



GOBIERNO DEL
ESTADO DE MÉXICO

TES OEM
TECNOLÓGICO DE ESTUDIOS SUPERIORES
ORIENTE DEL ESTADO DE MÉXICO



TESOEM

ING. EN SISTEMAS COMPUTACIONALES

NOMBRE: MANZO OLGUIN ALEX YAZMIN

GRUPO: 8S12

SEMESTRE: OCTAVO

FRAMEWORKS

PROYECTO

PROFESOR: LOPEZ CASTAÑEDA CESAR ALEJANDRO

TES OEM



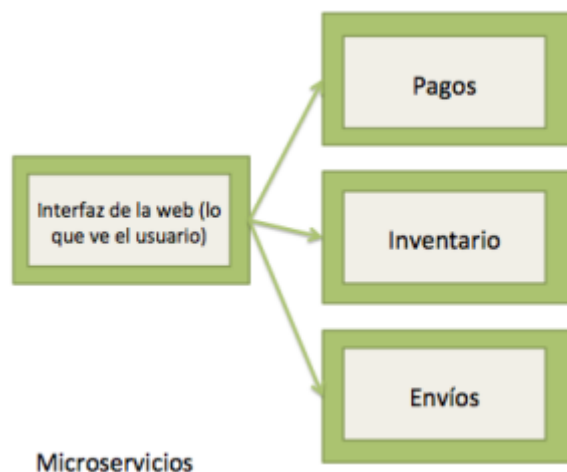
**TECNOLÓGICO DE ESTUDIOS SUPERIORES
ORIENTE DEL ESTADO DE MÉXICO**



“arquitectura de microservicios” es un enfoque para desarrollar una aplicación software como una serie de pequeños servicios, cada uno ejecutándose de forma autónoma y comunicándose entre sí, por ejemplo, a través de peticiones HTTP a sus API.

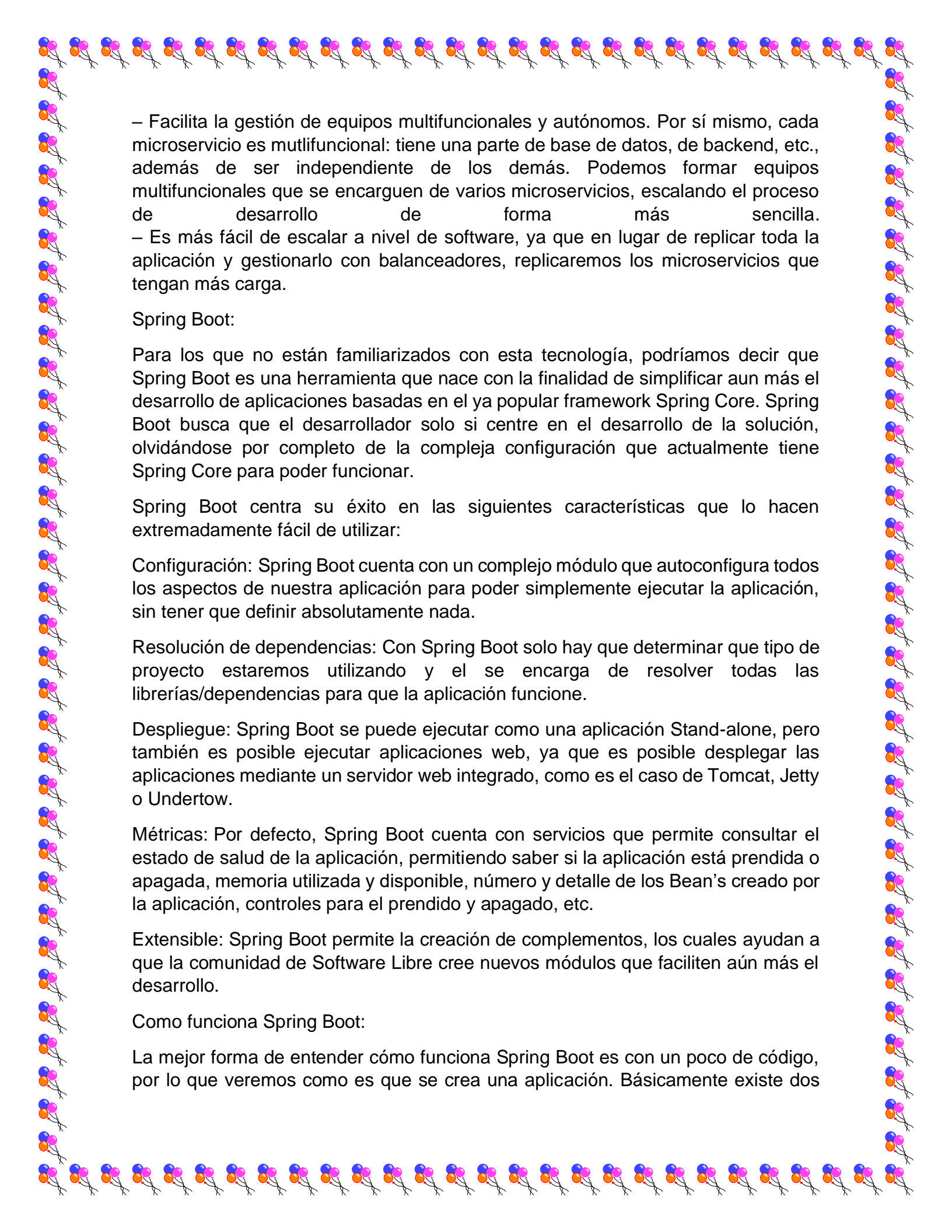
Normalmente hay un número mínimo de servicios que gestionan cosas comunes para los demás (como el acceso a base de datos), pero cada microservicio es pequeño y corresponde a un área de negocio de la aplicación. Además cada uno es independiente y su código debe poder ser desplegado sin afectar a los demás. Incluso cada uno de ellos puede escribirse en un lenguaje de programación diferente, ya que solo exponen la API (una interfaz común, a la que le da igual el lenguaje de programación en la que el microservicio esté programado por debajo) al resto de microservicios. No hay reglas sobre qué tamaño tiene que tener cada microservicio, ni sobre cómo dividir la aplicación en microservicios, pero algunos autores como Jon Eaves caracterizan un microservicio como algo que a nivel de código podría ser reescrito en dos semanas.

Visión de microservicios



En vez de tener un único ejecutable, cada componente es un ejecutable por sí mismo, y los servicios se comunican entre sí a través de llamadas. En este caso también tenemos un microservicio que implementa la interfaz web con la que interactúan los usuarios y cierta lógica por debajo para llamar al microservicio de pagos, inventario y envíos cuando sea necesario. Como beneficios con respecto al anterior enfoque tenemos que:

- Cada microservicio se puede desplegar de forma independiente: un cambio en el módulo de inventario, no afectará a los demás, solo tendremos que subir ese módulo.
- Es fácil de entender, ya que la lógica de negocio está bien separada.

- 
- Facilita la gestión de equipos multifuncionales y autónomos. Por sí mismo, cada microservicio es multifuncional: tiene una parte de base de datos, de backend, etc., además de ser independiente de los demás. Podemos formar equipos multifuncionales que se encarguen de varios microservicios, escalando el proceso de desarrollo de forma más sencilla.
 - Es más fácil de escalar a nivel de software, ya que en lugar de replicar toda la aplicación y gestionarlo con balanceadores, replicaremos los microservicios que tengan más carga.

Spring Boot:

Para los que no están familiarizados con esta tecnología, podríamos decir que Spring Boot es una herramienta que nace con la finalidad de simplificar aun más el desarrollo de aplicaciones basadas en el ya popular framework Spring Core. Spring Boot busca que el desarrollador solo se centre en el desarrollo de la solución, olvidándose por completo de la compleja configuración que actualmente tiene Spring Core para poder funcionar.

Spring Boot centra su éxito en las siguientes características que lo hacen extremadamente fácil de utilizar:

Configuración: Spring Boot cuenta con un complejo módulo que autoconfigura todos los aspectos de nuestra aplicación para poder simplemente ejecutar la aplicación, sin tener que definir absolutamente nada.

Resolución de dependencias: Con Spring Boot solo hay que determinar que tipo de proyecto estaremos utilizando y el se encarga de resolver todas las librerías/dependencias para que la aplicación funcione.

Despliegue: Spring Boot se puede ejecutar como una aplicación Stand-alone, pero también es posible ejecutar aplicaciones web, ya que es posible desplegar las aplicaciones mediante un servidor web integrado, como es el caso de Tomcat, Jetty o Undertow.

Métricas: Por defecto, Spring Boot cuenta con servicios que permite consultar el estado de salud de la aplicación, permitiendo saber si la aplicación está prendida o apagada, memoria utilizada y disponible, número y detalle de los Bean's creado por la aplicación, controles para el prendido y apagado, etc.

Extensible: Spring Boot permite la creación de complementos, los cuales ayudan a que la comunidad de Software Libre cree nuevos módulos que faciliten aún más el desarrollo.

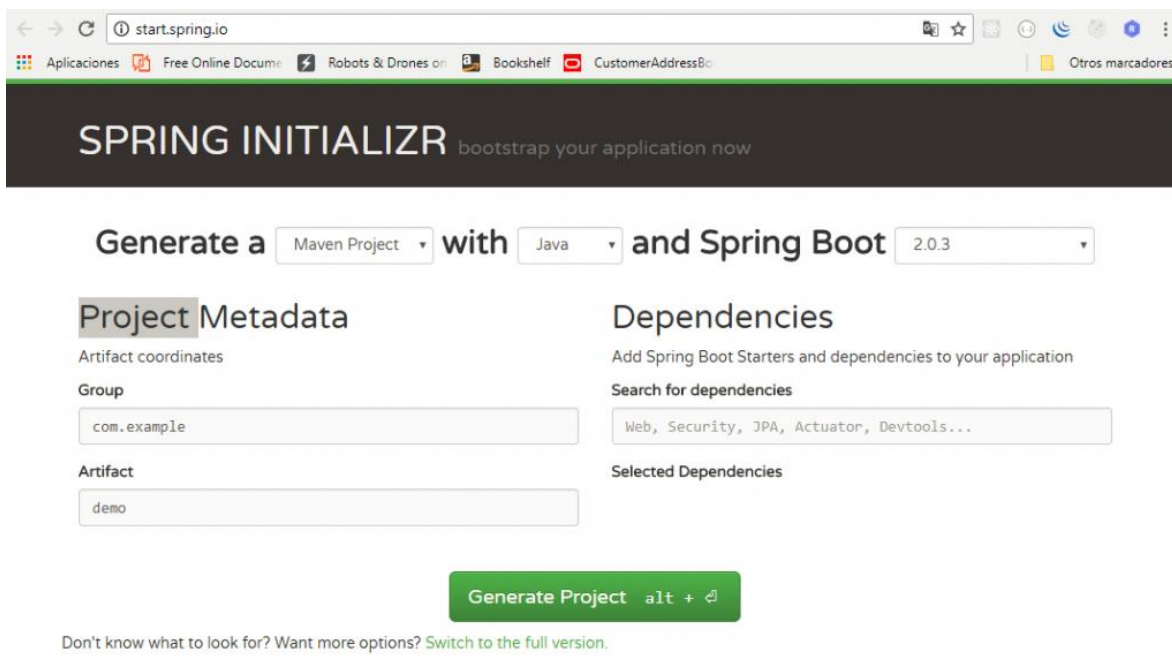
Como funciona Spring Boot:

La mejor forma de entender cómo funciona Spring Boot es con un poco de código, por lo que veremos como es que se crea una aplicación. Básicamente existe dos

formas de crear una aplicación, mediante ayuda del IDE o con ayuda de la página de Spring Boot:

Spring Start:

Spring Start es una página web que provee Spring para crear un proyecto por nosotros, para lo cual, nos solicitará los datos básicos del proyecto:



The screenshot shows the Spring Initializr web application in a browser. The address bar shows 'start.spring.io'. The page has a dark header with the text 'SPRING INITIALIZR bootstrap your application now'. Below the header, there are dropdown menus to 'Generate a' (set to 'Maven Project'), 'with' (set to 'Java'), and 'and Spring Boot' (set to '2.0.3'). The main content area is divided into two sections: 'Project Metadata' and 'Dependencies'. Under 'Project Metadata', there are input fields for 'Group' (containing 'com.example') and 'Artifact' (containing 'demo'). Under 'Dependencies', there is a search bar with the text 'Web, Security, JPA, Actuator, Devtools...' and a section for 'Selected Dependencies'. At the bottom, there is a green button labeled 'Generate Project alt + ⌘'. Below the button, there is a link that says 'Don't know what to look for? Want more options? [Switch to the full version.](#)'

Como podremos observar, es posible crear un proyecto basado en Maven o en Gradle, también podemos definir si el lenguaje será Java, Kotlin o Groovy, también nos pedirá la versión de Spring Boot y el grupo (namespace) y nombre del artefacto (nombre del proyecto), finalmente nos pedirá las que selecciones las dependencias, en nuestro caso, solo seleccionaremos Web.

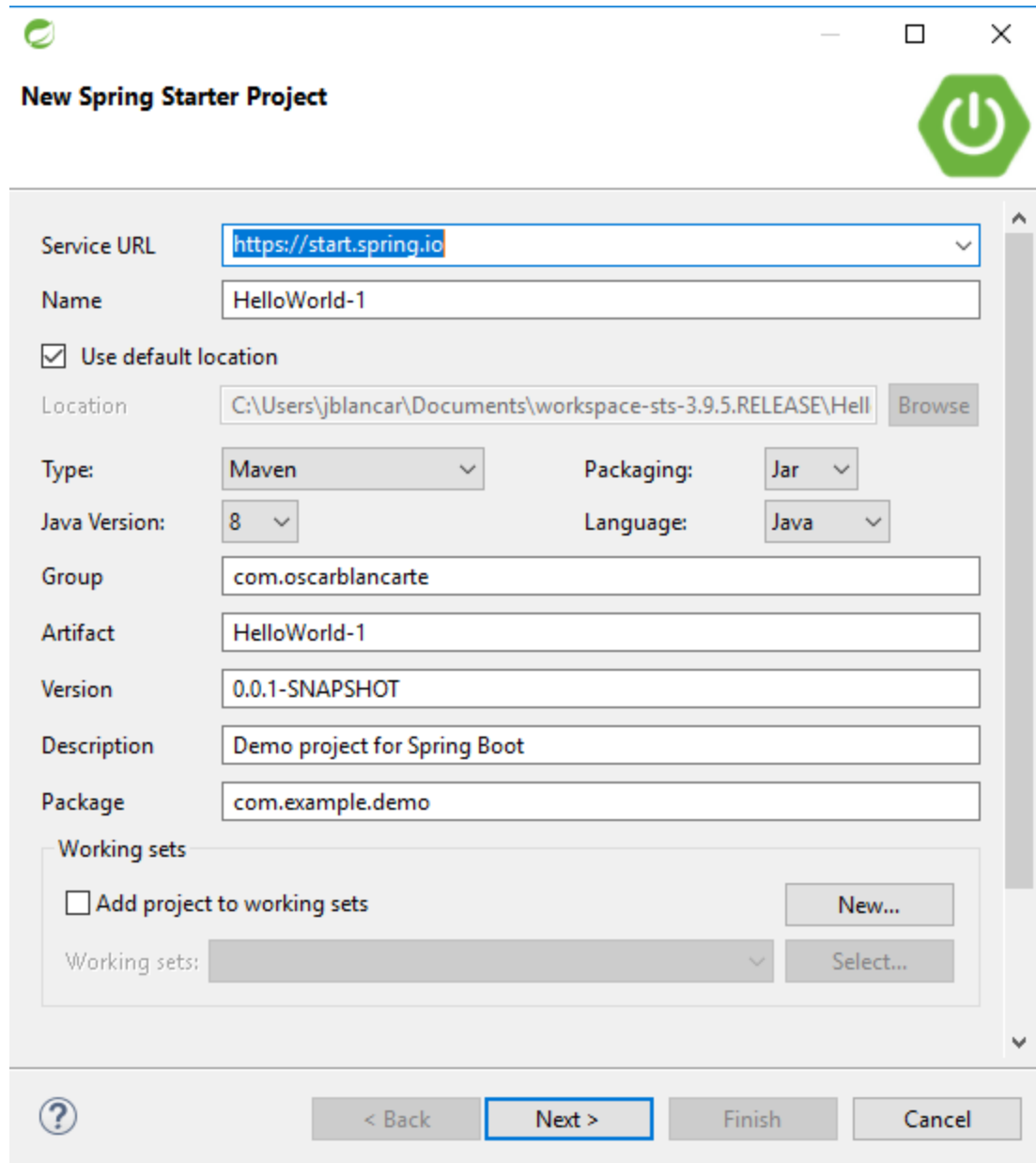
Si desconoces los módulos disponibles, puedes presionar el texto que dice “ Switch to the full versión.” Que se encuentra justo debajo del botón para generar el proyecto (“Generate Project”), para que desplegará todos los módulos disponibles, los cuales son muchísimos para mostrarlos en una imagen.

Una vez configurado el proyecto, solo nos resta presionar el botón “Generate Project” e iniciará la descarga del proyecto preconfigurado, el cual podrás importar en cualquier IDE que soporte Maven o Gladle.

Spring Boot Suite:

La segunda y más recomendable opción es utilizar Spring Boot Suite, el IDE por excelencia para el desarrollo de aplicaciones basadas en cualquier producto de Spring.

Spring Boot Suite es un IDE basado en Eclipse, por lo que no tendrás problemas en utilizarlos rápidamente. Para crear un proyecto tan solo es necesario dirigirse al menú principal y dar click en File => New => Spring Starter Project, el cual desplegará la siguiente pantalla:



New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

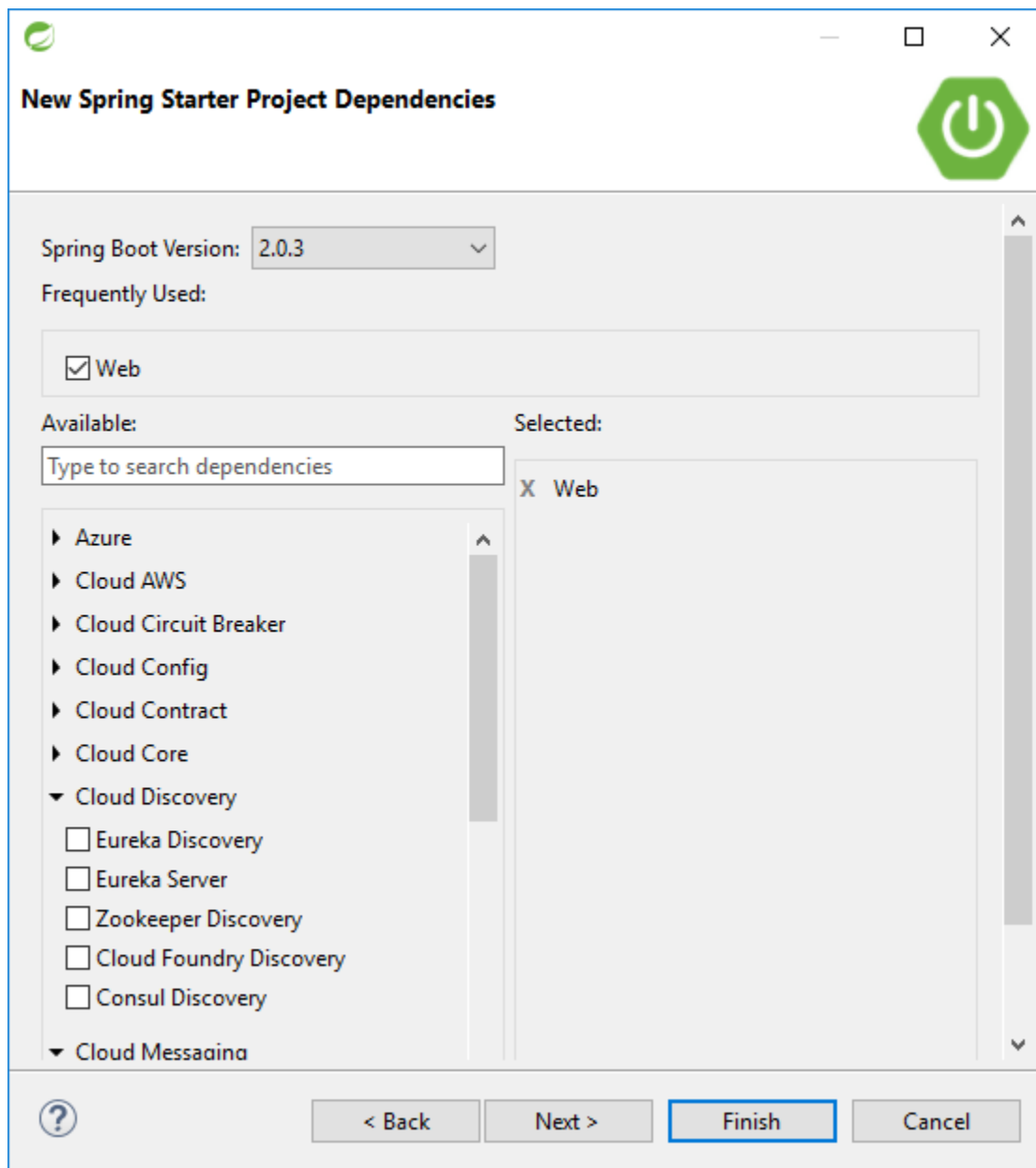
Package:

Working sets

☐ Add project to working sets

Working sets:

Como podemos ver, nos pedirá los mismos datos que la página web, como grupo, nombre, type de proyecto (Maven Gladle), etc. Presionamos Next para continuar:



The image shows a 'New Spring Starter Project Dependencies' dialog box. At the top, it has a title bar with a green icon and a power button icon. Below the title bar, there is a 'Spring Boot Version:' dropdown menu set to '2.0.3'. Underneath, it says 'Frequently Used:' followed by a checkbox for 'Web' which is checked. The main area is divided into two sections: 'Available:' and 'Selected:'. The 'Available:' section has a search bar with the text 'Type to search dependencies' and a list of dependencies. The 'Selected:' section shows 'X Web'. At the bottom, there are four buttons: a help button (question mark), '< Back', 'Next >', and 'Finish' (which is highlighted with a blue border). A 'Cancel' button is also present.

New Spring Starter Project Dependencies

Spring Boot Version: 2.0.3

Frequently Used:

☒ Web

Available:

Type to search dependencies

- ▶ Azure
- ▶ Cloud AWS
- ▶ Cloud Circuit Breaker
- ▶ Cloud Config
- ▶ Cloud Contract
- ▶ Cloud Core
- ▼ Cloud Discovery
 - ☐ Eureka Discovery
 - ☐ Eureka Server
 - ☐ Zookeeper Discovery
 - ☐ Cloud Foundry Discovery
 - ☐ Consul Discovery
- ▼ Cloud Messaging

Selected:

X Web

? < Back Next > Finish Cancel

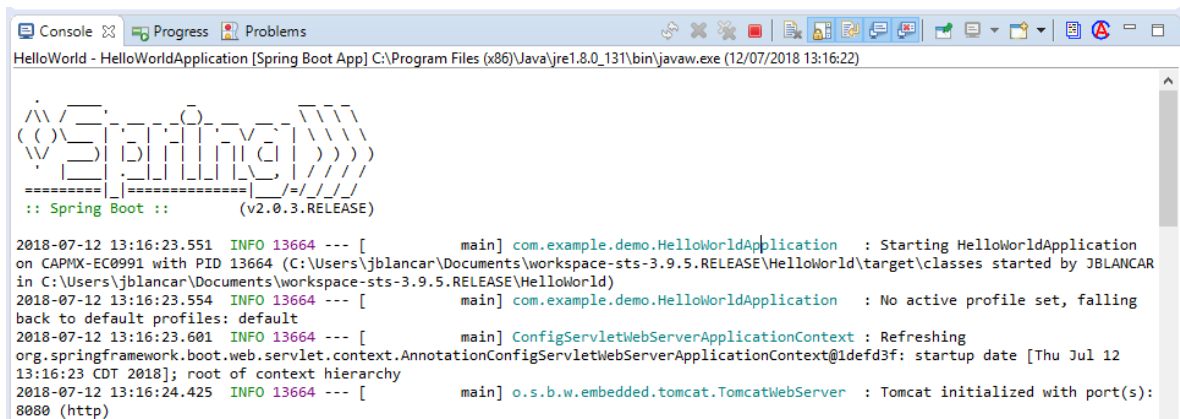
En esta nueva pantalla nos pedirá que seleccionemos las dependencias, en nuestro caso, solo seleccionaremos el módulo Web y damos en finalizar.


```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class HelloWorldApplication {
8     public static void main(String[] args) {
9         SpringApplication.run(HelloWorldApplication.class, args);
10    }
11 }
```

En este momento podríamos ejecutar nuestra aplicación Web, pero no tendría mucho sentido, pues no hemos desarrollado nada, por ese motivo, vamos a crear un servicio REST para poder probarlo desde el navegador. Para ello, crearemos una nueva clase llamada HelloRest en el mismo paquete que la clase anterior:

```
1 package com.example.demo;
2
3 import org.springframework.web.bind.annotation.RequestMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 public class HelloRest {
8     @RequestMapping("/hello")
9     public String helloWorld() {
10         return "Hello World";
11     }
12 }
```

Una vez creada, ejecutamos la aplicación dando click derecho sobre el proyecto y seleccionando Run As => Spring Boot App, lo que iniciará el despliegue de la aplicación sobre un Tomcat embebido:



```
Console Progress Problems
HelloWorld - HelloWorldApplication [Spring Boot App] C:\Program Files (x86)\Java\jre1.8.0_131\bin\javaw.exe (12/07/2018 13:16:22)

:: Spring Boot :: (v2.0.3.RELEASE)

2018-07-12 13:16:23.551 INFO 13664 --- [main] com.example.demo.HelloWorldApplication : Starting HelloWorldApplication
on CAPMX-EC0991 with PID 13664 (C:\Users\jblancar\Documents\workspace-sts-3.9.5.RELEASE\HelloWorld\target\classes started by JBLANCAR
in C:\Users\jblancar\Documents\workspace-sts-3.9.5.RELEASE\HelloWorld)
2018-07-12 13:16:23.554 INFO 13664 --- [main] com.example.demo.HelloWorldApplication : No active profile set, falling
back to default profiles: default
2018-07-12 13:16:23.601 INFO 13664 --- [main] ConfigServletWebServerApplicationContext : Refreshing
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@1defd3f: startup date [Thu Jul 12
13:16:23 CDT 2018]; root of context hierarchy
2018-07-12 13:16:24.425 INFO 13664 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s):
8080 (http)
```

Lo siguiente será ir al navegador y entrar a la URL <http://localhost:8080/hello> para ver lo siguiente:

