Spring Boot**.

2 Conceptualización del problema.

La temática general del problema que vamos a describir es la presentación del framework de java **SPRING BOOT** y las tecnologías utilizadas para poder llevar a cabo su desarrollo.

La principal razón que me han llevado a estudiar este **framework** es la laboral, ya que **JAVA** y sus Frameworks son lo más utilizado en el mundo empresarial.

Considero que este asunto es importante de estudio por que el framework lo veo con mucho futuro, ya que hoy por hoy los desarrolladores de software solo se dedican a eso a desarrollar la lógica y no a la tediosa **configuración del entorno**, para eso existen otros puestos de trabajo específicos, y nuestro framework de estudio nos proporciona una fácil configuración de nuestro entorno de trabajo, hasta nos provee de un **servidor embebido**.

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

3 Objetivos

El principal objetivo del proyecto es realizar una aplicación con **Spring Boot** y demostrar las bondades de este framework y su agilidad a la hora de crear aplicaciones con respecto a utilizar java sin ningún framework o la diferencia de utilizar **spring core** con **Spring Boot**.

También explicaremos que es un CRUD y como programarlo con nuestro framework y las clases utilizadas.

Y bien explicado y documentado para que todo quede bien claro y conciso y todo el mundo que lea este proyecto entienda las tecnologías utilizadas y su funcionamiento.


4 Que es spring boot y sus características

Spring Boot es una herramienta que nos facilita el desarrollo de aplicaciones en el marco de trabajo del framework de **java Spring Core**. Con Spring Boot solo te centraras en el desarrollo de la aplicación, olvidándote de la compleja configuración de Spring Core.

Por lo tanto es una herramienta para desarrollar todo tipo de aplicaciones de java (Tanto WEB como de escritorio) en un marco ya pre configurado y listo para empezar a desarrollar.

Antes hemos mencionado que **Spring Boot** es una herramienta para implementar la configuración de **Spring Core**, este es un framework de java más popular para el mundo empresarial, para crear código de alto rendimiento, liviano y reutilizable. Su finalidad es estandarizar y agilizar los marcos de desarrollo, todo esto está enfocado a que los equipos de desarrollo se centren en la lógica de negocio.

Pero la principal desventaja que tiene es que es difícil la configuración inicial, por eso vino Spring Boot, para facilitarnos las cosas y que los desarrolladores solo se centren en la lógica de negocio.


	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

Con todas estas tecnologías ya tenemos las bases para realizar nuestro proyecto, aunque hay muchas más tecnologías que más adelante mostraremos.

Características

Como antes ya hemos explicado lo que era Spring Boot en líneas generales ahora vamos a explicar las principales características de este framework de java.

- **Configuración:** Spring Boot viene con un complejo módulo de auto-configuración para poder ejecutar la aplicación sin tener que definir absolutamente nada.
- **Dependencias:** Con Spring Boot solo determinaremos que tipo de proyecto vamos a desarrollar y el solo implementa todas las dependencias y librerías para que pueda funcionar.
- **Despliegue:** Spring Boot se puede ejecutar como una aplicación estándar, o como una aplicación WEB, ya que trae incorporado un servidor tomcat.
- **Métricas:** Spring Boot por defecto cuenta con servicios para saber el estado actual de nuestra aplicación, saber si está encendida o apagada memoria utilizada y disponible y número y detalle de los beans (en el siguiente capítulo explicaremos que son) creados por la aplicación.

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

5 Tecnologías de Spring Boot

Como en todos los proyectos de cualquier lenguaje de programación necesitamos unas tecnologías adicionales para poder desarrollarlos completamente, bases de datos a utilizar gestor de dependencias método de conexión a la base de datos etc...

Por eso a continuación describiremos cada una de las herramientas para poder desarrollar en *Spring Boot*.

5.1 Dependencias

Existen dos configuraciones para esta herramienta que son maven o gradle, los gestores de dependencias se crearon para simplificar los procesos de build, antes en los proyectos existía una persona para la creación de estas build. Ahora los crea el programa automáticamente por lo que la agilidad al crear proyectos es mucho mayor que antes.


Maven es una herramienta para generar proyectos de java que se basa en ficheros XML. Su objetivo es la compilación del código y la generación del proyecto empaquetado, se encarga de obtener todas las dependencias del proyecto y de subirlo al repositorio final para que otros proyectos que dependan de él puedan usarlo.

También se puede utilizar maven para desplegar el proyecto en un servidor o para generar test de calidad de código.

Gradle es otro gestor de dependencias para java, pero también se utiliza para Groovy o Scala, Gradle se configura con ficheros DSL basados en Groovy en vez de los XML que utiliza Maven.

Básicamente Gradle se creó para poder ser utilizado en otros lenguajes diferentes a java.

Conociendo los 2 vamos a optar en nuestra aplicación por maven.

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

5.2 Base de datos MYSQL

Como todo proyecto para tener una persistencia de datos necesitamos un motor de base de datos para guardar toda nuestra información.

En nuestro caso vamos a trabajar con mysql ya que es de las más utilizadas.

Con maven agregamos las dependencias al archivo pom.xml para poder utilizar la base de datos

```

1
2 <!-- Configuración del ORM de spring boot para persistencia -->
3 <dependency>
4   <groupId>org.springframework.boot</groupId>
5   <artifactId>spring-boot-starter-data-jpa</artifactId>
6 </dependency>
7
8 <!-- Conector/librería de MYSQL para java -->
9 <dependency>
10  <groupId>mysql</groupId>
11  <artifactId>mysql-connector-java</artifactId>
12  <scope>runtime</scope>
13 </dependency>


```

Y luego añadimos en el archivo **application.properties** lo relativo a la conexión a la base de datos.

```

1
2 #Data source
3 #Indica el driver/lib para conectar java a mysql
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5
6 #Url donde esta el servicio de tu mysql y el nombre de la base de datos
7 spring.datasource.url=jdbc:mysql://localhost:3306/mydatabase
8
9 #Usuario y contraseña para tu base de datos descrita en la línea anterior
10 spring.datasource.username=root
11 spring.datasource.password=root
12
13 #[opcional]Imprime en tu consola las instrucciones hechas en tu base de datos.
14 spring.jpa.show-sql = true

```

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

5.3 Método de conexión a la base de datos JPA

Existen varios métodos para conexión a la base de datos en JAVA, pero nosotros vamos a utilizar El sistema JPA, que nos permite crear una correlación entre una base de datos relacional y un sistema orientado a objetos. Esta correlación se llama ORM (Objet Relational Mapping) esta a su vez genera anotaciones sobre entidades.

JPA es implementada por un gestor de persistencia de nuestra elección (Hibernate, TopLink, EclipseLink).

5.4 Anotaciones y beans


Hemos mencionado antes que eran anotaciones e identidades, vamos a profundizar más en lo que son estos conceptos:

Una **identidad** o **Bean** es una clase POJO que debe proporcionar un constructor por defecto (sin argumentos), todos los getters y setters de los atributos de la clase, no puede ser final y debe implementar Serializable para accesos remotos.

```

SpringCRUD2/pom.xml  *Persona.java
2
3 import java.io.Serializable;
4
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9 import javax.persistence.Table;
10
11 @Entity
12 @Table(name= "persona")
13 public class Persona implements Serializable{
14
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private int id;
18     private String nombre;
19     private String telefono;
20
21     public Persona() {
22         // TODO Auto-generated constructor stub
23     }
24

```

	<p>CÓDIGO DEL PROYECTO: código</p> <p>TÍTULO: SPRING BOOT FRAMEWORK</p> <p>AUTOR: Ismael Heras Salvador</p>
---	--

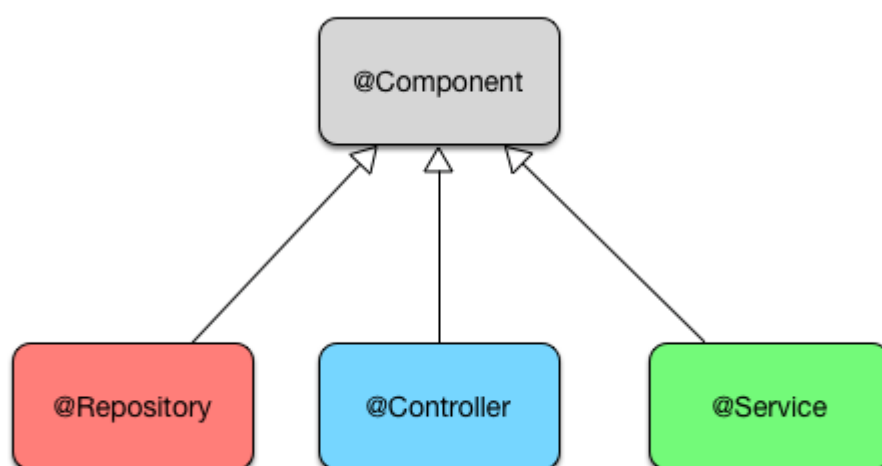
Como podemos observar en la captura anterior hemos creado una clase llamada persona, hemos puesto sobre ella dos anotaciones, con la primera **@Entity** indicamos que es una entidad y con la segunda **@table (name = "persona")** le indicamos que cree una tabla en la base de datos con el nombre persona.


Ya en las propiedades de la clase hemos añadido otras anotaciones, **@Id y @GeneratedValue (Strategy = GenerationType.IDENTITY)** Que indican que el atributo de la clase persona ID es la clave primaria en nuestra tabla persona.

Con todo esto podemos decir que las anotaciones son unas marcas para que categoricen cada uno de los componentes asociándoles una responsabilidad concreta. Y con los beans conseguimos que estas clases sean reutilizables y estén encapsuladas y tienen una mejor estructura.

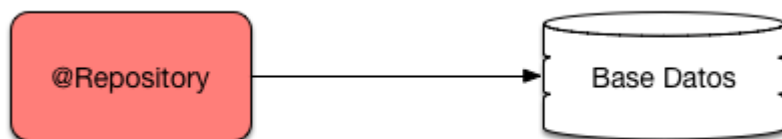
Existen cuatro tipos de anotaciones generales que las explicaremos a continuación.

- **@Component** es el estereotipo general y permite anotar un bean (más adelante explicaremos que son los bean) para que spring lo considere uno de sus objetos.

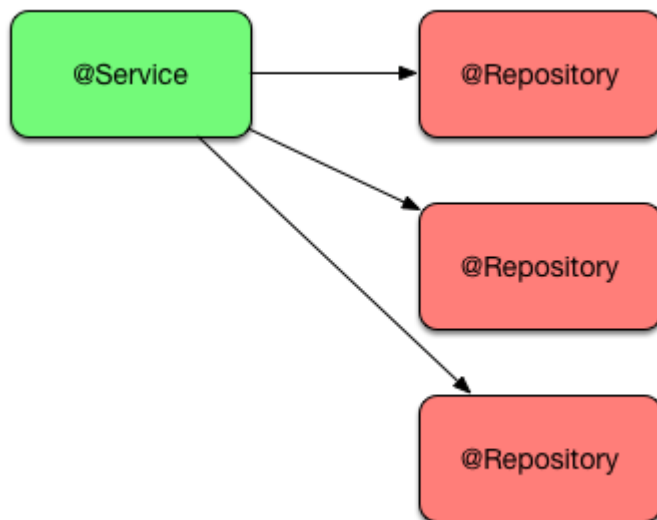



	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

- **@Repository** es el estereotipo que se encarga de dar de alta un bean para que implemente el patrón repositorio que es el encargado de almacenar datos en una base de datos.

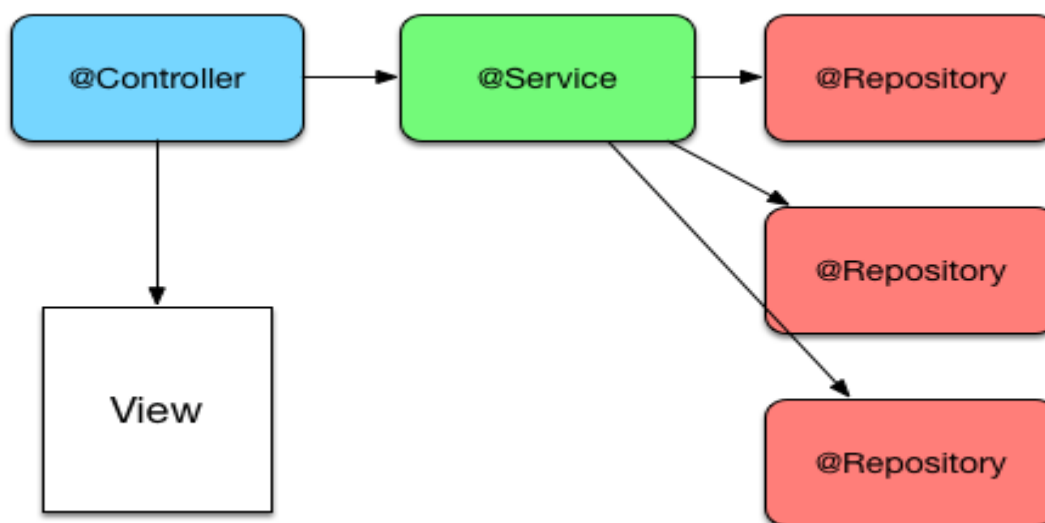


- **@Service** es el estereotipo que se encarga de gestionar las operaciones de negocio más importantes a nivel de aplicación, su tarea principal es la de agregador.



	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

- **@Controller** realiza las tareas de controlador y gestión de la comunicación entre el usuario y la aplicación, normalmente se apoya en algún motor de plantillas que facilitan la creación de páginas.



5.5 Contenedores y beans


Los beans no son unas simples clases de java, si no que sentido tendrían. Lo que le da sentido a los beans es la inyección de dependencias.

La **inyección de dependencias** es un patrón de diseño orientado a objetos, cuya finalidad es la suministrar objetos a una clase mediante el Spring

Container, sin que la propia clase tenga que crearlos.

En resumen, con Spring Boot definimos objetos (Beans) y mediante el spring Container (Contenedores) los inyectamos en la clase que los necesite.

Todo ello con las anotaciones que hemos visto antes y con muchas más que veremos después.

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

5.6 Patrones de diseño en el scope de SPRING BOOT

Como ya hemos dicho en el punto anterior, la inyección de dependencias es un patrón de diseño orientado a objetos.

Existen varios patrones principales de diseño que son el singleton y el prototype.

- **Singleton** este patrón de diseño tiene como objetivo asegurar que solo allá una instancia u objeto por clase y un punto global a ella.

Las ventajas que tiene es que tiene un control estricto a como se accede a las instancias y un mejor desempeño de la herencia, este es el patrón por defecto en el core de Spring.

- **Prototype** este patrón es lo contrario al singleton, su objetivo es la creación de varios objetos a partir de un modelo o prototipo, esto lo hace con la clonación de objetos creados previamente.

Siempre es más óptimo clonar un objeto que no crearlo desde cero, el objeto tendrá sus propios valores desde los setters.


- **Request** en este patrón el contenedor de Spring creara una nueva instancia del objeto definido por el bean cada vez que reciba un HTTP Request.



```
Bean de ámbito Request
1  @Bean
2  @Scope(value = WebApplicationContext.SCOPE_REQUEST)
3  public BeanSample beanSample() {
4      return new BeanSample ();
5  }
6
7
8  // Beans basados en @Component
9  @RestController()
10 @RequestScope
11 public class RequestScopedController extends AbstractController{
12
13     @GetMapping(AbstractController.REQUEST_SCOPE_ENDPOINT)
14     public String getUuid() {
15         return super.getUuid();
16     }
17
18 }
```

- **Session** el contenedor de Spring creara una nueva instancia del objeto definido por el bean por cada una de las sesiones HTTP y entregara esa misma instancia cada vez que reciba una nueva petición dentro de la misma sesión.

```
1  // Beans basados en anotaciones @Bean
2  @Bean
3  @Scope(value = WebApplicationContext.SCOPE_SESSION)
4  public BeanSample beanSample() {
5      return new BeanSample ();
6  }
7
8  // Beans basados en componentes
9  @RestController
10 @SessionScope
11 public class SessionScopedController extends AbstractController {
12
13     @GetMapping(AbstractController.SESSION_SCOPE_ENDPOINT)
14     public String getUuid() {
15         return super.getUuid();
16     }
17 }
```

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

- **Websockets** el contenedor de Spring creará una nueva instancia del objeto definido por el bean y la devolverá por cada petición que se produzca dentro del ciclo de vida de un websocket.

Un websocket es una tecnología avanzada que hace posible abrir una sesión comunicación interactiva entre el navegador del cliente y el servidor, con esta API puedes enviar mensajes a un servidor y recibir mensajes controlados por eventos sin tener que consultar el servidor para una respuesta.

En resumen el patrón de dependencias nos permite crear aplicaciones más fácilmente ya que nos permite unir las distintas dependencias con las clases que las necesitan.

Spring Container es el aliado perfecto para usar estos patrones de diseño ya que nos permite configurar las dependencias a inyectar mediante anotaciones o código XML de manera que podemos librar a nuestras clases de tener código innecesario.


En nuestra aplicación utilizaremos el patrón singleton (el que utiliza por defecto Spring).

6 Implementación

6.1 Modelo de red, equipos y sistemas operativos

El modelo de red que he utilizado es **RED DE AREA PERSONAL O PAM** ya que todo lo he hecho en mi casa particular mente.

He utilizado un ordenador portátil con un sistema operativo Windows 10.

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

6.2 IDE y herramientas de desarrollo utilizados

Instalaremos el Spring Tools Suite, que es un IDE basado en eclipse por lo cual es portátil y está muy optimizado para Spring Boot, pero se podría utilizar cualquier IDE del mundo de java (NetBeans, IntelliJ, Eclipse).

No necesitaremos instalar gestor de dependencias ya que Spring Boot ya tiene instalado el que quieras utilizar, **maven o gradle**.

6.3 Lenguajes de programación y lenguajes de marcas utilizados.

Los lenguajes de utilizados son **JAVA JAVASCRIPT** como lenguajes principales de programación. JAVA será el principal lenguaje de programación de nuestra aplicación, JAVASCRIPT lo utilizaremos para mostrar modales y para la librería de la paginación.


Y como lenguajes de marcas utilizaremos **HTML CSS Y XML**, HTML lo utilizaremos para la parte visual de la aplicación y el XML lo utilizaremos para el manejo de dependencias de JAVA.

6.4 SGBD o sistema de almacenamiento utilizado

Para la gestión de la base de datos vamos a utilizar el phpmyadmin que es un SGBD que corre en nuestro navegador y es muy popular en la gestión de las bases de datos.

6.5 Repositorio de software

El repositorio de código que vamos a utilizar es el **GitHub**, es un repositorio de código que está en la web y está basado en **GIT**.

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

Los repositorios de código se utilizan para almacenar nuestro código en un sitio seguro para poder hacer nuestros commit y volver a un estado anterior que nos interese y mezclar ramas de desarrolla con la master.

6.6 Otras tecnologías utilizadas de Spring Boot

THYMELEAF

Thymeleaf es un motor de plantillas que nos permite trabajar con archivos HTML en lugar de los JSP, se basa de en incluir nuevos atributos (no etiquetas) a las etiquetas de HTML, esto se conoce como ***natural templating***.

Esto permite que nuestra plantilla pueda renderizarse en local y que esa misma plantilla también sea procesada dentro del motor de plantillas, con lo cual las tareas de programación y diseño se pueden llevar conjuntamente.

Atributos básicos

Th: text permite remplazar el texto de la etiqueta por el valor de la etiqueta que le demos.

```
<p th: text="{saludo}">saludo</p>
```

Th: each nos permite repetir tantas veces como se indique o iterar sobre los elementos de un array.

```
<li Th: each="libro: {libros}" th: text="{libro.titulo}">El Quijote</li>
```

Th: class permite modificar las clases de un elemento de forma que podríamos cambiar el **CSS** de ese elemento.

```
<p th: class="{row.even}"? 'even': 'odd'"></p>
```

6.7 Análisis y diseño del supuesto práctico:

El análisis de nuestro **CRUD** es muy sencillo, simplemente tenemos una base de datos donde almacenamos usuarios con su nombre, apellido, email, nombre de usuario y password.

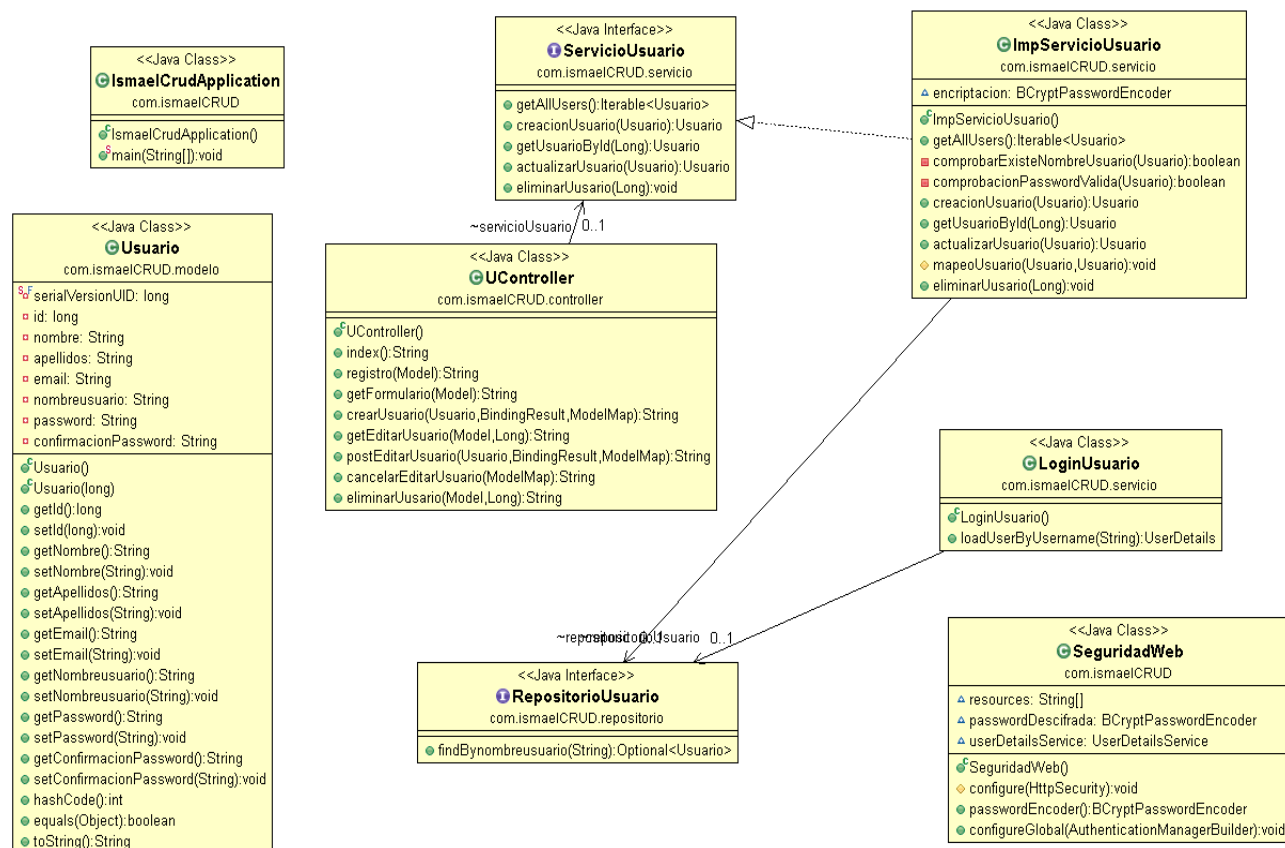
Los usuarios pueden registrarse con los datos mencionados anteriormente y **para realizar el login tendrá que mostrar las credenciales de nombre de usuario y password** y con ellas podrá acceder a la aplicación.


Una vez logeados con éxito el usuario podrá modificar sus datos (menos su contraseña) crear más usuarios, borrar usuarios o Mostar y filtrar usuarios en la tabla por nombre apellidos etc...

Esto es todo lo tiene nuestro CRUD, como veis es una cosa sencillita para presentar el framework SPRING BOOT.

6.8 Modelo de clases

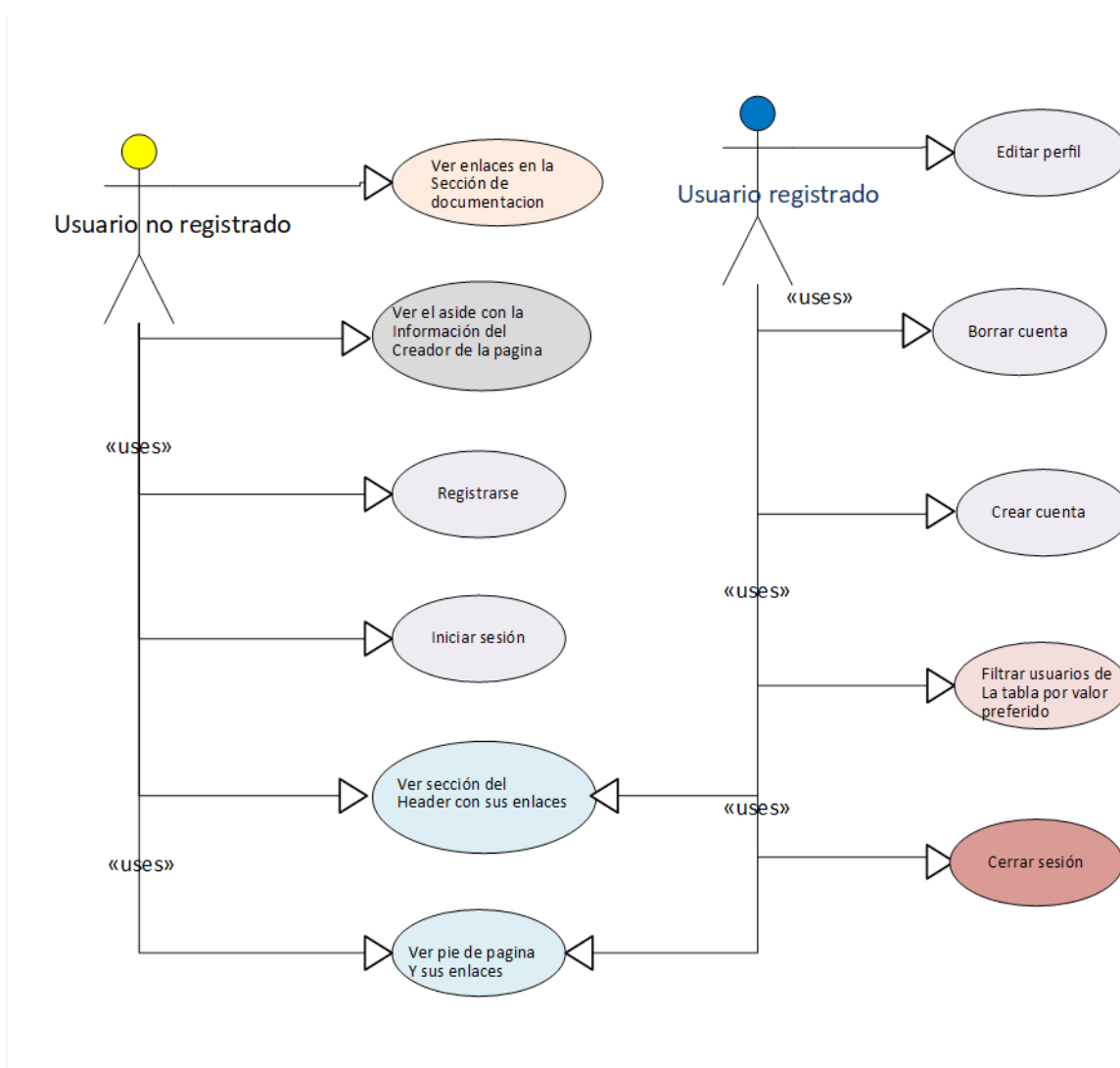
Diagrama de clases del login de nuestro CRUD realizado con spring boot. En él nos muestra todas las **clases e interfaces** con sus **métodos y atributos y la relación entre ellas**.




	<p>CÓDIGO DEL PROYECTO: código</p> <p>TÍTULO: SPRING BOOT FRAMEWORK</p> <p>AUTOR: Ismael Heras Salvador</p>
---	--

6.9 Modelo de casos de uso

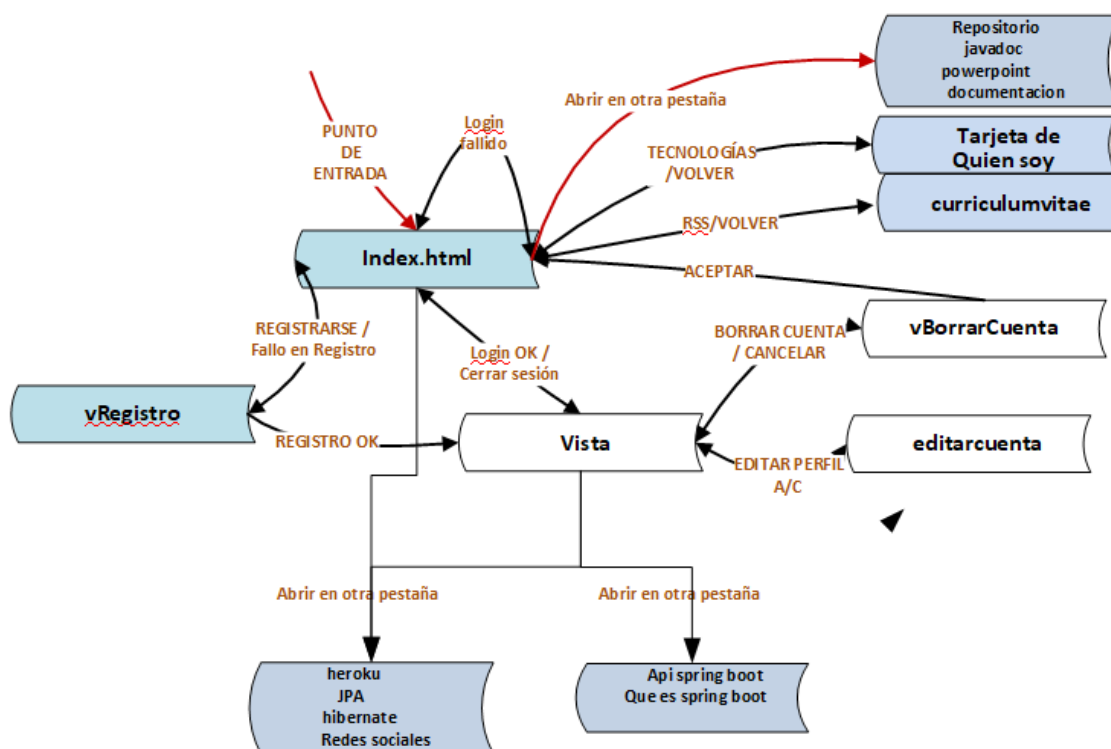
En este modelo veremos lo que puede hacer un usuario registrado y otro no registrado




	<p>CÓDIGO DEL PROYECTO: código</p> <p>TÍTULO: SPRING BOOT FRAMEWORK</p> <p>AUTOR: Ismael Heras Salvador</p>
---	--

6.10 Árbol de navegación









El árbol de navegación es muy similar al modelo de casos de uso pero es una visión un poco más gráfica de la aplicación



	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

6.11 Modelo físico de datos

El modelo físico de datos en nuestro caso solo cuenta con una tabla usuario, que es donde almacenamos los usuarios y con los datos de ella podemos hacer el login del usuario.

  ismaelcrud usuario
 id : bigint(20)
 apellidos : varchar(50)
 email : varchar(255)
 nombre : varchar(20)
 nombreusuario : varchar(15)
 password : varchar(255)



7 Conclusiones y líneas futuras

Una vez ya explicado el proyecto y todas las tecnologías que vamos a utilizar para realizarlo toca el turno para hacer las comparaciones de este proyecto con lo que hemos aprendido en el curso.

La primera y principal diferencia es el lenguaje que vamos a utilizar en nuestra aplicación en el lado del servidor. En el módulo de **DAW** (desarrollo de aplicaciones WEB) el lenguaje de back-end que hemos utilizado es el **PHP**, y nosotros en la aplicación vamos a utilizar **JAVA**.

La principal diferencia de JAVA con PHP es que JAVA es un lenguaje compilado (se compila en la JVM que es la máquina virtual de java) mientras PHP es un lenguaje de scripting, es decir no necesita ser compilado.


Otra de las grandes diferencias es que JAVA tiene un tipado fuerte mientras PHP tiene un tipado débil esto quiere decir que PHP es más flexible y confía en la programación con sentido común, pero puede llevar a complicaciones en la programación orientada a objetos, mientras que java es un lenguaje de programación totalmente orientado a objetos y es más óptimo para este paradigma de programación.

En resumen JAVA se utiliza en **equipos de desarrollo muy grandes** orientado a la creación de aplicaciones empresariales y PHP en entornos mas pequeños y para **soluciones más pequeñas** tipo **CMS o blogs**.

Con respecto a los lenguajes de marcas **HTML** y **CSS** varían poco lo único que en nuestros archivos HTML incrustaremos código de la herramienta que nos da Spring Boot, thymeleaf como ya explicamos en un punto anterior.

En el tema bases de datos La base de datos es la misma **MYSQL**, la única diferencia es la conexión a la base de datos, que en primero utilizamos JDBC y en este proyecto utilizaremos JPA.

La principal diferencia entre **JPA** y **JDBC**, es que JDBC ejecuta directamente consultas nativas de SQL sobre la base de datos, mientras que en JPA permite interactuar con la base de datos por medio de objetos, de esta forma JPA es el

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

Encargado de convertir los objetos de JAVA en instrucciones para el manejador de la base de datos (MDB).

Con respecto al uso de **JavaScript** en el módulo de DAW básicamente trabajábamos con **JavaScript** nativo (por otra parte lo mejor para aprender),

Solo utilizamos **Ajax** como librería y en este proyecto utilizaremos varias librerías para agilizar el proceso de programación de nuestros scripts.

8 Anexos


8.1 Instalación de spring boot

Ahora mostraremos la documentación de la instalación de todo el componente Para poder realizar nuestro proyecto con **Spring Boot**.

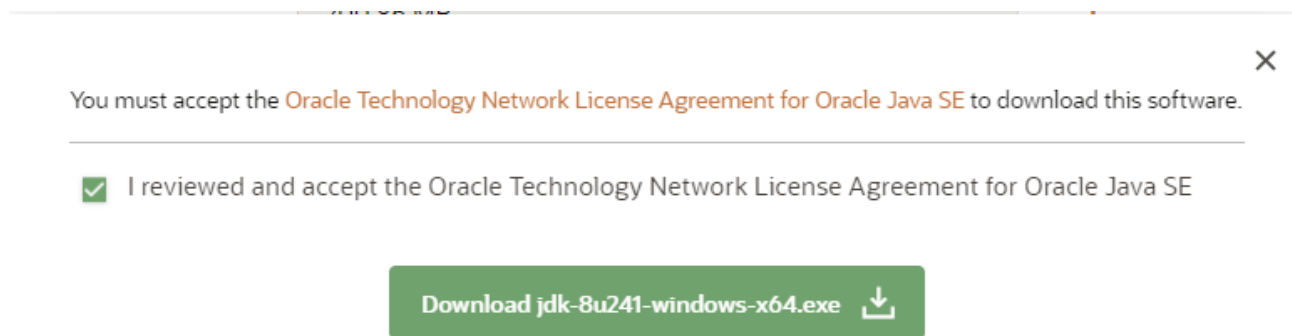
Lo primero es instalar el **JDK** que es una herramienta para convertir nuestro código fuente (.java) en byteCode (.class) el cual será interpretado por la máquina virtual de java **JVM**.

Nos dirigimos a la página de Oracle oficial para descargar el JDK

<https://www.oracle.com/java/technologies/javase-jdk8-downloads.html>

	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

instalamos la versión 8 de **JDK**




Como nota cabe destacar que ahora para descargar **JDK** hay que registrarse en Oracle para poder descargarla.

Una vez instalado el **JDK** lo que tendremos que instalar es el **IDE** para el desarrollo de la aplicación que en este caso sería el **eclipse**.

Nos dirigimos a la pagina oficial de eclipse <https://www.eclipse.org/downloads/>

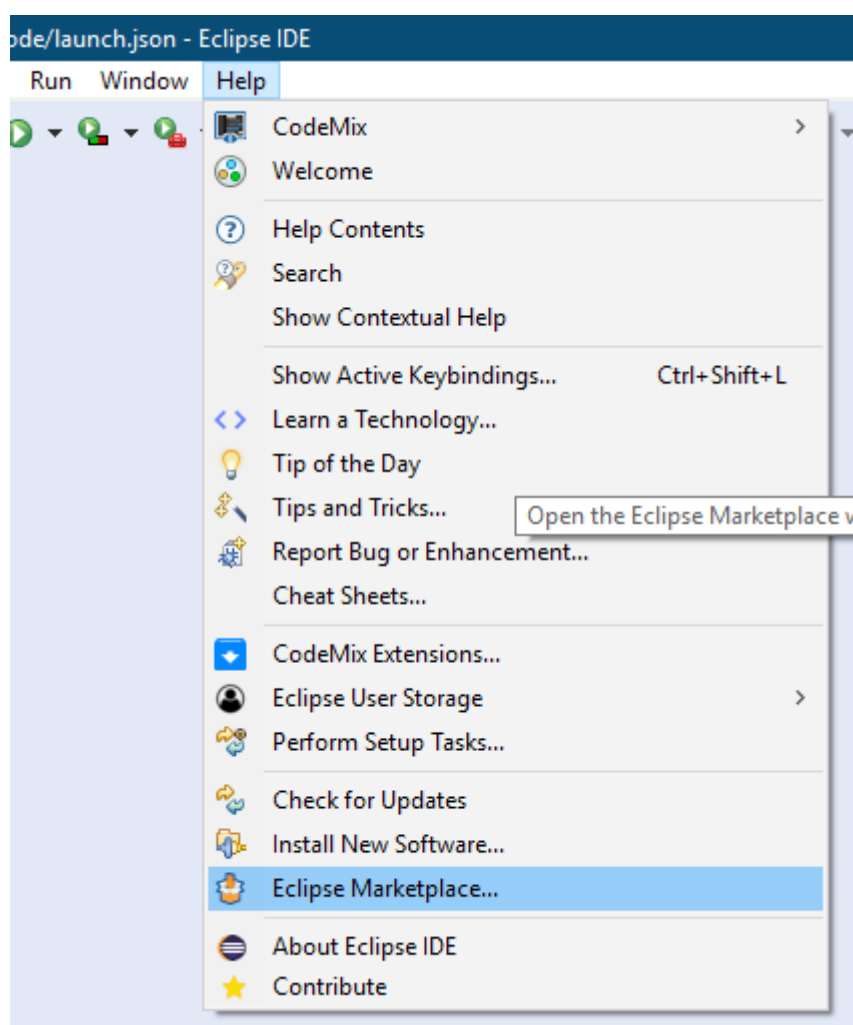
Y nos descargamos la versión de 64 bits para Windows




	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

Una vez instalado el **eclipse** tenemos que instalar el spring suite tools para que nos genere todo el proyecto de Spring Boot.

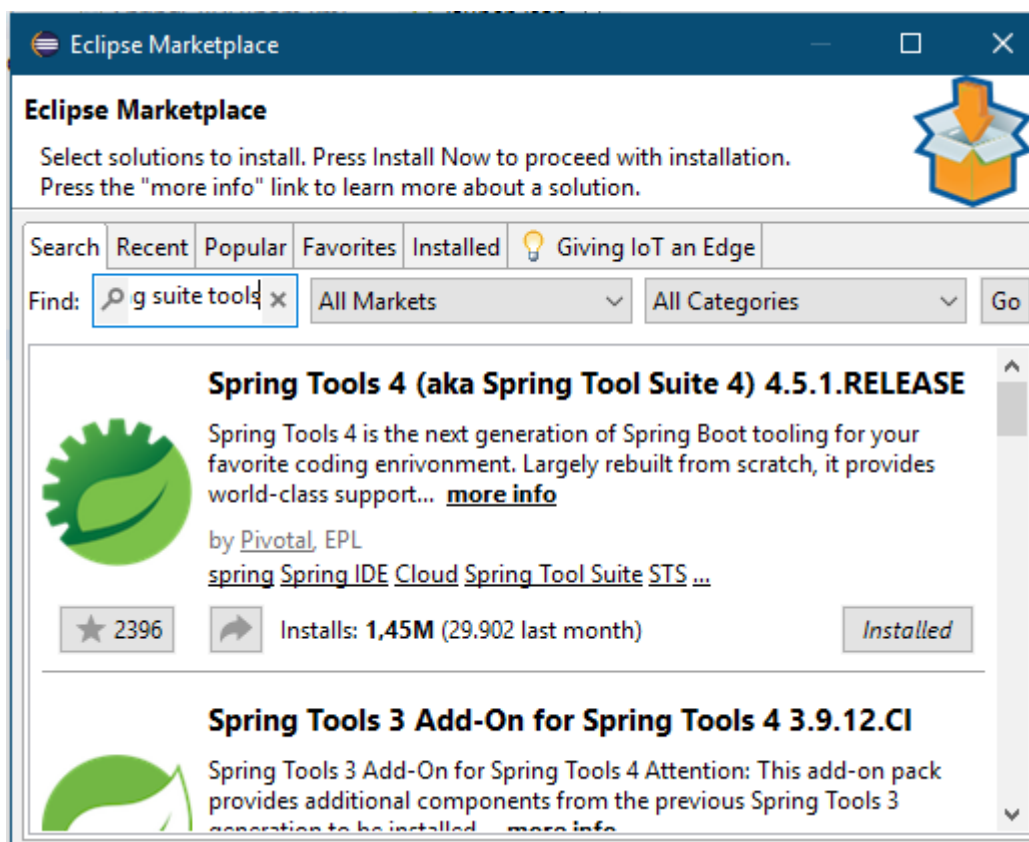
En la ventana de help de eclipse abrimos el eclipse **Marketplace**




	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

Una vez en el **Marketplace** buscamos el **Spring Suite Tools**.

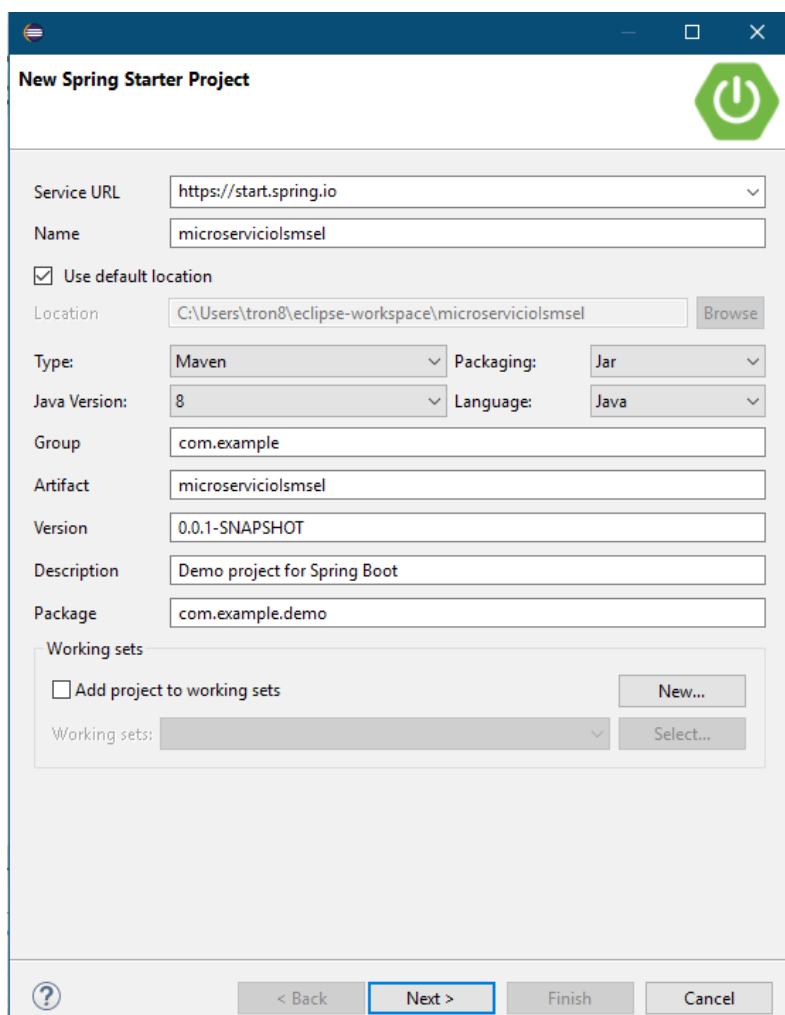
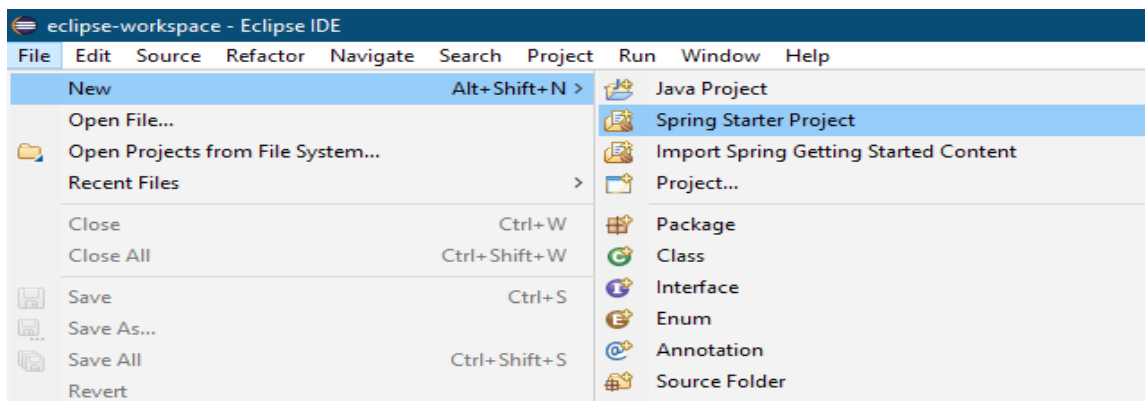
Y lo instalamos.




Ahora ya tenemos instalado todo lo necesario para poder generar nuestro proyecto de Spring Boot y poder realizar la aplicación.

	<p>CÓDIGO DEL PROYECTO: código</p> <p>TÍTULO: SPRING BOOT FRAMEWORK</p> <p>AUTOR: Ismael Heras Salvador</p>
---	--

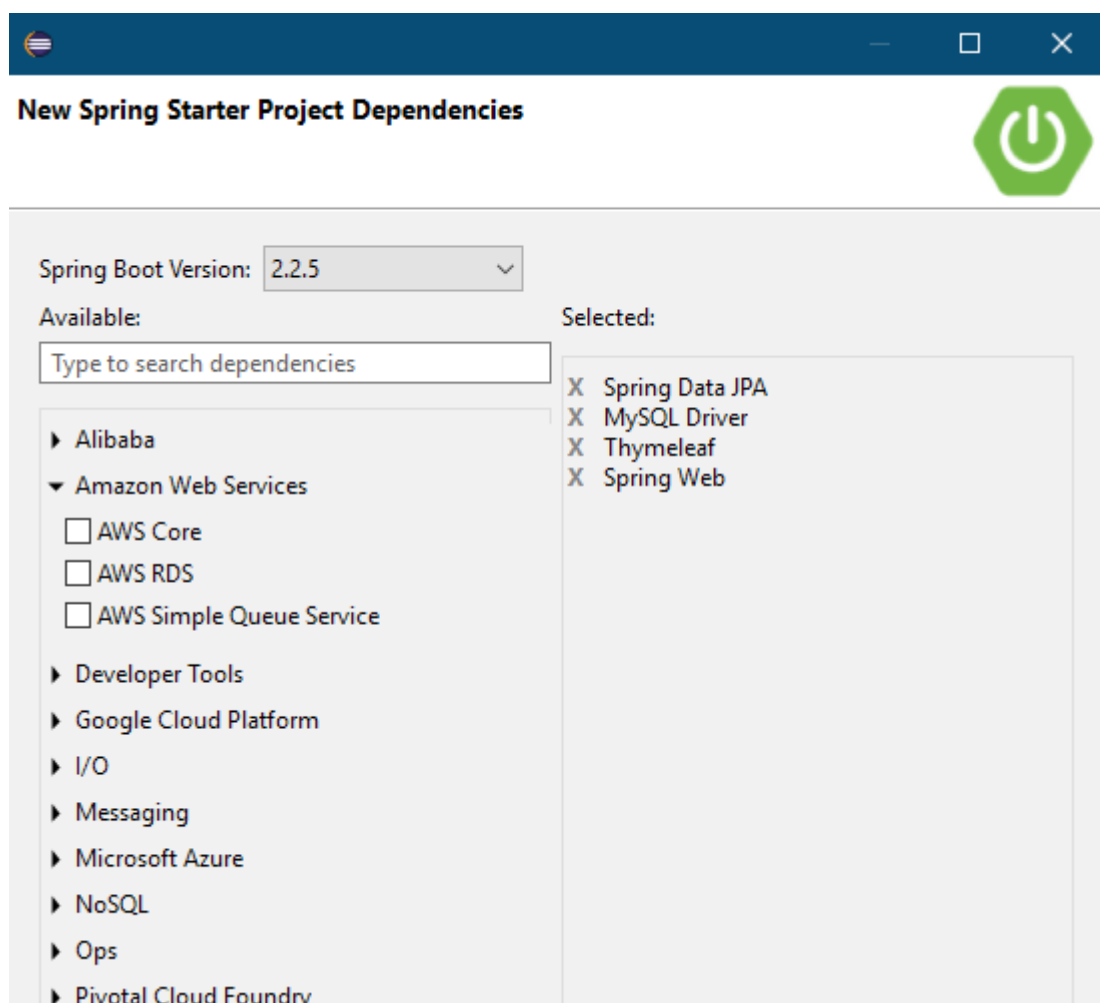
Lo último que queda es **generar el proyecto con elipse.**
 en new vamos a spring starter project para generl un nuevo proyecto.




	<p>CÓDIGO DEL PROYECTO: código</p> <p>TÍTULO: SPRING BOOT FRAMEWORK</p> <p>AUTOR: Ismael Heras Salvador</p>
---	--

Como vemos en la captura anterior al generar el nuevo proyecto de **Spring Boot** nos sale la pantalla donde elegimos el **packaging** (aunque sea un proyecto web elegiremos jar) el nombre y el nombre de los paquetes.

Y en la siguiente captura es donde elegimos todas las dependencias que necesitamos en nuestro proyecto, como vemos vamos a utilizar **Spring WEB, MYSQL Driver Thymeleaf y Spring Data JPA**. Todos ellos necesarios para la creación de nuestra aplicación.



	CÓDIGO DEL PROYECTO: código
	TÍTULO: SPRING BOOT FRAMEWORK
	AUTOR: Ismael Heras Salvador

Y con todo esto ya tendríamos preparado nuestro entorno para poder realizar nuestra aplicación (Spring Boot viene con un servidor tomcat embebido), cosa que destriparemos en próximos punto.

