

PRACTICAS DE LABORATORIO  
PARA EL DESARROLLO DE MICROSERVICIOS  
CON JAVA EE, SPRING BOOT

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

## INTRODUCCIÓN

### Sesión 1 Lunes 18 de Mayo 2hrs

#### Introducción

1. Introducción a Arquitectura Microservicio
2. Herramientas y plugins para implementar microservicios Eclipse o Spring STS
3. Implementar un microservicio desde el inicio (Spring initializr)
4. Dependencias principales (pom.xml)
5. Implementar funcionalidad con Base de Datos
6. Generar servicio REST a partir de la información de BD
7. Crear Jar de microservicio
8. Pruebas invocaciones del microservicio (Postman)

### Sesión 2 Miercoles 20 de Mayo 2hrs

#### Arquitectura

1. Configurar Microservicio Service Registry
2. Configurar Microservicio Gateway
3. Configurar Microservicio Config
4. Microservicio propio funcionando con los microservicios Service Registry, Gateway y Config
5. Documentación servicios REST (Eureka, Swagger, Zipkin)

### Sesión 3 Jueves 21 de Mayo 2hrs

#### Arquitectura

1. Configurar Microservicio Seguridad
2. Configurar Microservicio Notificaciones y Auditoria
3. Microservicio propio funcionando con los microservicios de Seguridad, Noficaciones y Auditoria

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

## DESARROLLO

La 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. Cada microservicio 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 al resto de microservicios.

Muchos desarrolladores están descubriendo cómo esta forma de creación de software favorece el tiempo, rendimiento y estabilidad de sus proyectos.

No hay reglas sobre qué tamaño tiene que tener cada microservicio, ni sobre cómo dividir la aplicación en microservicio. Algunos autores, caracterizan un microservicio como algo que a nivel de código podría ser reescrito en dos semanas. Por lo tanto, se tendrá una aplicación modular a base de “pequeñas piezas”, que se pueden ir ampliando o reduciendo a medida que se requiera.

Los microservicios ofrecen una serie de beneficios potenciales tanto sobre enfoques más tradicionales como SOA y arquitecturas monolíticas. Los microservicios, cuando se hacen bien, son maleables, escalables y resilientes, y se pueden implementar en corto plazo desde el inicio de la implementación hasta el despliegue en la producción. Esta combinación, a menudo resulta difícil de alcanzar para sistemas de software complejos.

Gracias a su sencilla escalabilidad, la arquitectura de microservicios se considera especialmente adecuada cuando se tiene que procurar la compatibilidad con una amplia variedad de plataformas. Por ejemplo IoT, web, móvil, wearables, o simplemente cuando no se sabe hacia qué tipo de dispositivos se está orientando el trabajo.

La forma en que un microservicio se comunica con otros dependerá de las habilidades de los desarrolladores, como de las preferencias o el sistema que mejor se adapte a los requerimientos. Muchos usan HTTP/REST con JSON o Protobuf. REST se está posicionando en extensión de uso, al ser uno de los métodos de integración cuya curva de aprendizaje y uso es más rápida respecto a otros protocolos.

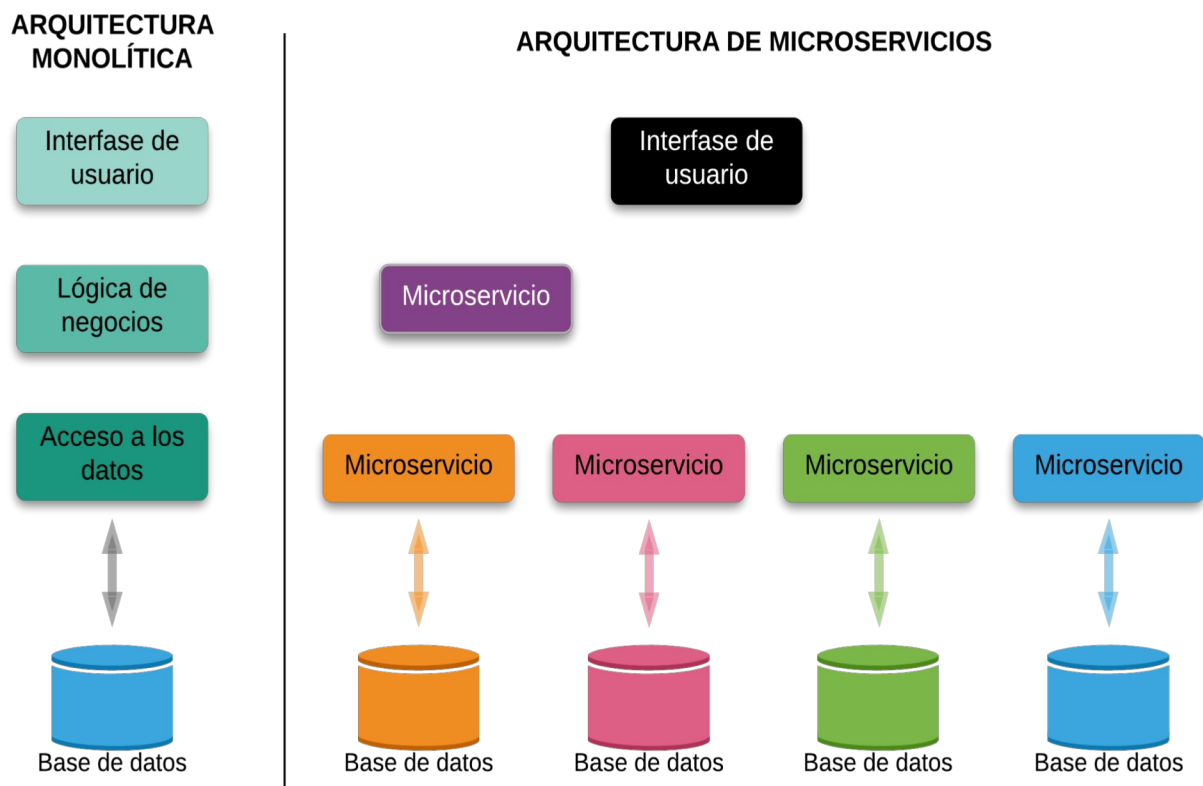
### ¿Para qué sirven los microservicios?

Es más fácil diseñar, probar, implementar y actualizar microservicios que aplicaciones monolíticas. Red Hat considera que esto responde a la pregunta "¿cómo logro que mi empresa reaccione más rápido ante las demandas nuevas, en lugar de tener que esperar la cantidad de años que supone el desarrollo tradicional de software?". En la actualidad, las distintas partes del equipo de desarrollo pueden trabajar simultáneamente en los productos de un modo ágil para ofrecer beneficios a los clientes de inmediato.

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

Conozca los aspectos fundamentales de los microservicios y las ventajas y desventajas de utilizarlos, consulte nuestro curso [Developing Cloud-Native Applications with Microservices Architectures](#) y entérese de cómo diseñar una arquitectura basada en microservicios.

Los microservicios son un tipo de arquitectura que sirve para diseñar aplicaciones. Lo que distingue a la arquitectura de microservicios de los enfoques tradicionales y monolíticos es la forma en que desglosa una aplicación en sus funciones principales. Cada función se denomina servicio y se puede diseñar e implementar de forma independiente. Esto permite que funcionen separados (y también, fallen por separado) sin afectar a los demás.



DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

## Microservicio pros y contras

La arquitectura de microservicios plantea una serie de concesiones para los desarrolladores de software. En términos de ventajas, se observan:

- Se despliegan fácilmente.
- Requieren menos tiempo de desarrollo.
- Puede escalar rápidamente
- Se puede reutilizar en diferentes proyectos.
- Contienen mejor aislamiento de fallas.
- Puede ser desplegado en equipos relativamente pequeños.
- Trabaja bien con los contenedores.

Sin embargo, también hay inconvenientes con la arquitectura de microservicios, tales como:

- Potencialmente demasiada granularidad.
- Esfuerzo extra de diseño para la comunicación entre servicios.
- Latencia durante el uso pesado.
- Pruebas complejas.

Mas sobre ventajas y desventajas da la implantación de microservicios

### Ventajas

Agilidad. Dado que microservicios se implementan de forma independiente, resulta más fácil de administrar las correcciones de errores y las versiones de características. Puede actualizar un servicio sin volver a implementar toda la aplicación y revertir una actualización si algo va mal. En muchas aplicaciones tradicionales, un error en una parte de la aplicación puede bloquear todo el proceso de lanzamiento. Es posible que se requieran nuevas características a la espera de que se integre, pruebe y publique una corrección de errores.

Equipos pequeños y centrados. Un microservicio debe ser lo suficientemente pequeño como para que un solo equipo de características lo pueda compilar, probar e implementar. Los equipos pequeños favorecen la agilidad. Los equipos grandes suelen ser menos productivos, porque la comunicación es más lenta, aumenta la sobrecarga de administración y la agilidad disminuye.

Base de código pequeña. En las aplicaciones monolíticas, con el paso del tiempo se da la tendencia de que las dependencias del código se acaben enredarse, por lo que para agregar una nueva

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

característica, es preciso tocar el código en muchos lugares. Al no compartir el código ni los almacenes de datos, la arquitectura de microservicios minimiza las dependencias y resulta más fácil agregar nuevas características.

**Mezcla de tecnologías.** Los equipos pueden elegir la tecnología que mejor se adapte al servicio de una combinación de pilas de tecnología, según corresponda.

**Aislamiento de errores.** Si un microservicio individual no está disponible, no interrumpe toda la aplicación, siempre que los microservicios de nivel superior estén diseñados para controlar los errores correctamente (por ejemplo, mediante la implementación de disyuntores).

**Escalabilidad.** Los servicios se pueden escalar de forma independiente, lo que permite escalar horizontalmente los subsistemas que requieren más recursos, sin tener que escalar horizontalmente toda la aplicación. Mediante un orquestador como Kubernetes o Service Fabric se puede empaquetar una mayor densidad de servicios en un solo host, lo que aumenta la eficacia en el uso de los recursos.

**Aislamiento de los datos.** Al verse afectado solo un microservicio, es mucho más fácil realizar actualizaciones del esquema. En una aplicación monolítica, las actualizaciones del esquema pueden ser muy complicadas, ya que las distintas partes de la aplicación pueden tocar los mismos datos, por lo que realizar modificaciones en el esquema resulta peligroso.

## Desafíos

Las ventajas de los microservicios tienen un "precio". Estos son algunos de los aspectos que deben tenerse en cuenta antes de embarcarse en una arquitectura de microservicios.

**Complejidad.** Una aplicación de microservicios tiene más partes en movimiento que la aplicación monolítica equivalente. Cada servicio es más sencillo, pero el sistema como un todo es más complejo.

**Desarrollo y pruebas.** La escritura de un servicio pequeño que utilice otros servicios dependientes requiere un enfoque que no sea escribir una aplicación tradicional monolítica o en capas. Las herramientas existentes no siempre están diseñadas para trabajar con dependencias de servicios. La refactorización en los límites del servicio puede resultar difícil. También supone un desafío probar las dependencias de los servicios, especialmente cuando la aplicación está evolucionando rápidamente.

**Falta de gobernanza.** El enfoque descentralizado para la generación de microservicios tiene ventajas, pero también puede causar problemas. Puede acabar con tantos lenguajes y marcos de trabajo diferentes que la aplicación puede ser difícil de mantener. Puede resultar útil establecer algunos

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

estándares para todo el proyecto sin restringir excesivamente la flexibilidad de los equipos. Esto se aplica especialmente a las funcionalidades transversales, como el registro.

**Congestión y latencia de red.** El uso de muchos servicios pequeños y detallados puede dar lugar a más comunicación interservicios. Además, si la cadena de dependencias del servicio se hace demasiado larga (el servicio A llama a B, que llama a C...), la latencia adicional puede constituir un problema. Tendrá que diseñar las API con atención. Evite que las API se comuniquen demasiado, piense en formatos de serialización y busque lugares para utilizar patrones de comunicación asincrónica.

**Integridad de datos.** Cada microservicio es responsable de la conservación de sus propios datos. Como consecuencia, la coherencia de los datos puede suponer un problema. Adopte una coherencia final cuando sea posible.

**Administración.** Para tener éxito con los microservicios se necesita una cultura de DevOps consolidada. El registro correlacionado entre servicios puede resultar un desafío. Normalmente, el registro debe correlacionar varias llamadas de servicio para una sola operación de usuario.

**Control de versiones.** Las actualizaciones de un servicio no deben interrumpir servicios que dependen de él. Es posible que varios servicios se actualicen en cualquier momento; por lo tanto, sin un cuidadoso diseño, podrían surgir problemas con la compatibilidad con versiones anteriores o posteriores.

**Conjunto de habilidades.** Los microservicios son sistemas muy distribuidos. Evalúe cuidadosamente si el equipo tiene los conocimientos y la experiencia para desenvolverse correctamente.

## Microservicios y DevOps

DevOps combina tareas entre la aplicación y los equipos de operaciones del sistema. Con el aumento de las comunicaciones entre los desarrolladores y el personal de operaciones, un equipo de TI puede crear y administrar mejor la infraestructura. Las operaciones de TI aseguran el presupuesto para capacidades, tareas operativas, actualizaciones y más. Los desarrolladores y los equipos de aplicaciones pueden administrar bases de datos, servidores, software y hardware utilizados en la producción.

El trabajo en equipo y la colaboración entre los equipos de desarrollo y operaciones son necesarios para soportar el ciclo de vida de los microservicios, prestándose a los equipos de DevOps. También es la razón por la que los equipos de DevOps experimentados están bien equipados para emplear arquitectura de microservicios en proyectos de desarrollo de software.

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

## Seguridad de la arquitectura de microservicios

La arquitectura de microservicios puede aliviar algunos problemas de seguridad que surgen con las aplicaciones monolíticas. Los microservicios simplifican el monitoreo de seguridad porque las diversas partes de una aplicación están aisladas. Una brecha de seguridad podría ocurrir en una sección sin afectar otras áreas del proyecto. Los microservicios ofrecen resistencia contra los ataques distribuidos de denegación de servicio (DDoS) cuando se usan con contenedores al minimizar una toma de control de la infraestructura con demasiadas solicitudes de servidor.

Sin embargo, todavía existen desafíos al proteger aplicaciones de microservicios, que incluyen:

Más áreas de red están abiertas a vulnerabilidades.

Una menor coherencia general entre las actualizaciones de la aplicación permite más violaciones de seguridad.

Hay una mayor área de ataque, a través de múltiples puertos y APIs; sin embargo esta debilidad actualmente puede ser cubierta con la inclusión de microservicio especializado para el acceso y enrutamiento de recursos denominado API GATEWAY y OAUTH2.

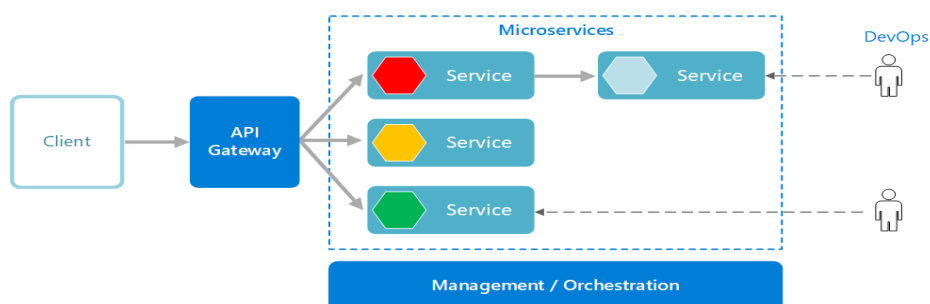
Hay una falta de control de software de terceros.

La seguridad debe mantenerse para cada servicio.

Los desarrolladores de microservicios han ideado estrategias para aliviar los problemas de seguridad. Para ser proactivo, use un escáner de seguridad, utilice limitaciones de control de acceso, redes internas seguras, incluidos los entornos Docker, y opere fuera de los silos, comunicándose con todas las partes de la operación.

## Estilo de arquitectura de microservicios

Una arquitectura de microservicios consta de una colección de servicios autónomos y pequeños. Los servicios son independientes entre sí y cada uno debe implementar una funcionalidad de negocio individual.



DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

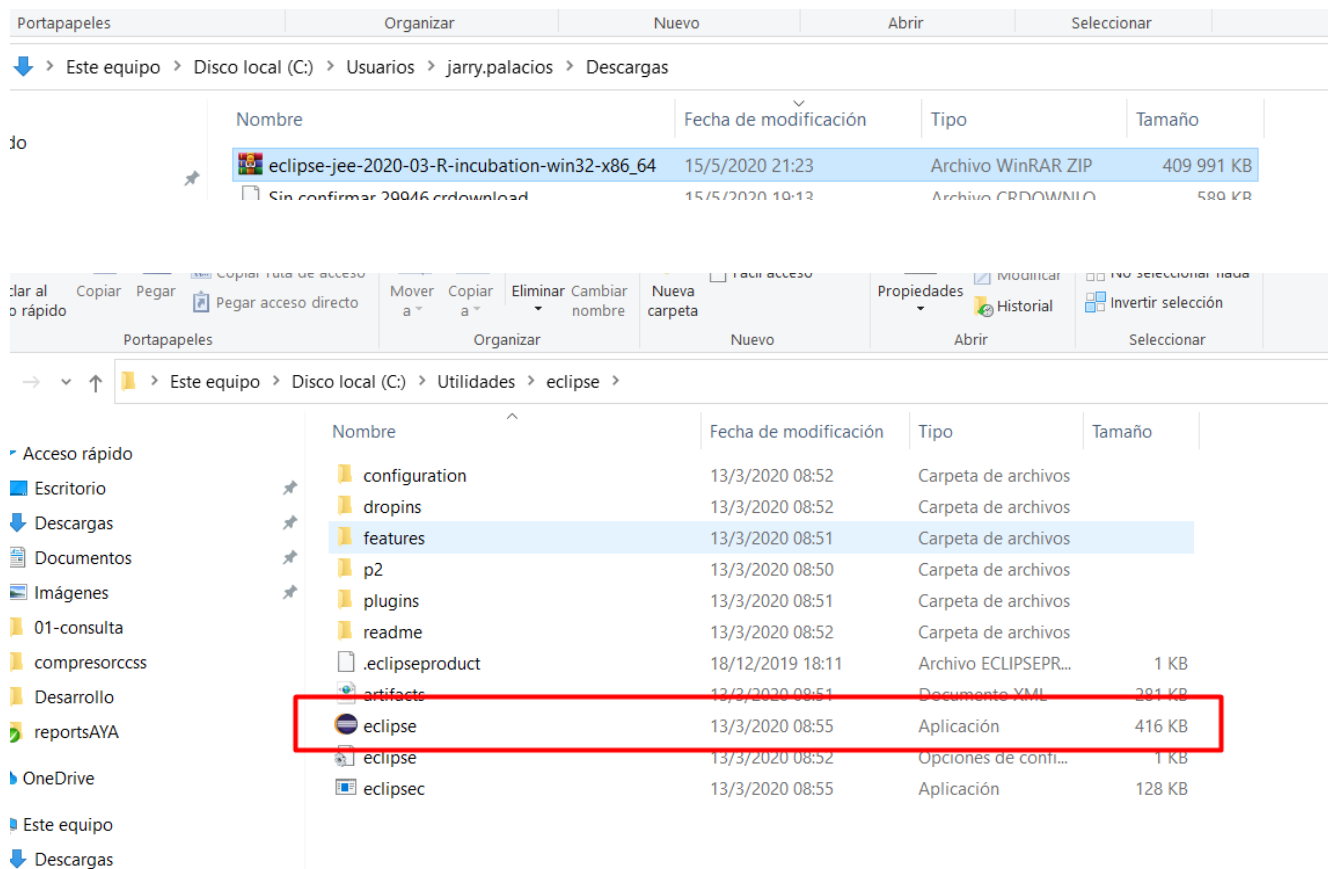


## SOFTWARE DE PLATAFORMA

Java SE Development Kit 8u251 64 Bits	<a href="#">Descargar</a>
Eclipse Ide for enterprise Java Developers	<a href="#">Descargar</a>
Spring Tools Suite (STS)	<a href="#">Descargar</a>
Spring Initializr	<a href="#">Descargar</a>
POSTMAN	<a href="#">Descargar</a>

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

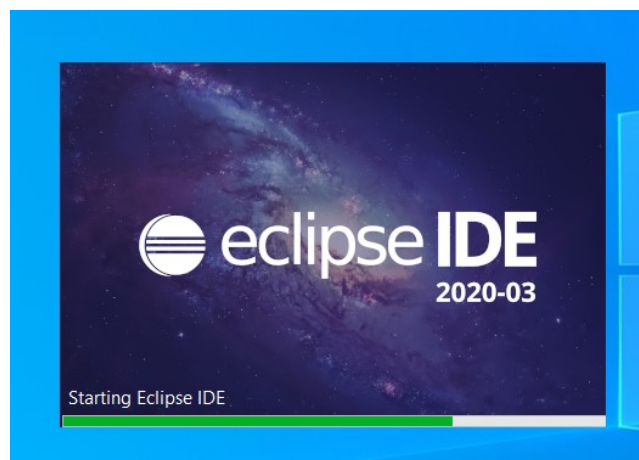
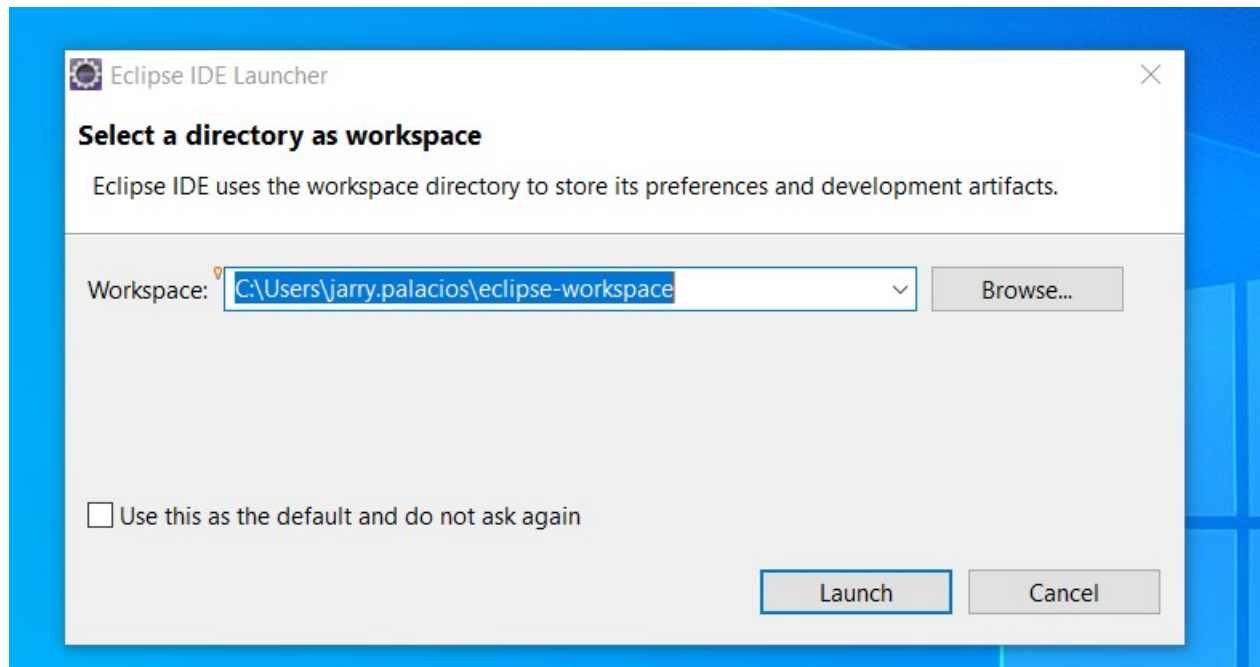
## PREPARAR IDE ECLIPSE 2020-03



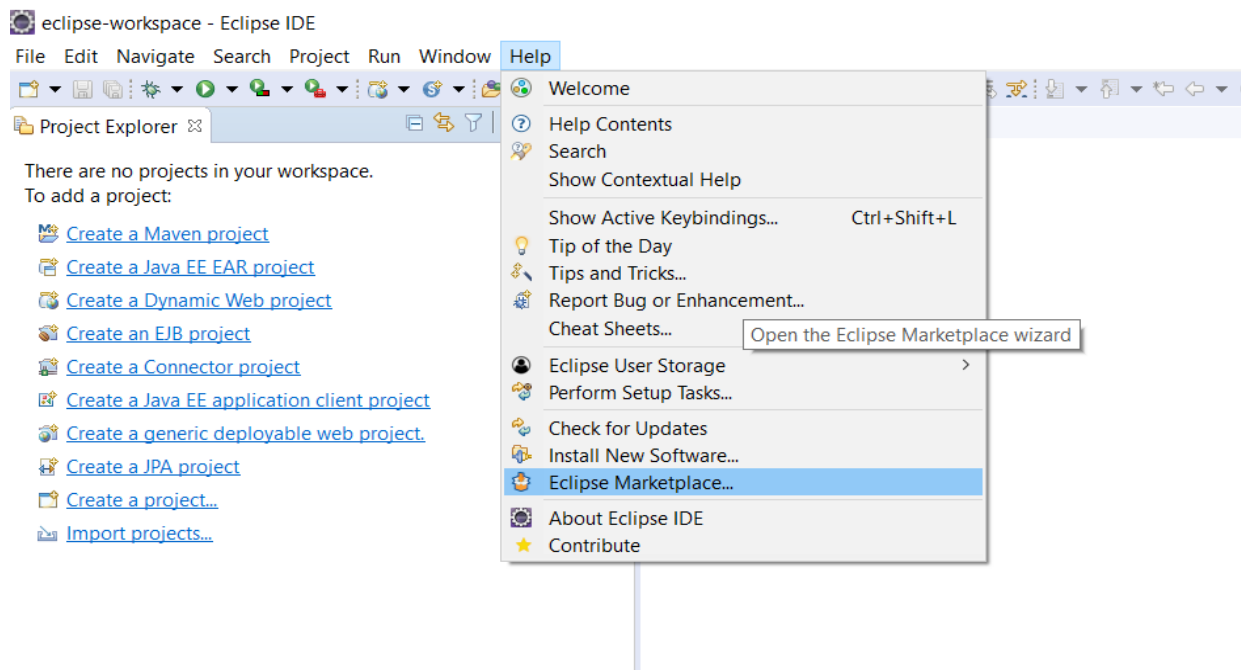
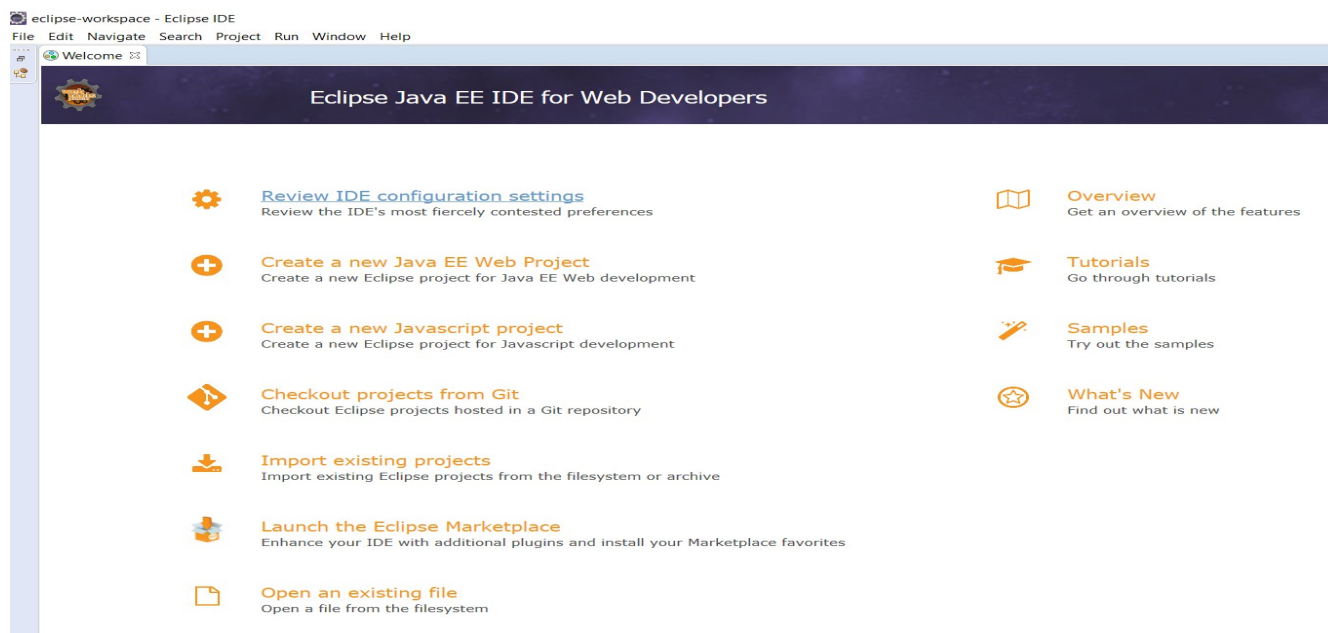
The screenshot shows a Windows File Explorer window with the address bar set to 'Este equipo > Disco local (C:) > Usuarios > jarry.palacios > Descargas'. The file list shows two files: 'eclipse-jee-2020-03-R-incubation-win32-x86\_64' (409,991 KB) and 'Sin confirmar 20046.crdownload' (580 KB). The first file is selected. The left sidebar shows the 'Descargas' folder selected under 'Acceso rápido'.



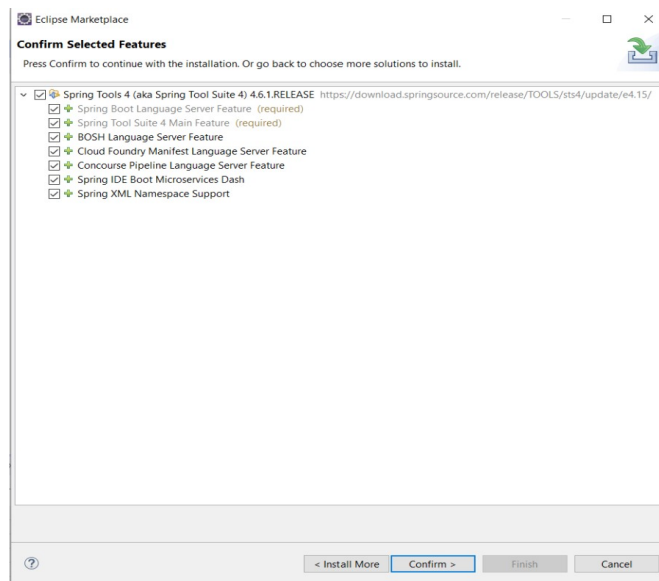
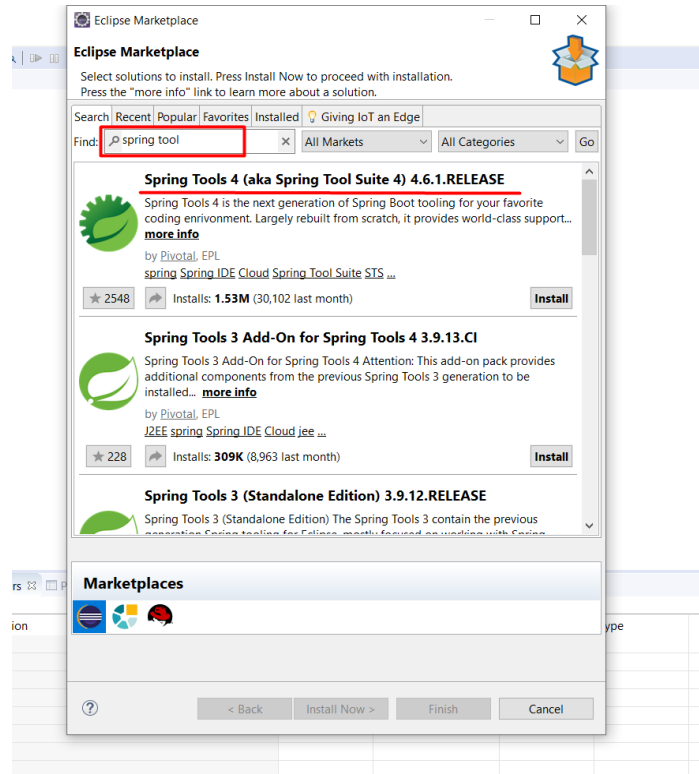
DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	



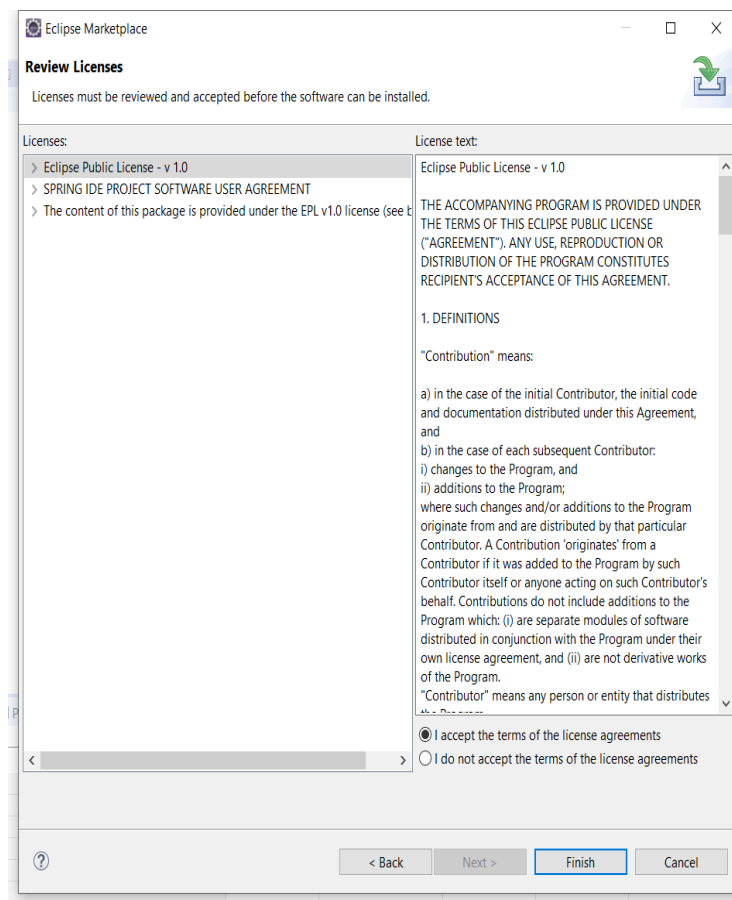
DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	



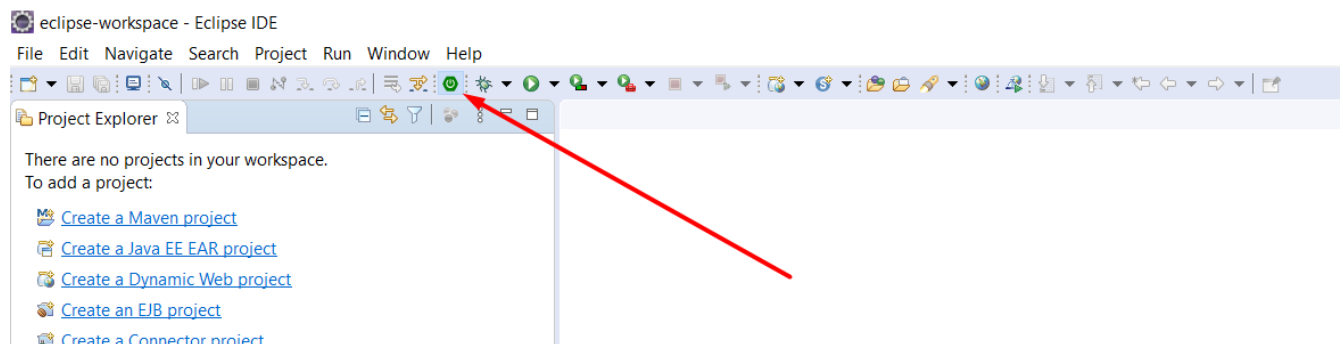
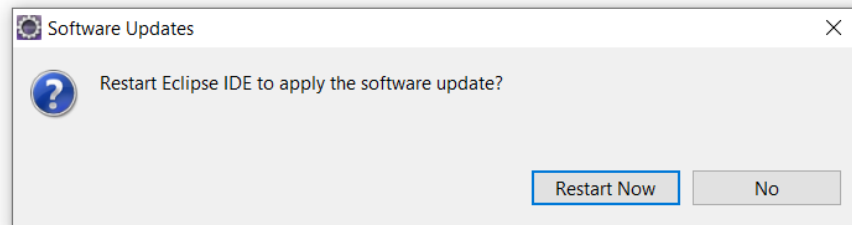
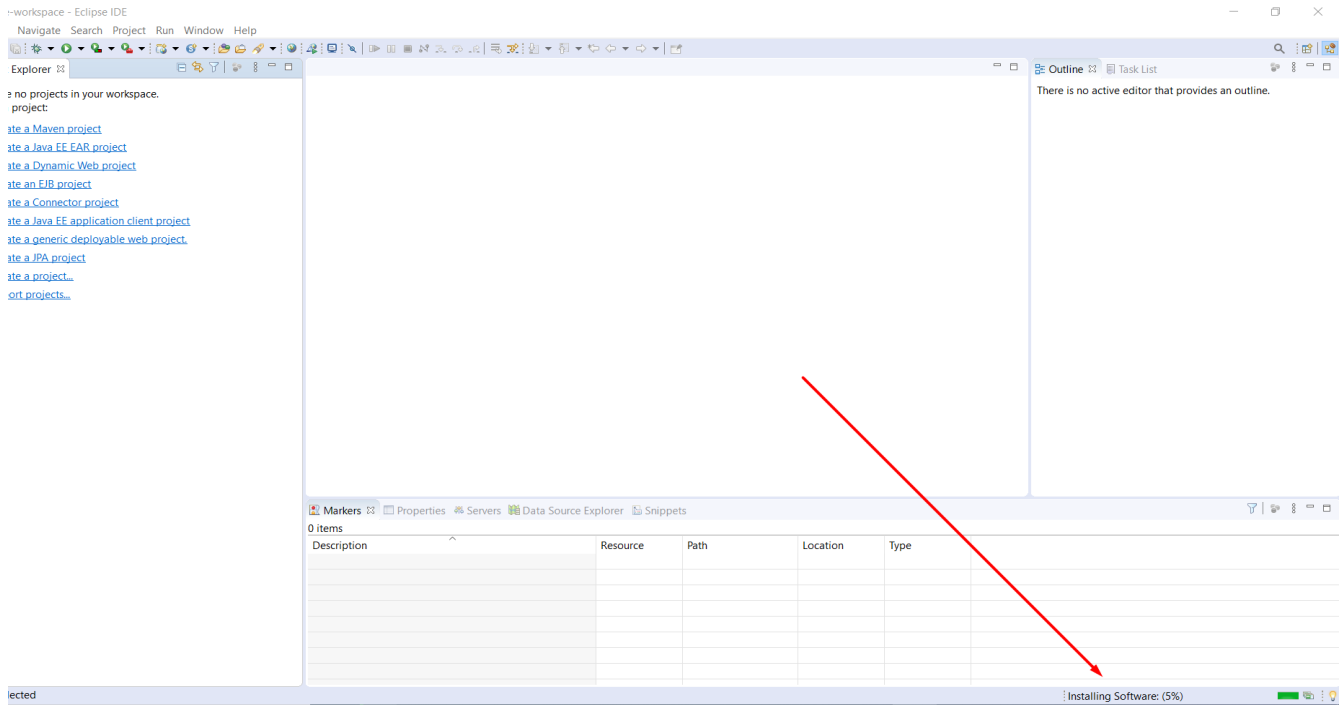
DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	



DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

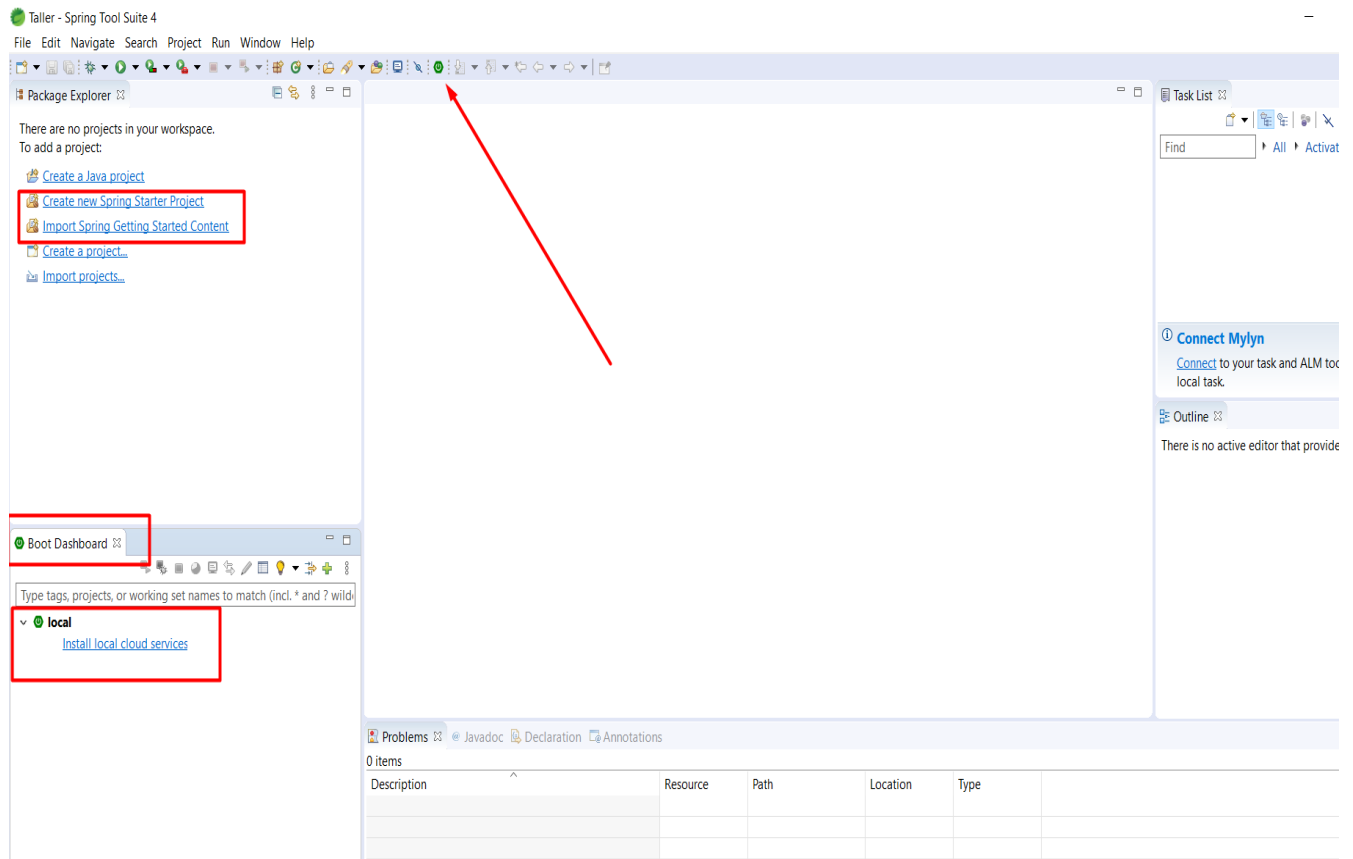


DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	



DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

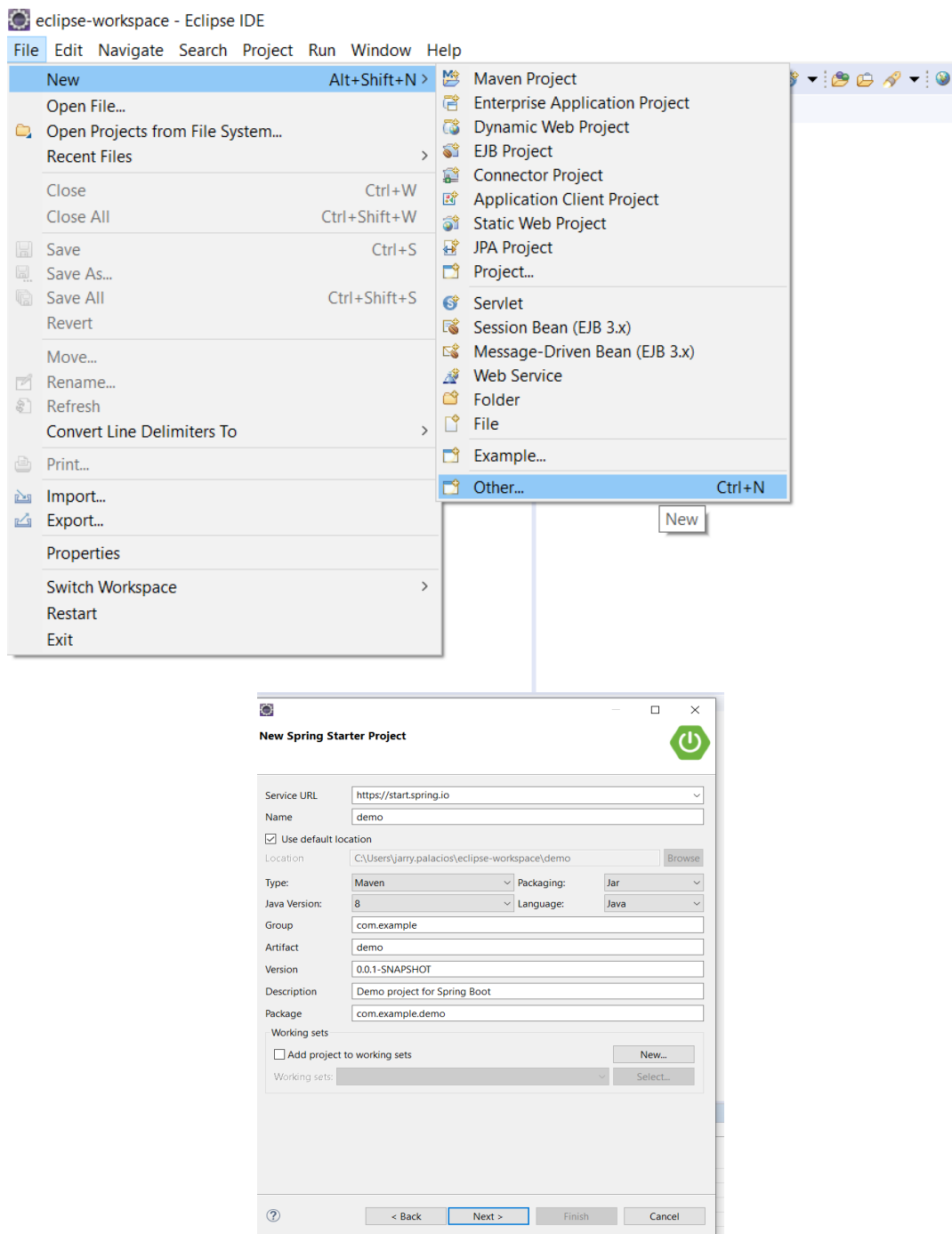
## SPRING TOOLS SUITES



DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

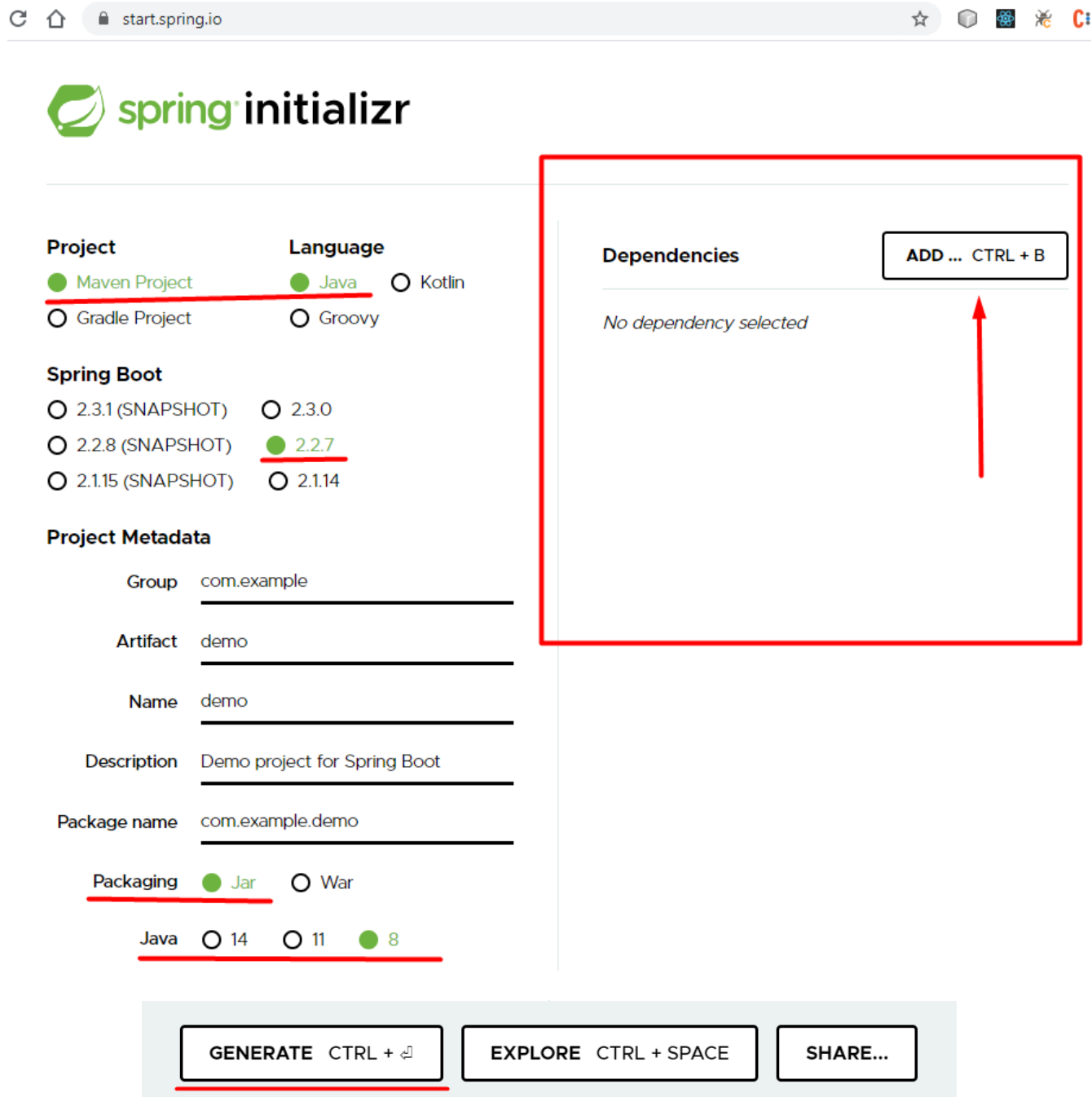


## CREAR UN NUEVO PROYECTO CON ECLIPSE



DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

## CREAR UN NUEVO PROYECTO CON SPRING INITIALIZR



start.spring.io

**spring initializr**

**Project**

☒ Maven Project ☐ Gradle Project

**Language**

☒ Java ☐ Kotlin ☐ Groovy

**Spring Boot**

☐ 2.3.1 (SNAPSHOT) ☐ 2.3.0 ☒ 2.2.7 ☐ 2.1.15 (SNAPSHOT) ☐ 2.1.14

**Project Metadata**

Group

Artifact

Name

Description

Package name

**Packaging**

☒ Jar ☐ War

**Java**

☐ 14 ☐ 11 ☒ 8

**Dependencies**

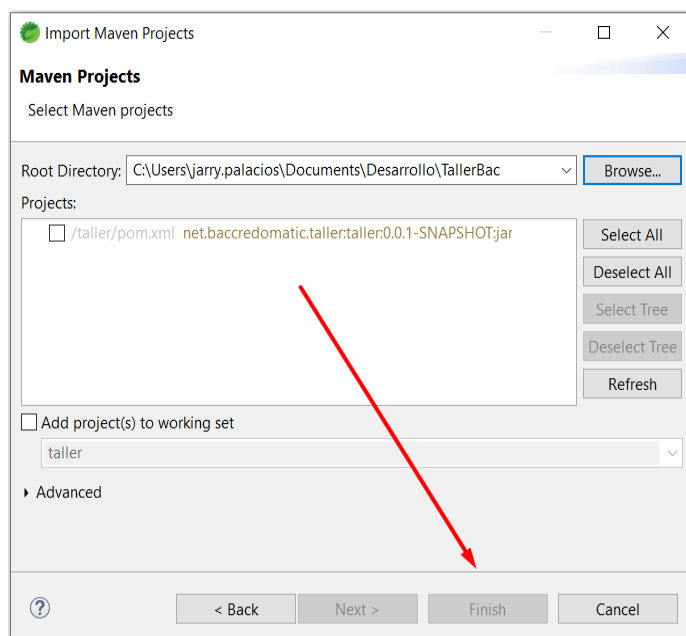
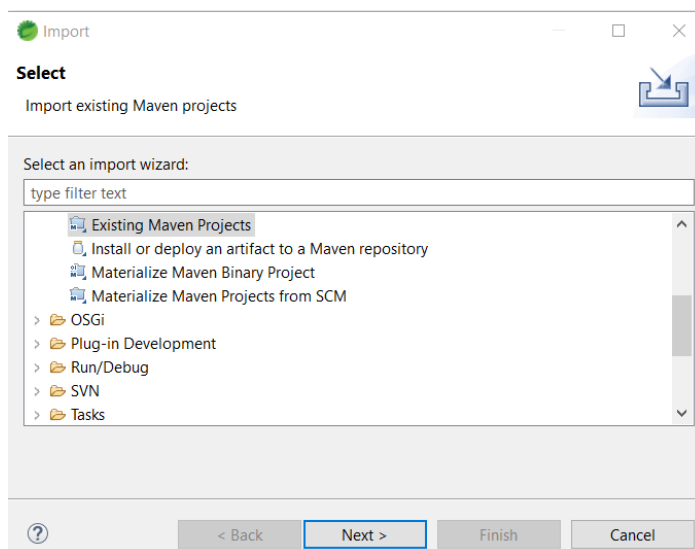
**ADD ... CTRL + B**

No dependency selected

**GENERATE CTRL + G** **EXPLORE CTRL + SPACE** **SHARE...**

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

Al hacer click sobre el botón GENERATE resulta un archivo comprimido con los archivos del proyecto, se deben descomprimir e importar en el IDE de desarrollo usando la opción Existing Maven Projects.

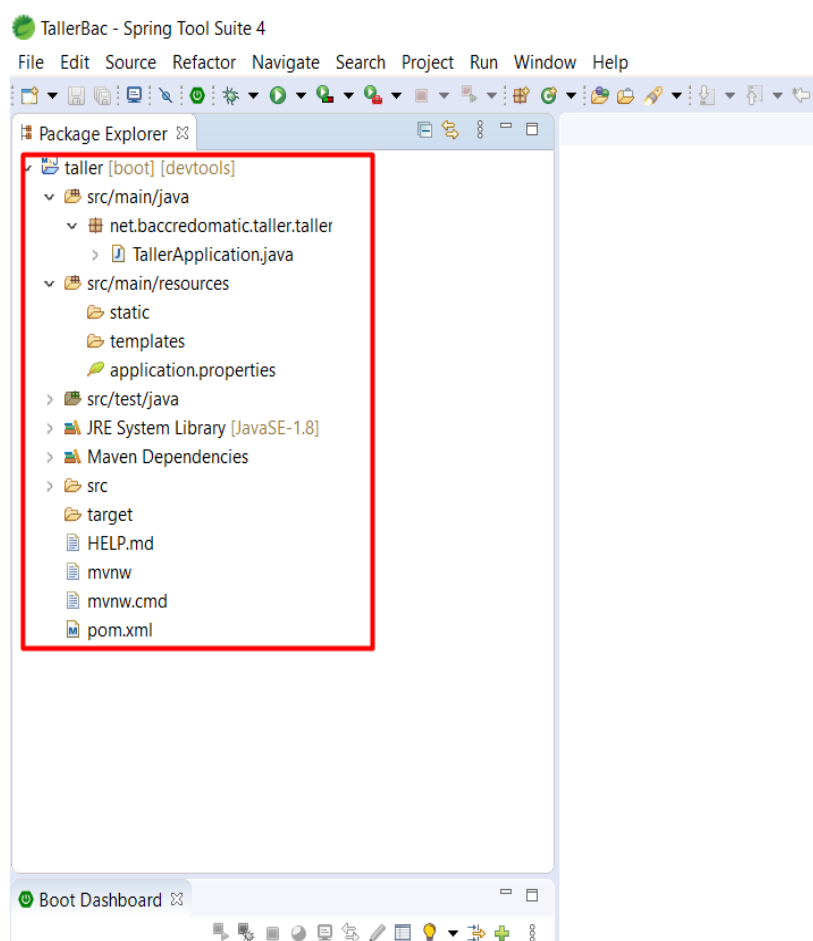


DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

Al importar el proyecto se debe esperar unos instantes para que maven descargue las dependencias presentes en el aechivo POM.XML, inialmente en el proyecto solo apreciara los siguientes:

.mvn			Carpeta de archivos	16/5/2020 05:18	
src			Carpeta de archivos	16/5/2020 05:18	
.gitignore	333	220	Documento de texto	16/5/2020 05:18	03D5F075
HELP.md	1 110	459	Archivo MD	16/5/2020 05:18	76D5E4E9
mvnw	10 070	3 246	Archivo	16/5/2020 05:18	95D62CDD
mvnw.cmd	6 608	2 519	Script de comandos de ...	16/5/2020 05:18	F277415F
pom.xml	1 864	609	Documento XML	16/5/2020 05:18	CDFE2240

Luego de cumplirse el proceso de descarga de maven podrá apreciar la siguiente estructura de directorios:



DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

## Configuraciones iniciales:

### 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-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.7.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>
  <groupId>net.baccredomatic.taller</groupId>
  <artifactId>taller</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>taller</name>
  <description>Proyecto para taller</description>

  <properties>
    <java.version>1.8</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-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>com.microsoft.sqlserver</groupId>
      <artifactId>mssql-jdbc</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
      <exclusions>
        <exclusion>
          <groupId>org.junit.vintage</groupId>
          <artifactId>junit-vintage-engine</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
  </dependencies>
</project>
```

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

```
        </dependency>
    </dependencies>

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

En la sección de src/main/resources editamos el fichero **application.properties** para agregar los siguientes:

```
spring.application.name=taller-agency-api
server.port=9000
```

```
spring.datasource.url = jdbc:sqlserver://IP:PUERTO;databaseName=db_name;
spring.datasource.username = user
spring.datasource.password = contraseña
```

### Clase Principal

```
1 package net.baccredomatic.taller.taller;
2
3 import org.springframework.boot.SpringApplication;
4
5
6 @SpringBootApplication
7 public class TallerApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(TallerApplication.class, args);
11     }
12
13 }
14
```

Proyecto de Ejemplo: <https://github.com/jarryp/taller-agency-api>

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	

URLs para ejecutar peticiones:

<http://localhost:9000/rest/agency/all>

<http://localhost:9000/rest/agency/CURRIDABAT>

<http://localhost:9000/rest/agency/byname?name=LA BANDERA>

<http://localhost:9000/rest/agency/add>

```
{
  "id": 3,
  "nameAgency": "DESAMPARADOS",
  "address": "VIA PRINCIPAL",
  "phoneNumber": "12341234 "
}
```

Consumo desde app PHP

<http://localhost/agency/index.php>

```
<?php
$response = file_get_contents('http://localhost:9000/rest/agency/all');
echo $response;
echo "<br>";
$data = json_decode($response);
echo "<br>";
echo var_dump($data);
echo "<br><br><br>";
echo "<table border='1'>
    <tr>
        <th>ID</th><th>NOMBRE DE AGENCIA</th><th>DIRECCION</th><th>TELEFONO</th></tr>";
foreach ($data as $dato) {
    echo "<tr>";
    echo "<td>".$dato->id."</td>";
    echo "<td>".$dato->nameAgency."</td>";
    echo "<td>".$dato->address."</td>";
    echo "<td>".$dato->phoneNumber."</td>";
    echo "</tr>";
}
echo "</table>";
?>
```

DOC	FECHA	ELABORADO POR	REVISADO POR	APROBADO POR
PRACL-001	15/05/2020	ING. JARRY PALACIOS RIVAS	ING. LUIS DIEGO QUESADA	