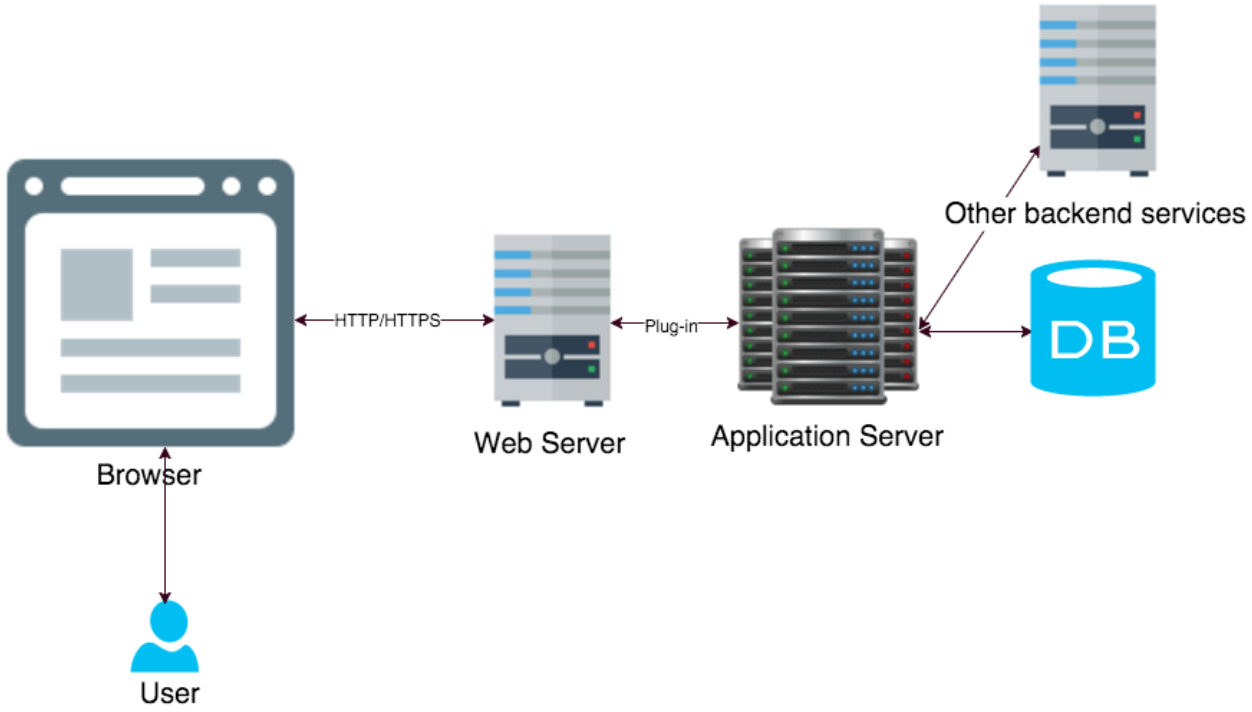
DAW
Despliegue de aplicaciones web
3. Servidores de aplicaciones y de datos

Juan Jesús Tortajada Cordero
jtortajada@florida-uni.es

Florida
Grup Educatiu

Índice

1. Servidores de aplicaciones



El servidor de aplicaciones es un intermediario entre el cliente que accede al servidor web y las bases de datos, así como diferentes servicios y aplicaciones empresariales.

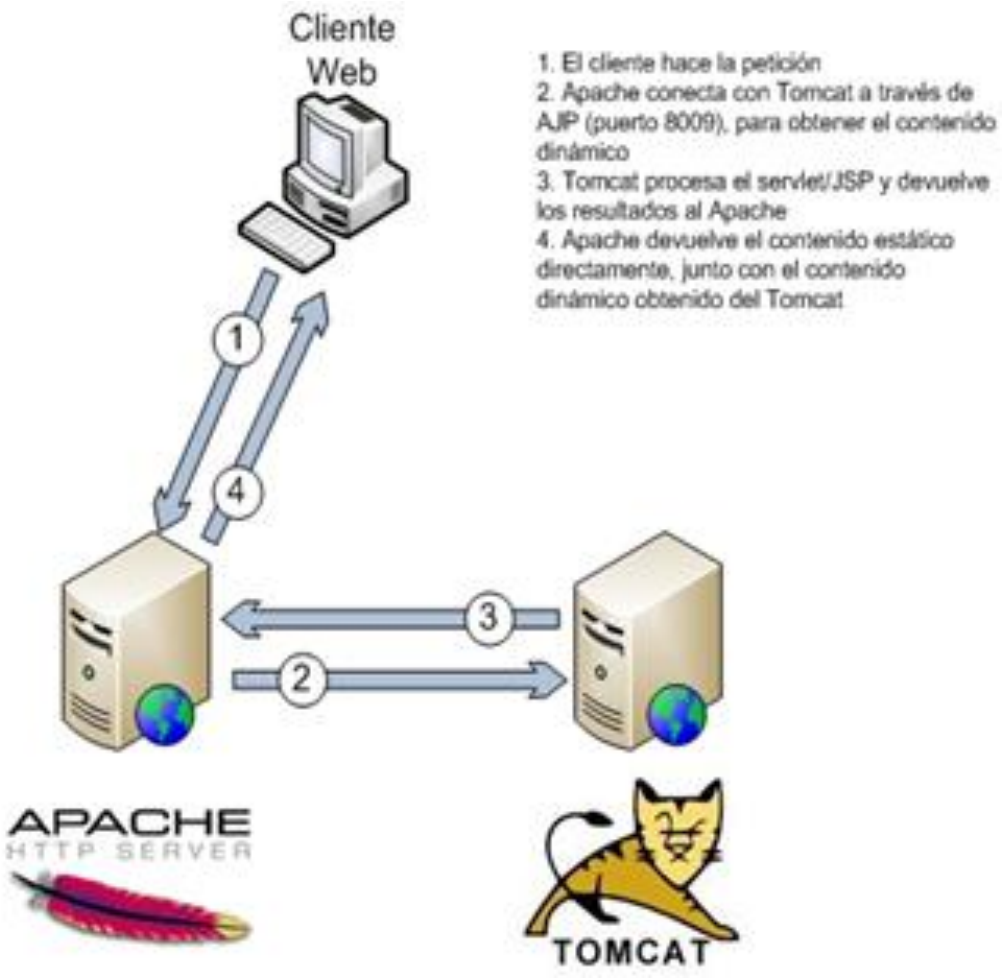
1. Servidores de aplicaciones

Tomcat

Tomcat (Apache Tomcat o Jakarta Tomcat) es un servidor que permite conectar las peticiones HTTP de un servidor web con aplicaciones Java que se ejecuten en el *backend*.

Estas aplicaciones Java se denominan *servlets* y *JSPs* (JavaServer Pages) en el contexto web y serían el equivalente en Java de los scripts PHP.

Estrictamente, Tomcat no es un servidor de aplicaciones, sino un contenedor web de *servlets/JSPs*, aunque en muchos sentidos tiene la misma función.



https://alexcastel.wordpress.com/2020/09/10/conexion-apache-tomcat/

1. Servidores de aplicaciones

Tomcat

Funcionamiento básico:

Desarrollo de una aplicación en Java

➔ compilación como ejecutable (creación de un fichero WAR)

- ✓ Copiar el fichero WAR en la carpeta `/webapps` del servidor
- ✓ El servidor descomprime el fichero WAR y despliegue la aplicación en el puerto indicado (habitualmente 8080).

➔ compilación como ejecutable (creación de un fichero JAR)

NOTA: ejecución de aplicaciones Java

Java requiere disponer de una máquina virtual (JVM) instalada en el sistema operativo para poder ejecutar la aplicación.

Fichero JAR: aplicaciones de propósito general ejecutables por la JVM mediante la instrucción `java -jar miAplicacion.jar`

Fichero WAR: aplicación web preparada para ejecutarse por un servidor Tomcat.

1. Servidores de aplicaciones

Tomcat ➔ Spring Boot: aplicaciones web y microservicios

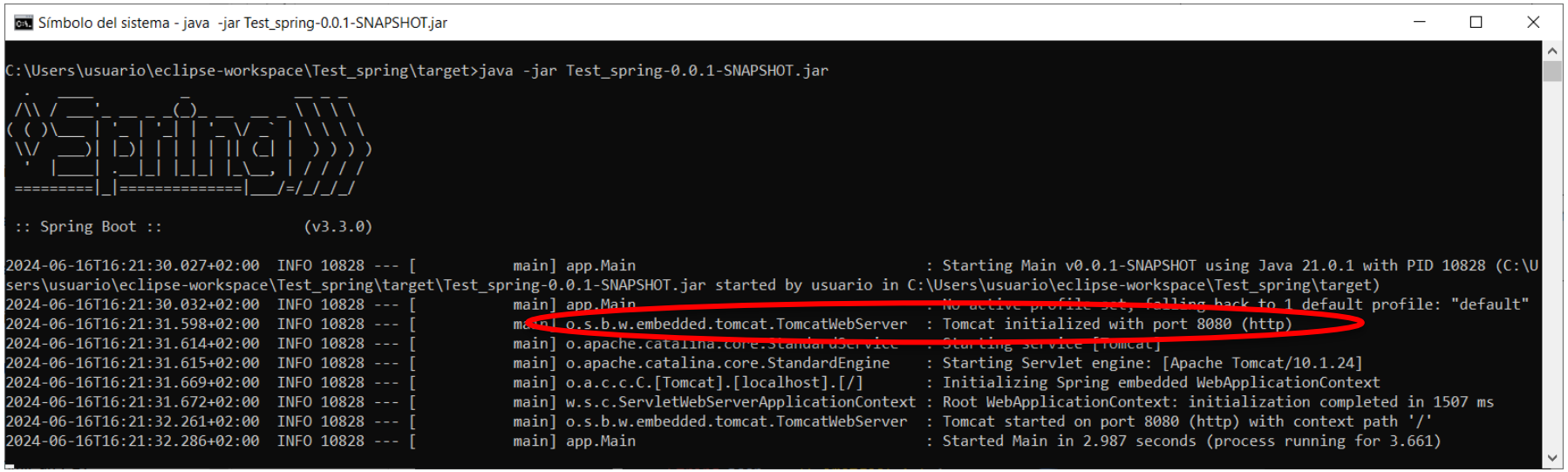
Spring Boot

- ✓ *Framework* para desarrollo en Java de aplicaciones web.
- ✓ Integra por defecto un servidor Tomcat.
- ✓ Múltiples opciones de configuración automatizada.
- ✓ Facilita la gestión e inyección de dependencias (Maven).
- ✓ Permite generar un fichero JAR ejecutable con todo lo necesario para desplegar la aplicación en un servidor (solo requiere que esté instalada la JVM).
- ✓ Permite no incluir el servidor Tomcat y generar un fichero WAR que se ejecute en un servidor Tomcat externo.
- ✓ Optimizado para la generación de microservicios y APIs.

1. Servidores de aplicaciones

Tomcat ➔ Spring Boot: aplicaciones web y microservicios

Instala una versión de Java (≥ 17) si no tienes ninguna instalada (puedes comprobar qué versión tienes instalada ejecutando en el CMD el comando `java -version`).
Descarga el ejecutable `Test_spring-0.0.1-SNAPSHOT.jar` y ejecútalo en un CMD:

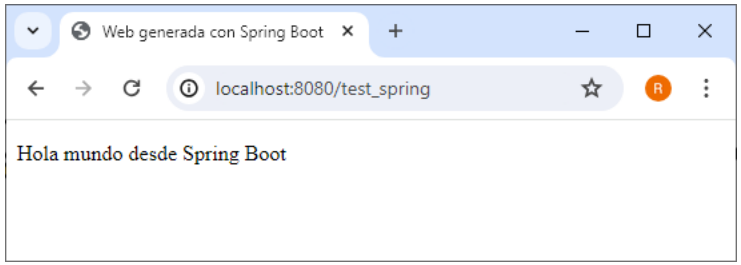


https://www.oracle.com/es/java/technologies/downloads/

1. Servidores de aplicaciones

Tomcat ➔ Spring Boot: aplicaciones web y microservicios

En este caso hemos arrancado la aplicación servidor en local. Podemos acceder desde un cliente (navegador) en la ruta http://localhost:8080/test_spring



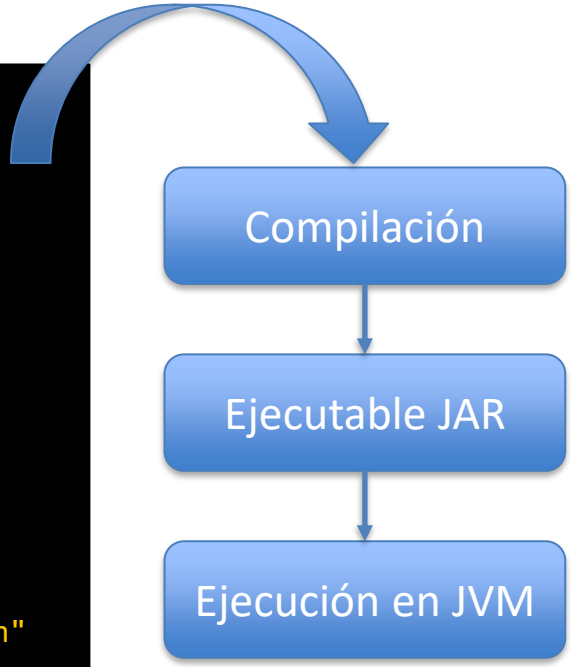
Ahora tenemos la aplicación JAVA `Test_spring-0.0.1-SNAPSHOT.jar`, que se puede desplegar en cualquier máquina y sistema operativo, siempre que disponga de una máquina virtual Java (JVM) instalada.

1. Servidores de aplicaciones

Tomcat ➔ Spring Boot: aplicaciones web y microservicios

Ejemplo: código Java

```
package app.controller;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class Saludo {

	@RequestMapping("/test_spring")
	public ResponseEntity<String> saludo() {
		String html = "<!DOCTYPE html>\r\n"
				+ "<html>\r\n"
				+ "<head>\r\n"
				+ "<title>Web generada con Spring Boot</title>\r\n"
				+ "</head>\r\n"
				+ "<body>\r\n"
				+ "<p>Hola mundo desde Spring Boot</p>\r\n"
				+ "</body>\r\n"
				+ "</html>";
		return ResponseEntity.status(HttpStatus.OK).body(html);
	}
}
```



1. Servidores de aplicaciones

Spring Boot → Ejemplo de proyecto

En VSCode, instala las extensiones "Extension Pack for Java" y "Spring Boot Extension Pack"

En "View" → "Command Palette":

→ Introducir: `Spring Initializr: Create a Maven project...`

→ Seleccionar última versión estable disponible: p.ej. `3.3.3`.

→ Seleccionar lenguaje: `Java`

→ Indicar identificador del grupo (nombre del grupo o namespace al que pertenece la aplicación que vamos a desarrollar): p.ej. `es.florida`

→ Indicar identificador del artefacto (nombre de la aplicación): p.ej. `miapp`

→ Seleccionar el tipo de paquete: `JAR` (el servidor Tomcat irá embebido en la aplicación)

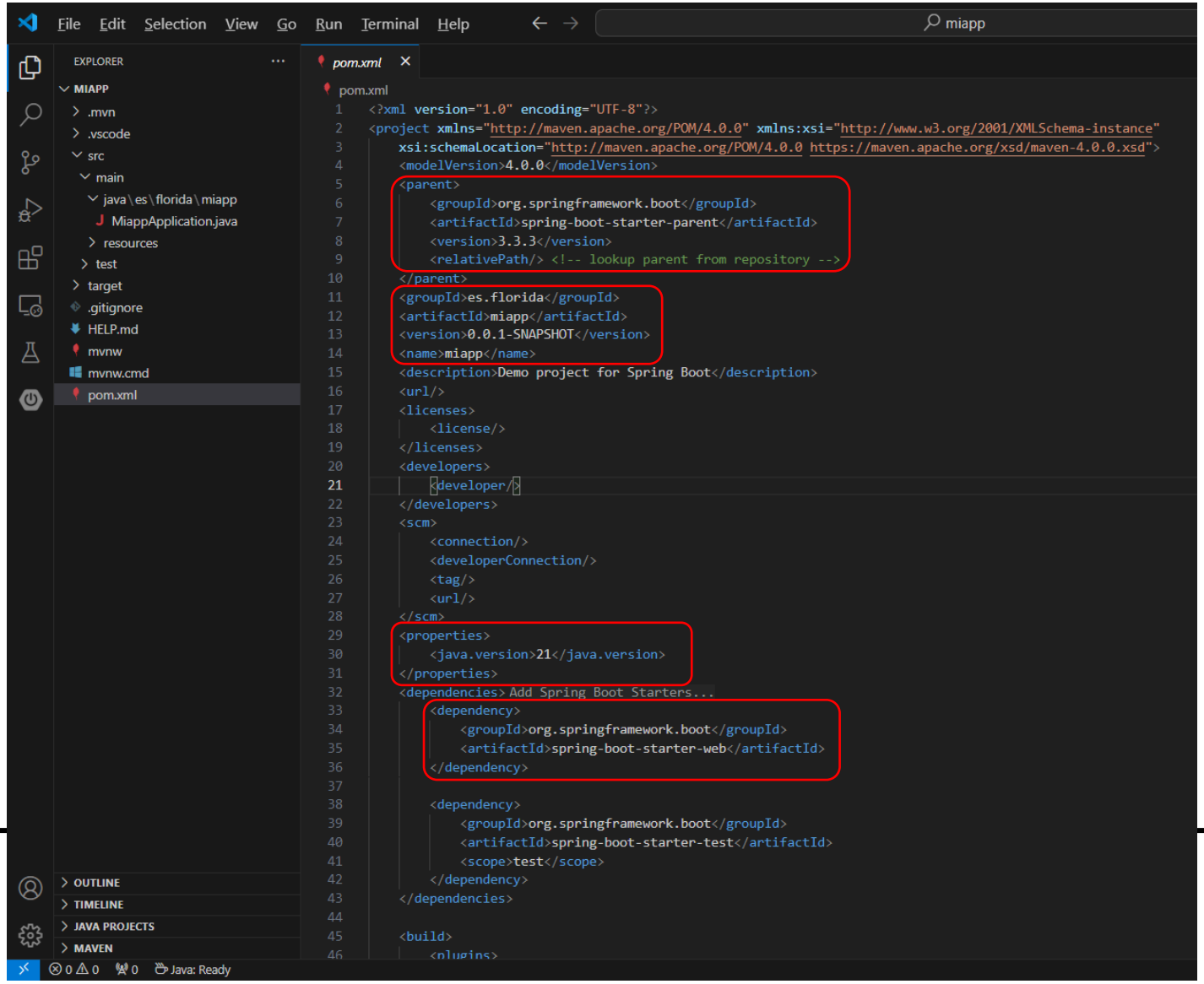
→ Seleccionar versión de Java: p.ej. `21` (última LTS)

→ Seleccionar dependencias: `Spring Web`

→ A continuación, solicita un directorio para crear el proyecto, indicarlo y hacer clic en "Open" para que cree la estructura de directorios necesaria.

1. Servidores de aplicaciones

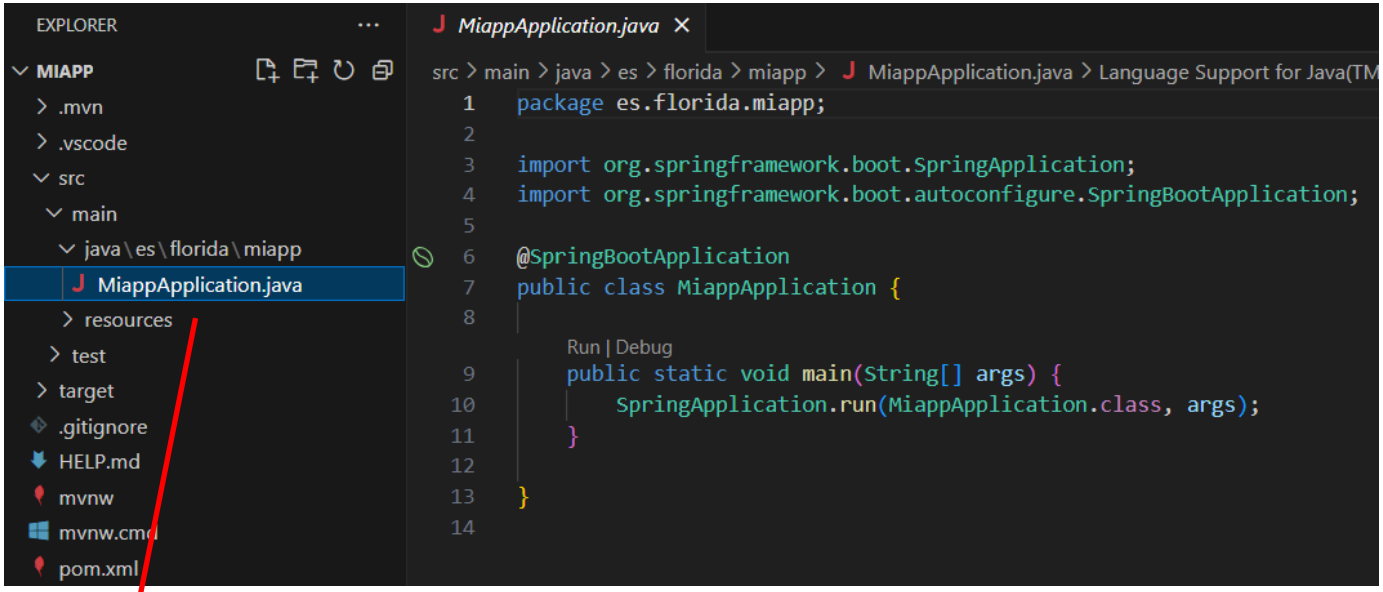
Spring Boot → Ejemplo de proyecto



Florida
Grup Educatiu

1. Servidores de aplicaciones

Spring Boot → Ejemplo de proyecto



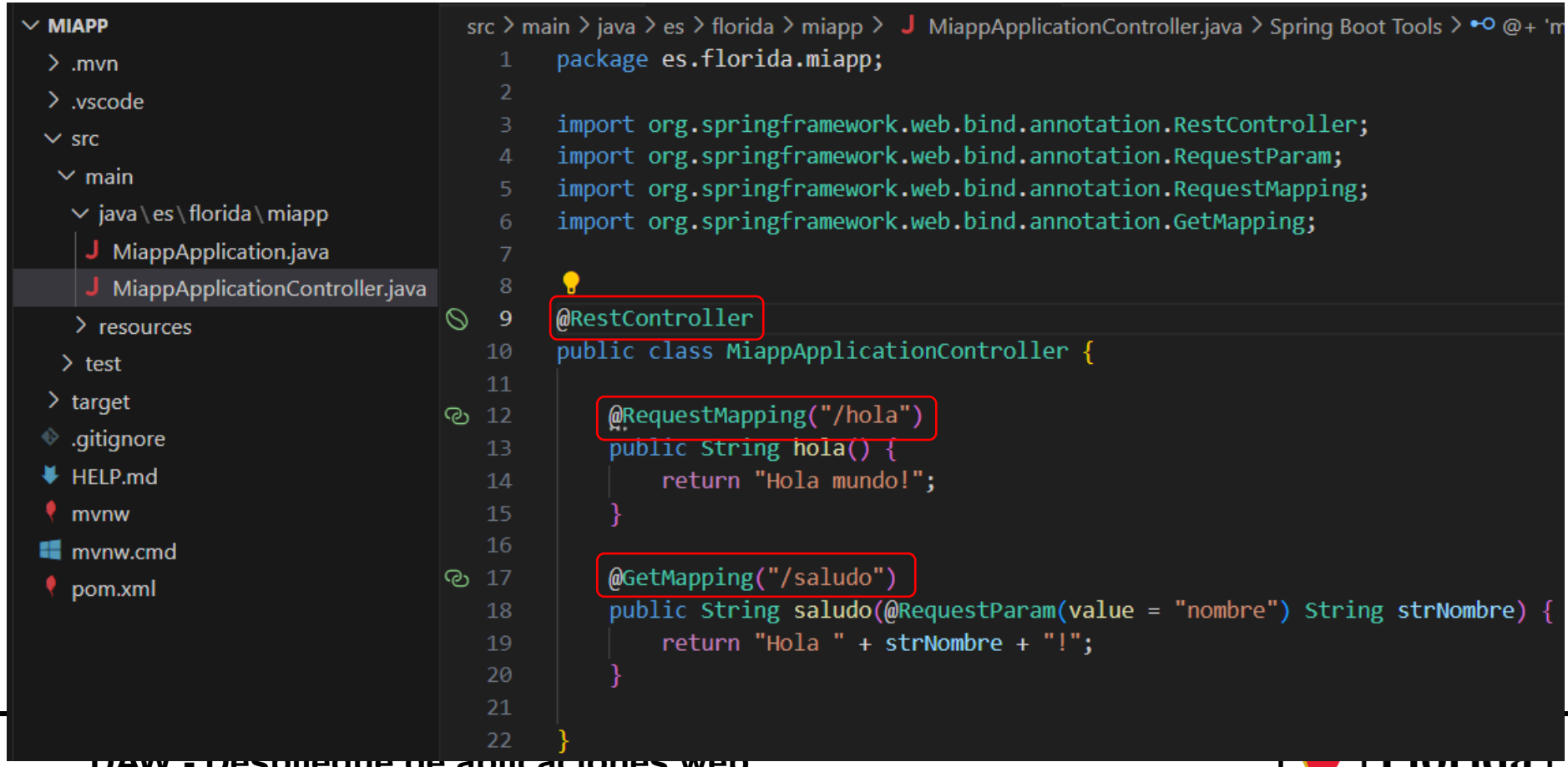
Clic derecho → `Run Java`

```
PS D:\Florida\DAW\2024_2025\Despliegue\spring\miapp>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '@C:\Users\usuario\AppD
ata\Local\Temp\cp_e38wz6vy8doefub35hvx05i3h.argfile' 'es.florida.miapp.MiappApplication'

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.3)

2024-09-15T12:02:42.071+02:00  INFO 3564 --- [miapp] [           main] es.florida.miapp.MiappApplication        : Starting
MiappApplication using Java 21.0.1 with PID 3564 (D:\Florida\DAW\2024_2025\Despliegue\spring\miapp\target\classes started b
y usuario in D:\Florida\DAW\2024_2025\Despliegue\spring\miapp)
2024-09-15T12:02:42.078+02:00  INFO 3564 --- [miapp] [           main] es.florida.miapp.MiappApplication        : No active
 profile set, falling back to 1 default profile: "default"
2024-09-15T12:02:43.954+02:00  INFO 3564 --- [miapp] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat in
itialized with port 8080 (http)
2024-09-15T12:02:43.982+02:00  INFO 3564 --- [miapp] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-09-15T12:02:43.983+02:00  INFO 3564 --- [miapp] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.28]
2024-09-15T12:02:44.091+02:00  INFO 3564 --- [miapp] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-09-15T12:02:44.094+02:00  INFO 3564 --- [miapp] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1912 ms
2024-09-15T12:02:44.859+02:00  INFO 3564 --- [miapp] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
2024-09-15T12:02:44.881+02:00  INFO 3564 --- [miapp] [           main] es.florida.miapp.MiappApplication        : Started MiappApplication in 3.444 seconds (process running for 4.048)
```

1. Servidores de aplicaciones

Spring Boot → Ejemplo de proyecto

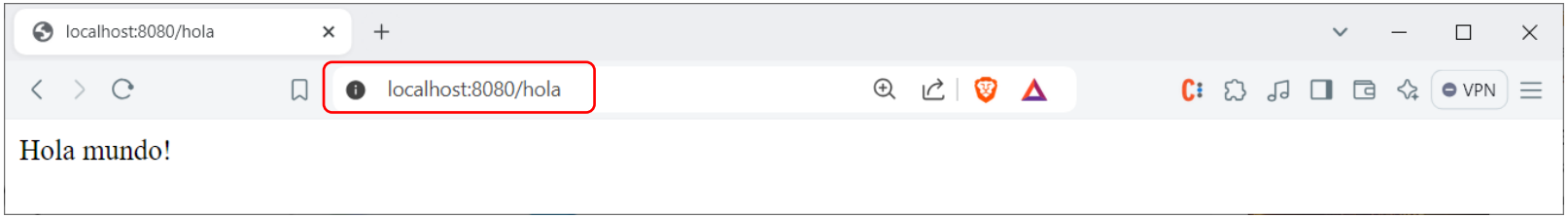
Creamos clase de tipo Controller:

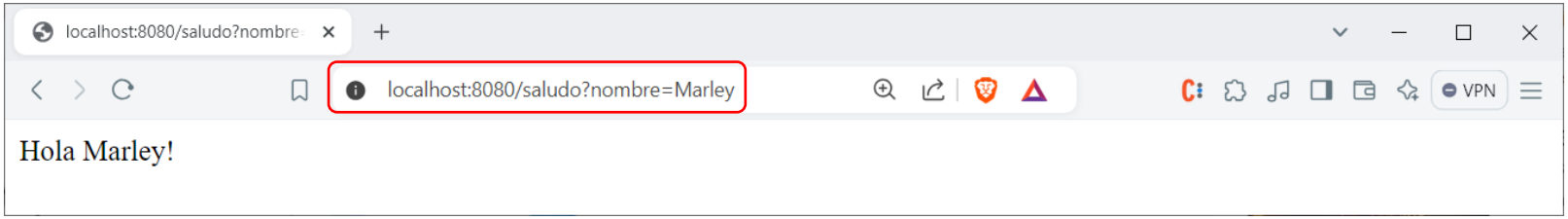


1. Servidores de aplicaciones

Spring Boot → Ejemplo de proyecto

Ejecutamos la aplicación, arranca el servidor y accedemos vía navegador:





1. Servidores de aplicaciones

Spring Boot → Ejemplo de proyecto

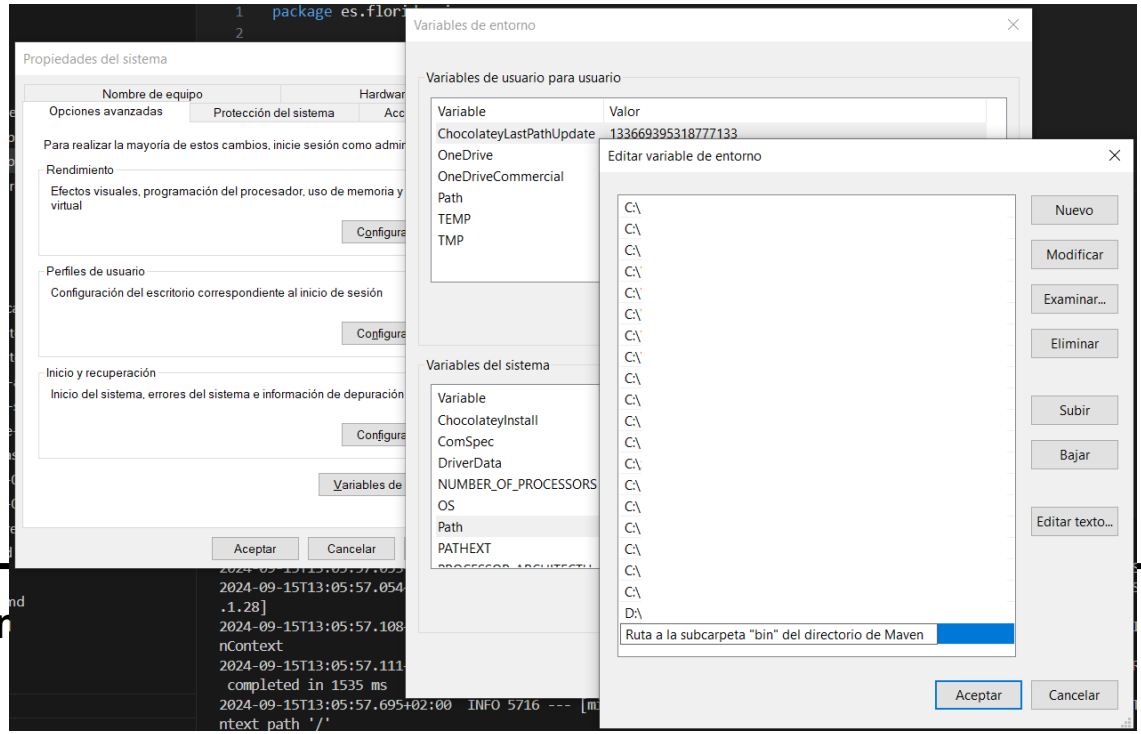
Compilación y creación del ejecutable JAR con Tomcat embebido

Comprueba si tienes instalado Maven desde un terminal: `mvn -v`

En caso de que no esté instalado:

- Descarga el "Binary zip archive" de https://maven.apache.org/download.cgi
- Descomprime el ZIP en algún directorio y añade el directorio "bin" a la variable de entorno Path del sistema
- Reinicia VSCode

Abre un terminal y desde la misma ruta donde esté alojado el fichero pom.xml, ejecuta la instrucción `mvn clean install`

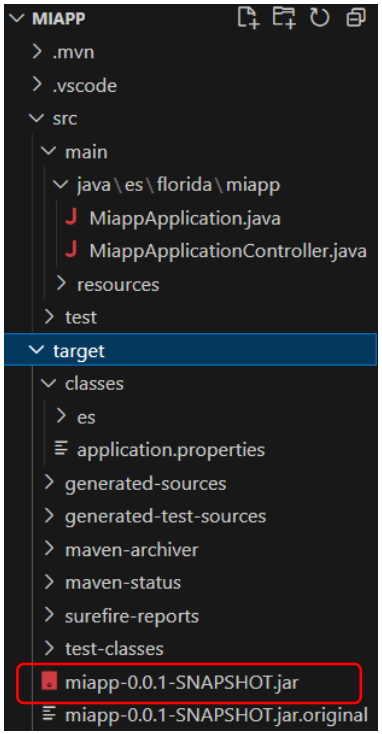


1. Servidores de aplicaciones

Spring Boot → Ejemplo de proyecto

Compilación y creación del ejecutable JAR con Tomcat embebido

El ejecutable se habrá creado en la carpeta "target" del proyecto:



Ahora solo queda levantar el servicio ejecutando la instrucción

```
java -jar nombreDeLaApp.jar
```

Este fichero JAR ya se puede desplegar en cualquier servidor y sistema operativo que tenga instalada una máquina virtual de Java (JVM).

1. Servidores de aplicaciones

Tomcat ➔ Spring Boot: aplicaciones web y microservicios

En la MV con Ubuntu, abrir un Terminal y asegurarse que tiene instalado Java con el comando `java -version`, igual que hemos hecho anteriormente con Windows.

Si no está instalado (no reconoce el comando), instalarlo, mediante el comando `sudo apt install openjdk-17-jre` (versión 17, penúltima versión LTS -long term support-, puedes instalar también alguna más reciente).

Comprobar que está correctamente instalado con `java -version`.

Ahora podemos utilizar el cliente FileZilla para transferir el .JAR desde nuestro sistema Windows a la máquina virtual con Ubuntu, tal y como vimos en el tema 2.

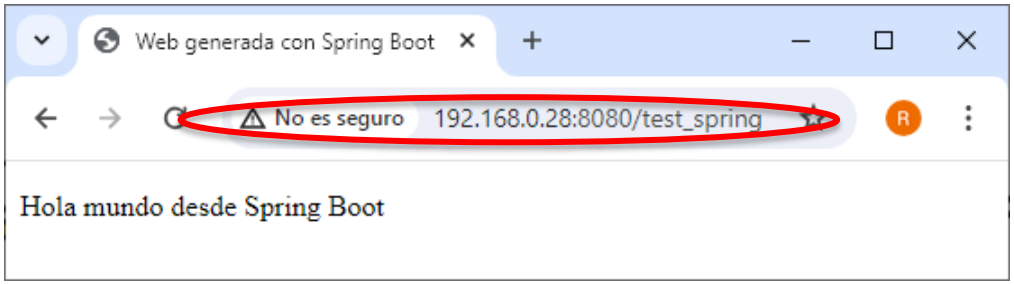
En el Terminal de Ubuntu, ejecutar la aplicación Java: `java -jar Test_spring-0.0.1-SNAPSHOT.jar`

1. Servidores de aplicaciones

Tomcat ➔ Spring Boot: aplicaciones web y microservicios

```
roberto@roberto-VirtualBox:~/Desktop$ java -jar Test_spring-0.0.1-SNAPSHOT.jar
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.0)

2024-06-16T19:46:58.472+02:00  INFO 5842 --- [           main] app.Main                                 : Starting Main
 v0.0.1-SNAPSHOT using Java 17.0.7 with PID 5842 (/home/roberto/Desktop/Test_spring-0.0.1-SNAPSHOT.jar started by rober
to in /home/roberto/Desktop)
2024-06-16T19:46:58.488+02:00  INFO 5842 --- [           main] app.Main                                 : No active pro
file set, falling back to 1 default profile: "default"
2024-06-16T19:47:01.679+02:00  INFO 5842 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initia
lized with port 8080 (http)
2024-06-16T19:47:01.792+02:00  INFO 5842 --- [           main] o.apache.catalina.core.StandardService   : Starting serv
ice [Tomcat]
2024-06-16T19:47:01.799+02:00  INFO 5842 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Serv
let engine: [Apache Tomcat/10.1.24]
2024-06-16T19:47:02.292+02:00  INFO 5842 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing
Spring embedded WebApplicationContext
2024-06-16T19:47:02.301+02:00  INFO 5842 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebAppli
cationContext: initialization completed in 3628 ms
2024-06-16T19:47:03.995+02:00  INFO 5842 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat starte
d on port 8080 (http) with context path '/'
2024-06-16T19:47:04.051+02:00  INFO 5842 --- [           main] app.Main                                 : Started Main
in 7.059 seconds (process running for 8.464)
2024-06-16T19:47:36.062+02:00  INFO 5842 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing
Spring DispatcherServlet 'dispatcherServlet'
2024-06-16T19:47:36.063+02:00  INFO 5842 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing
Servlet 'dispatcherServlet'
2024-06-16T19:47:36.065+02:00  INFO 5842 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed ini
tialization in 1 ms
```

Servidor con Spring Boot y Tomcat ejecutándose en la máquina virtual



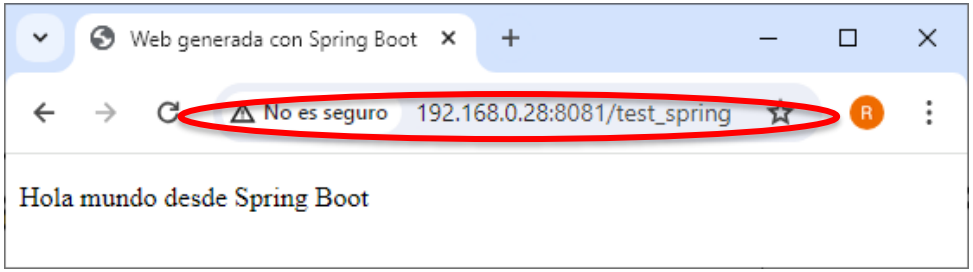
Acceso a la aplicación desde cliente

1. Servidores de aplicaciones

Tomcat ➔ Spring Boot: aplicaciones web y microservicios

```
roberto@roberto-VirtualBox:~/Desktop$ java -jar Test_spring-0.0.1-SNAPSHOT.jar --server.port=8081

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.0)

2024-06-16T22:53:14.372+02:00  INFO 1996 --- [           main] app.Main
  : Starting Main v0.0.1-SNAPSHOT using Java 17.0.7 with PID 1996 (/home/roberto/Desktop/Test_spring-0
.0.1-SNAPSHOT.jar started by roberto in /home/roberto/Desktop)
2024-06-16T22:53:14.393+02:00  INFO 1996 --- [           main] app.Main
  : No active profile set, falling back to 1 default profile: "default"
2024-06-16T22:53:20.351+02:00  INFO 1996 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer
  : Tomcat initialized with port 8081 (http)
2024-06-16T22:53:20.379+02:00  INFO 1996 --- [           main] o.apache.catalina.core.StandardService
  : Starting service [Tomcat]
2024-06-16T22:53:20.379+02:00  INFO 1996 --- [           main] o.apache.catalina.core.StandardEngine
  : Starting Servlet engine: [Apache Tomcat/10.1.24]
2024-06-16T22:53:20.796+02:00  INFO 1996 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]
  : Initializing Spring embedded WebApplicationContext
2024-06-16T22:53:20.798+02:00  INFO 1996 --- [           main] w.s.c.ServletWebServerApplicationContex
t : Root WebApplicationContext: initialization completed in 6246 ms
2024-06-16T22:53:22.806+02:00  INFO 1996 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer
  : Tomcat started on port 8081 (http) with context path '/'
2024-06-16T22:53:22.899+02:00  INFO 1996 --- [           main] app.Main
  : Started Main in 10.034 seconds (process running for 12.202)
```

¿Y si queremos tener varios microservicios en el mismo servidor?
➔ Cada uno funciona de manera independiente, lo único que hay que tener en cuenta es que ocuparán puertos distintos.



Acceso a la aplicación desde cliente. En esta ocasión está alojada en la misma IP, pero en el puerto 8081.

1. Servidores de aplicaciones

Tomcat ➔ Spring Boot: aplicaciones web y microservicios

Sugerencia de ampliación…

En los ejemplos anteriores hemos visto cómo crear un *endpoint* simple para una URL y un *endpoint* que acepta peticiones GET con un parámetro, ahora puedes probar de ampliar el *endpoint* de peticiones GET para que admita varios parámetros. Además, puedes investigar e implementar un *endpoint* para peticiones de tipo POST.

1. Servidores de aplicaciones

Node.js y Express

- ✓ Node.js es un entorno de ejecución de Javascript para servidor.
- ✓ Permite crear aplicaciones web y dispone de un servidor propio.
- ✓ Incluye NPM (*Node Package Manager*), que permite gestionar las dependencias de las aplicaciones.
- ✓ Express es un *framework* que facilita la creación de aplicaciones, APIs, microservicios, etc. sobre Node.js.
- ✓ Javascript no requiere compilar previamente el código, por lo que se puede ejecutar directamente en servidor o en cliente (en realidad se va compilando en tiempo de ejecución).

1. Servidores de aplicaciones

Node.js y Express ➔ Instalación en Windows

Comprobar desde un terminal si ya está instalado Node.js y el gestor de paquetes npm:

```
node -v
npm -v
```

Si no lo está, instalar Node.js (suele incluir el sistema gestor de paquetes npm) desde https://nodejs.org/en/download/prebuilt-installer.

Crea un directorio vacío y sitúate en él. Inicializa un nuevo proyecto: `npm init -y`

Añade Express al proyecto: `npm install express`

Crea un fichero `app.js` vacío en el directorio raíz.

1. Servidores de aplicaciones

Node.js y Express ➔ Instalación en Windows

Fichero app.js

```
const express = require('express');
const app = express();
const port = 3000;
app.get('/test_node', (req, res) => {
  const nombre = req.query.nombre;
  res.send(`Hola ${nombre} desde Node.js/Express`);
});
app.listen(port, () => {
  console.log('Servidor escuchando en puerto 3000');
});
```

1. Servidores de aplicaciones

Node.js y Express ➔ Instalación en Windows

Fichero app.js

```
const express = require('express');
const app = express();
const port = 3000;
app.get('/test_node', (req, res) => {
  const nombre = req.query.nombre;
  res.send(`Hola ${nombre} desde Node.js/Express`);
});
app.listen(port, () => {
  console.log('Servidor escuchando en puerto 3000');
});
```

Importación del framework Express

Inicializa la aplicación

Define el puerto de escucha del servidor (puedes utilizar cualquiera libre)

Definir una ruta HTTP de tipo GET

Recoge el parámetro "nombre" de la petición GET

Respuesta que devuelve el servidor

Inicia el servidor y realiza la acción de imprimir por consola un mensaje

NOTA: la instrucción **`res.send(`Hola ${nombre} desde Node.js/Express`);`**
es equivalente a **`res.send('Hola ' + nombre + ' desde Node.js/Express');`**

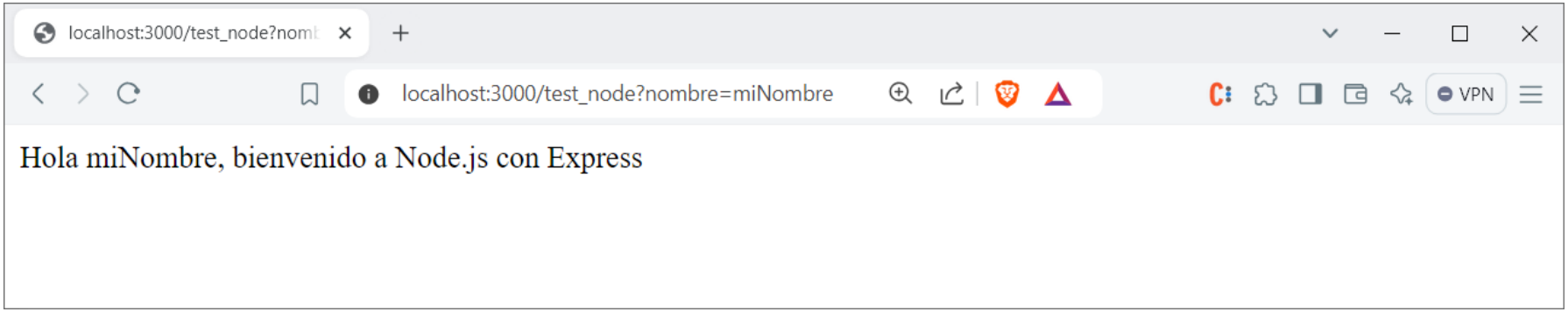
1. Servidores de aplicaciones

Node.js y Express ➔ Instalación en Windows

Levantar el servidor: `node app.js`

```
PS D:\Florida\DAW\2024_2025\Despliegue\nodejs> node app.js
Servidor escuchando en puerto 3000
```

Y acceder vía navegador: http://localhost:3000/test_node?nombre=niNombre



1. Servidores de aplicaciones

Node.js y Express ➔ Instalación en Windows

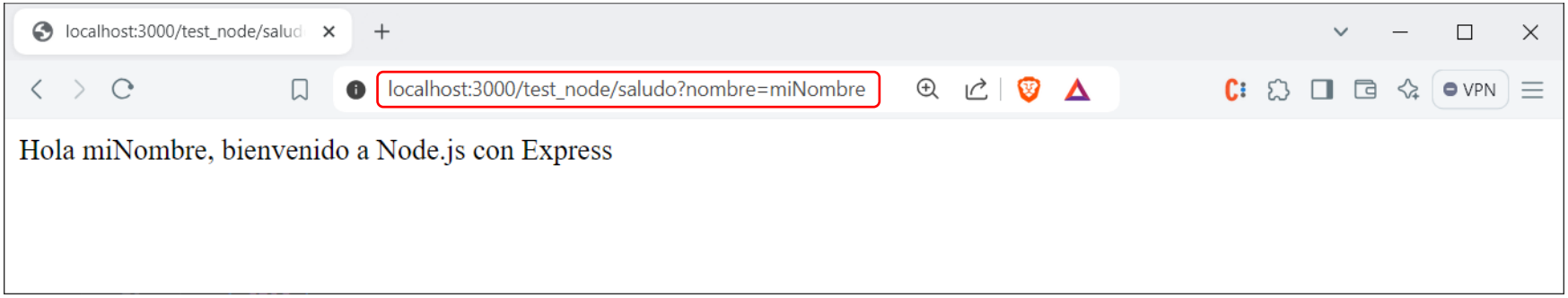
Fichero app.js con varias rutas para un mismo puerto:

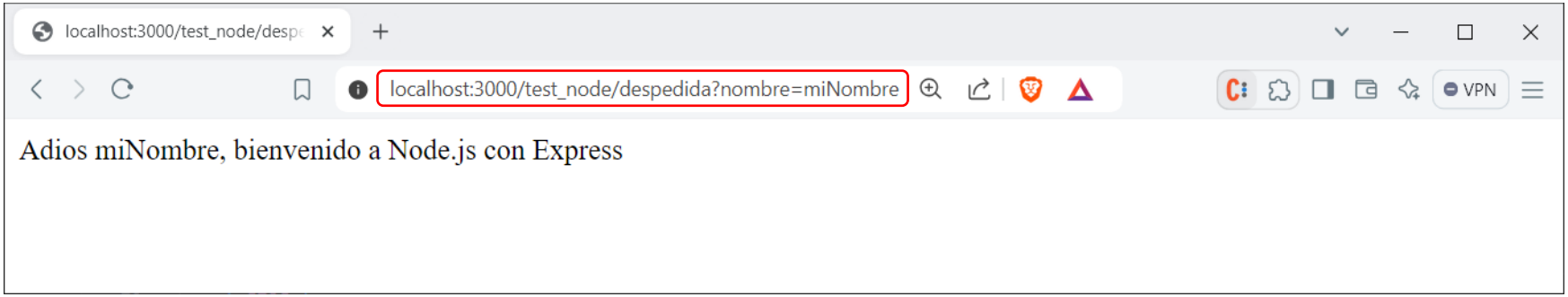
```
const express = require('express');
const app = express();
const port = 3000;
app.get('/test_node/saludo', (req, res) => {
  const nombre = req.query.nombre;
  res.send(`Hola ${nombre} desde Node.js/Express`);
});
app.get('/test_node/despedida', (req, res) => {
  const nombre = req.query.nombre;
  res.send(`Adios ${nombre} desde Node.js/Express`);
});
app.listen(port, () => {
  console.log('Servidor escuchando en puerto 3000');
});
```

1. Servidores de aplicaciones

Node.js y Express ➔ Instalación en Windows

Fichero `app.js` con varias rutas para un mismo puerto:





1. Servidores de aplicaciones

Node.js y Express ➔ Instalación en Windows

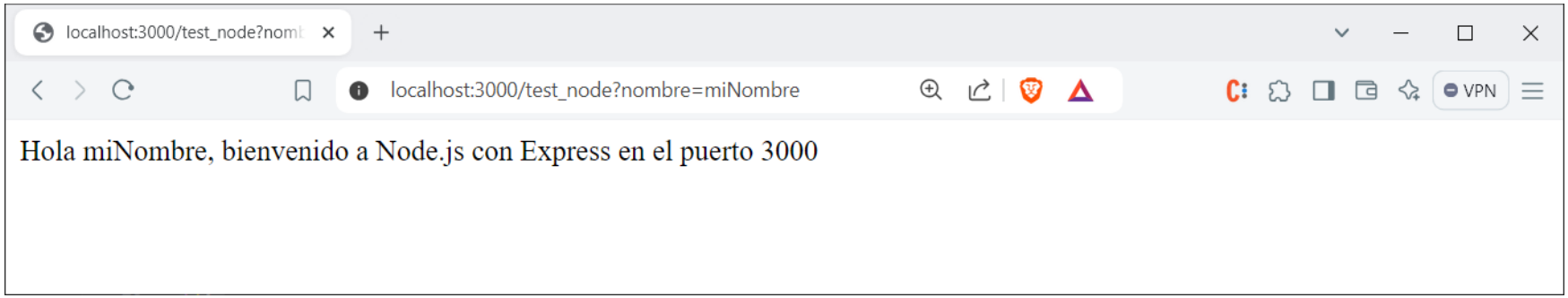
Fichero `app.js` con varios servidores en puertos distintos (en este ejemplo, la ruta es la misma, pero atendiendo en puertos distintos, por lo que no hay conflicto):

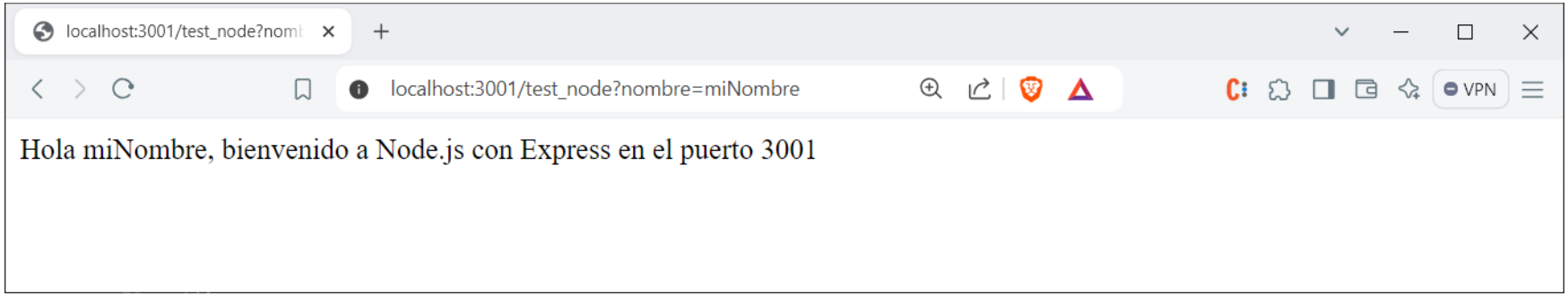
```
const express = require('express');
const app1 = express();
const app2 = express();
const port1 = 3000;
const port2 = 3001;
app1.get('/test_node', (req, res) => {
    const nombre = req.query.nombre;
    res.send(`Hola ${nombre}, bienvenido a Node.js con Express en el puerto ${port1}`);
});
app2.get('/test_node', (req, res) => {
    const nombre = req.query.nombre;
    res.send(`Hola ${nombre}, bienvenido a Node.js con Express en el puerto ${port2}`);
});
app1.listen(port1, () => {
    console.log(`Servidor 1 escuchando en puerto ${port1}`);
});
app2.listen(port2, () => {
    console.log(`Servidor 2 escuchando en puerto ${port2}`);
});
```

1. Servidores de aplicaciones

Node.js y Express ➔ Instalación en Windows

Fichero `app.js` con varios servidores en puertos distintos:





NOTA: en este ejemplo, la ruta es la misma, pero atendiendo en puertos distintos, por lo que no hay conflicto. Puedes probar de intentar arrancar el segundo servidor en el mismo puerto que el servidor 1 (`app2.listen(port2() => {})`; para ver qué tipo de error te genera.

1. Servidores de aplicaciones

Node.js y Express ➔ Instalación en Ubuntu 18.04

Desde el terminal, descargar e instalar nvm (*Node Version Manager*):

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh
| bash
```

Cargar nvm en el terminal:

```
export NVM_DIR="$([ -z "${XDG_CONFIG_HOME-}" ] && printf %s
"${HOME}/.nvm" || printf %s "${XDG_CONFIG_HOME}/nvm")"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
```

Instalar Node.js (versión 12.22.12, compatible con Ubuntu 18.04):

```
nvm install 12.22.12
nvm use 12.22.12
```

Verificar la instalación:

```
node -v
npm -v
```

1. Servidores de aplicaciones

Node.js y Express ➔ Instalación en Ubuntu 18.04

Crear un directorio para el proyecto (p.ej. `test_node`) y acceder a él:

```
mkdir test_node
cd test_node
```

Inicializar nuevo proyecto Node.js utilizando npm (*Node Package Manager*):

```
npm init -y
```

Instalar Express:

```
npm install express
```

Crear el archivo del servidor:

```
touch app.js
nano app.js
```

(puedes utilizar otro editor de texto si lo prefieres)

1. Servidores de aplicaciones

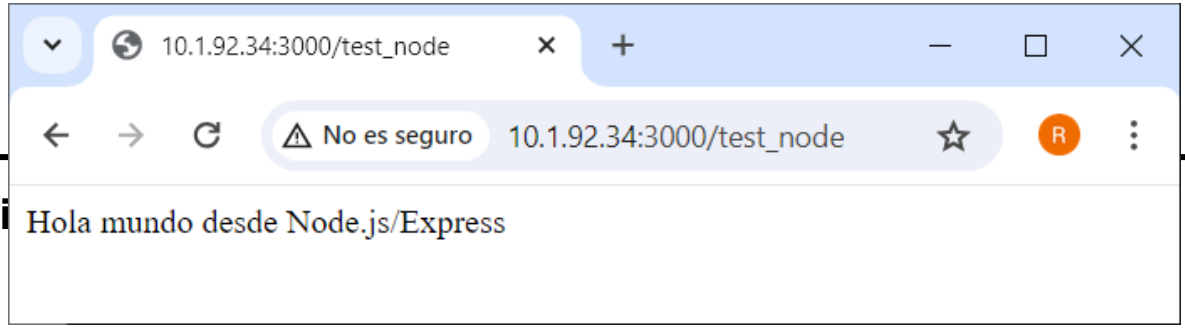
Node.js y Express ➔ Instalación en Ubuntu 18.04

Editar el código correspondiente en "app.js":

```
const express = require('express');
const app = express();
const port = 3000;
app.get('/test_node', (req, res) => {
  res.send('Hola mundo desde Node.js/Express');
});
app.listen(port, () => {
  console.log('Servidor escuchando en puerto 3000');
});
```

Iniciar el servidor:
node app.js

En este caso, la MV tiene asignada la IP 10.1.92:34, por lo que para acceder desde el cliente (navegador):



1. Servidores de aplicaciones

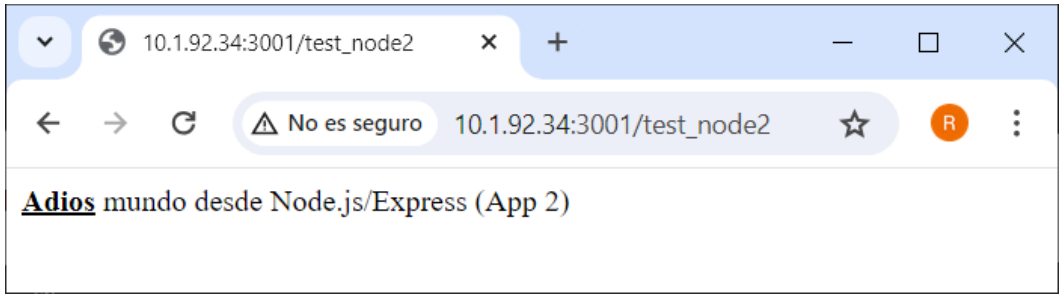
Node.js y Express ➔ Instalación en Ubuntu 18.04

Igual que antes, se pueden configurar la aplicación en otro puerto, simultanear varias aplicaciones en puertos distintos, añadir *endpoints* a cada aplicación, etc.

```
const express = require('express');
const app2 = express();
const port = 3001;
app2.get('/test_node2', (req, res) => {
  res.send('<b><u>Adios</u></b> mundo desde Node.js/Express');
});
app2.listen(port, () => {
  console.log('Servidor escuchando en puerto 3001');
});
```

Nuevo servidor:

```
export NVM_DIR="$HOME/.nvm"
nvm use 12.22.12
node app2.js
```



1. Servidores de aplicaciones

Node.js y Express

Sugerencia de ampliación…

En los ejemplos anteriores hemos visto cómo crear un *endpoint* simple para una URL y un *endpoint* que acepta peticiones GET con un parámetro, ahora puedes probar de ampliar el *endpoint* de peticiones GET para que admita varios parámetros. Además, puedes investigar e implementar un *endpoint* para peticiones de tipo POST.

1. Servidores de aplicaciones

Node.js y Express - Despliegue en servidores

Para desplegar la aplicación, primero el servidor necesita tener instalado Node.js.

- En el caso de servidor Linux:

```
# Instalar NVM (Node Version Manager)
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
# Cargar NVM en la sesión actual (puedes agregar esta línea a tu .bashrc o .zshrc)
source ~/.bashrc
# Instalar la última versión LTS de Node.js
nvm install --lts
```

- En el caso de servidor Windows: https://nodejs.org/en/download/prebuilt-installer.

1. Servidores de aplicaciones

Node.js y Express - Despliegue en servidores

La estructura habitual de una aplicación completa es similar a esta:

```
/miApp
|
|---/node_modules        # Carpeta de dependencias
|---/public              # Archivos estáticos
|---/views               # Vistas o plantillas HTML
|---/config              # Configuraciones
|---.gitignore           # Archivo .gitignore
|---app.js               # Archivo principal del servidor
|---package.json         # Descripción y dependencias del proyecto
|---package-lock.json    # Archivo de versiones de dependencias (generado por npm)
|---README.md            # Documentación del proyecto
```

Hay que transferir al servidor el código fuente y sus dependencias, pero en vez de transferir toda la carpeta del proyecto, prescindiremos de la subcarpeta `node_modules`, donde hay muchas bibliotecas de código que no son necesarias para la versión de producción de nuestra aplicación. De hecho, es habitual que esta carpeta esté incluida en el fichero `.gitignore`.

1. Servidores de aplicaciones

Node.js y Express - Despliegue en servidores

Ajustes en el fichero `package.json`:

```
{
  "name": "nodejs",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "express": "^4.21.0"
  }
}
```

Esta línea de script permite que la aplicación se pueda ejecutar desde consola con el comando `npm start`.
Antes había que utilizar la instrucción node seguida del nombre del fichero js. Con esta nueva instrucción (`npm start`) el despliegue se simplifica, porque siempre se utiliza la misma instrucción "estándar".

```
PS D:\Florida\DAW\2024_2025\Despliegue\nodejs> node app.js
Servidor escuchando en puerto 3000
PS D:\Florida\DAW\2024_2025\Despliegue\nodejs> npm start

> nodejs@1.0.0 start
> node app.js

Servidor escuchando en puerto 3000
```

1. Servidores de aplicaciones

Node.js y Express - Despliegue en servidores

Transferir los ficheros y carpetas (excluyendo `node_modules` y cualquier otra carpeta/fichero no necesario para producción) al servidor vía SFTP o similar.

En el servidor, situarse en la carpeta correspondiente donde estén los ficheros e instalar la aplicación: `npm install --production`

Finalmente, ejecutarla: `npm start`

2. Servidores de bases de datos

En el contexto del despliegue de aplicaciones web nos centramos en los servidores de aplicaciones, pero en el *stack* de una aplicación web tiene también un papel fundamental el servidor de bases de datos.

Algunos servidores de bases de datos muy utilizados:

MySQL / MariaDB: relacional (SQL) ➔ paquetes LAMP / XAMPP

PostgreSQL: relacional (SQL)

MongoDB: no relacional ➔ MongoDB Community Server

2. Servidores de bases de datos

Recordar que por un lado está el servidor de base de datos (servicio), que suele ejecutarse en segundo plano (sin interfaz gráfica), por lo que es habitual disponer de una aplicación que permita administrar la base de datos:

Algunas aplicaciones de gestión de bases de datos muy utilizadas:

phpMyAdmin, MySQL Workbench, HeidiSQL, DBeaver

pgAdmin, DBeaver, HeidiSQL

MongoDB Compass

2. Servidores de bases de datos

De manera análoga a los servidores web y de aplicaciones que hemos visto en apartados anteriores, un servicio de base de datos se identifica mediante una IP (protocolo IP) y un puerto (protocolo TCP).

Además, las bases de datos deben estar convenientemente protegidas, por lo que para acceder a su contenido será necesario autenticarse con usuario y contraseña.

Así, para conectarse al servicio de bases de datos desde una aplicación se necesita lo siguiente:

- IP
- Puerto
- Usuario
- Contraseña

NOTA: cuando se instala un servidor de bases de datos, la conexión está permitida desde un usuario (p.ej. root) que no tiene contraseña. Es importante modificar esta configuración por defecto, utilizando contraseñas fuertes y creando usuarios/contraseñas específicos para cada aplicación.

2. Servidores de bases de datos

Ejemplo: administración de servidor MySQL (MV Ubuntu) desde phpMyAdmin

Recordemos que en el tema 1 habíamos instalado la pila LAMP en la MV con Ubuntu:

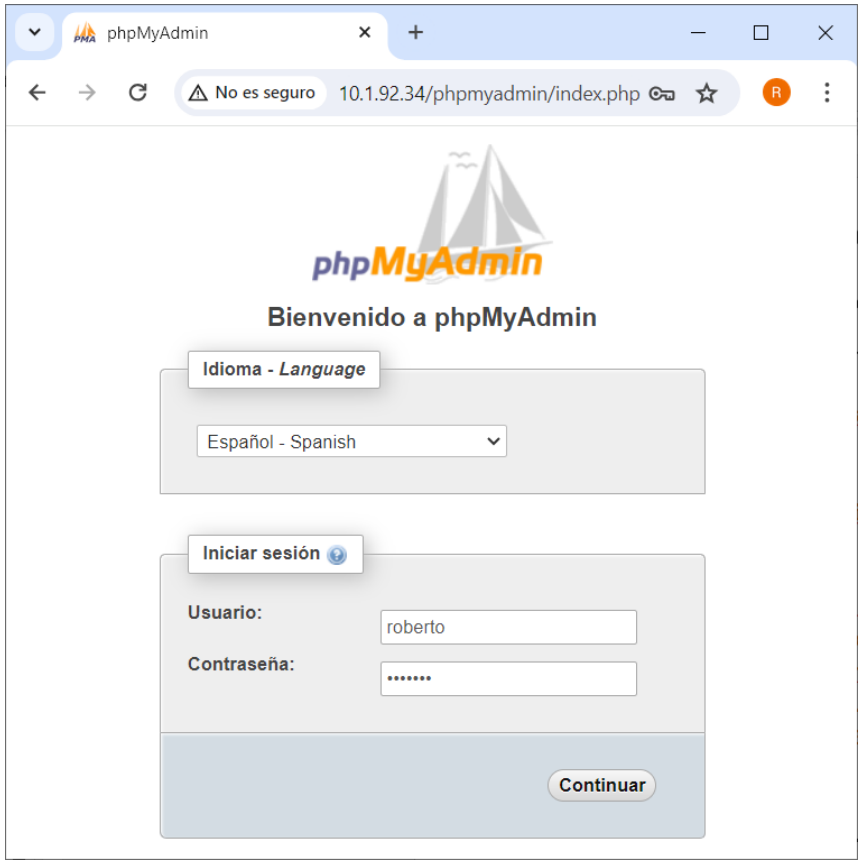
1. Instalar MySQL (MariaDB): `sudo apt install mariadb-server mariadb-client -y`
2. Configurar seguridad MariaDB: `sudo mysql_secure_installation`
3. Instalar PHP: `sudo apt install php libapache2-mod-php php-mysql -y`
4. Instalar phpMyAdmin: `sudo apt install phpmyadmin`
5. Configurar Apache para utilizar phpMyAdmin: `sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/sites-enabled/phpmyadmin.conf`
6. Reiniciar Apache: `sudo systemctl restart apache2`
7. Crear un usuario para MariaDB: `sudo mysql`

```
MariaDB> CREATE USER 'roberto'@'localhost' IDENTIFIED BY 'roberto'
MariaDB> GRANT ALL PRIVILEGES ON *.* TO 'roberto'@'localhost'
MariaDB> FLUSH PRIVILEGES
MariaDB> exit
```

8. Reiniciar el servidor MariaDB: `sudo systemctl restart mysqld`

2. Servidores de bases de datos

Ejemplo: administración de servidor MySQL (MV Ubuntu) desde phpMyAdmin

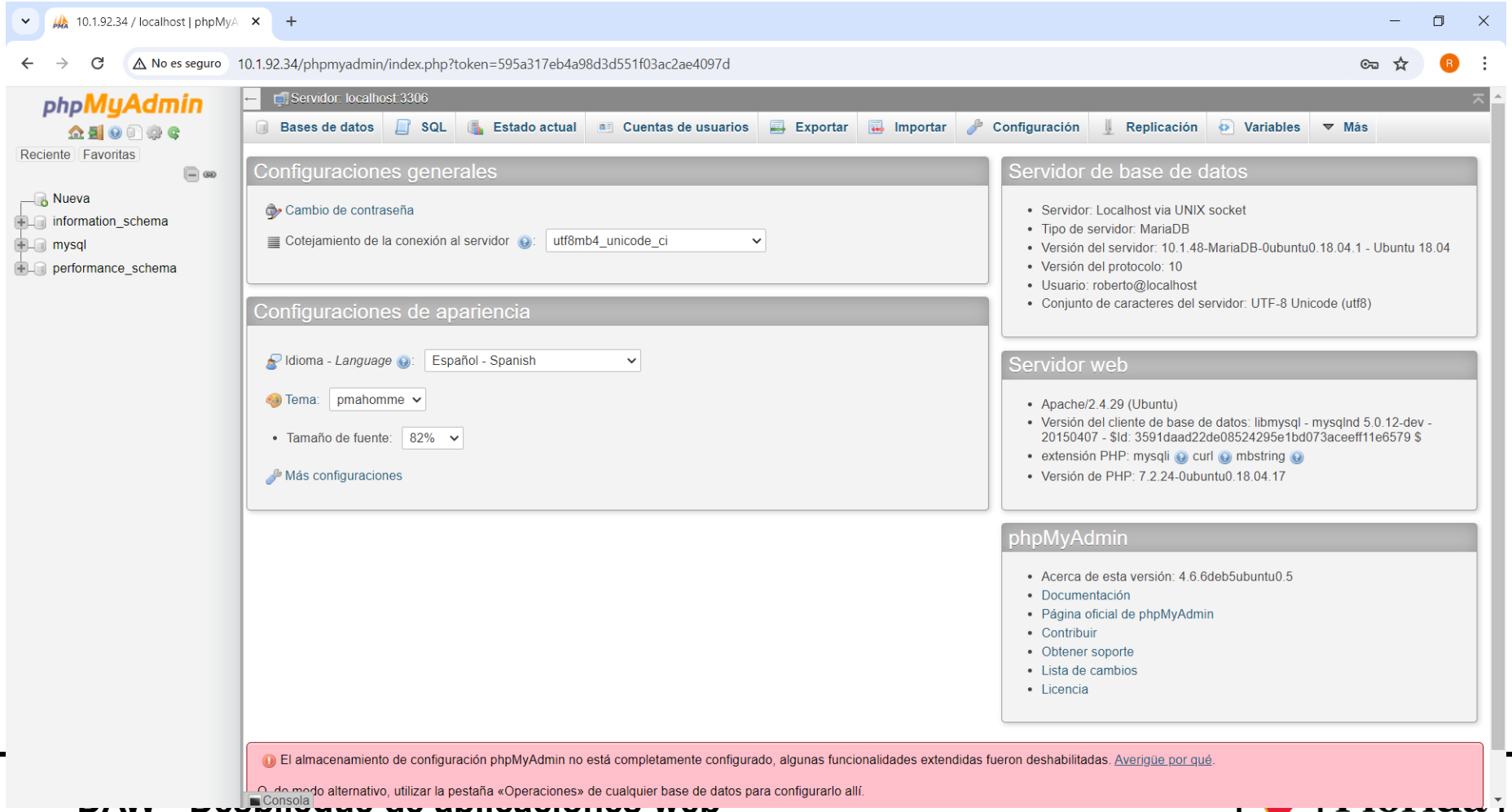


NOTA(1): a diferencia de aplicaciones como MySQL Workbench, HeidiSQL o DBeaver, phpMyAdmin es una aplicación web que corre sobre Apache, por lo que la instalamos en el lado del servidor (las otras habría que instalarlas en el lado del cliente). Por tanto, el acceso a phpMyAdmin es a través del navegador.

NOTA(2): en algunas instalaciones (p.ej. XAMPP), se permite el acceso por defecto sin contraseña (usuario 'root'), pero en otras (la que estamos viendo en Ubuntu), no permite un acceso sin contraseña por defecto. Es altamente aconsejable crear siempre un usuario y una contraseña (fuerte) para cada aplicación, especificando solo los permisos que sean necesarios.

2. Servidores de bases de datos

Ejemplo: administración de servidor MySQL (MV Ubuntu) desde phpMyAdmin



2. Servidores de bases de datos

Ejemplo: administración de servidor MySQL (MV Ubuntu) desde phpMyAdmin

Con phpMyAdmin u otras aplicaciones de administración de bases de datos ya podemos realizar operaciones de CRUD sobre la base de datos de manera remota.

Por ejemplo, vamos a desplegar la base de datos de ejemplo world database en el servidor MySQL que hemos configurado en nuestra máquina Ubuntu.

1. Descargar en local el fichero .zip correspondiente a "*world database*".
2. Extraer el fichero `world.sql`.
3. Importar el fichero .sql en phpMyAdmin.
4. Si lo has hecho correctamente, se habrá importado una base de datos llamada "*world*" que contiene tres tablas: "*city*", "*country*" y "*countrylanguage*".

SUGERENCIA: puedes probar a importar otras bases de datos de ejemplo y también explorar las diferentes opciones de exportación de bases de datos.

2. Servidores de bases de datos

Ejemplo: administración de MongoDB (MV Ubuntu) desde MongoDB Compass

Procedemos ahora a instalar en Ubuntu el servidor MongoDB Community Server.
NOTA: por compatibilidad con Ubuntu 18.04, instalaremos la versión 4.4, pero puedes probar otras versiones más recientes si tienes una versión de Ubuntu superior.

1. Desde el terminal, importamos la clave GPG para la instalación: `wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -`
2. Crear el archivo de lista para MongoDB v4.4: `echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list`
3. Actualizar el índice de paquetes locales: `sudo apt-get update`
4. Instalar MongoDB: `sudo apt-get install -y mongodb-org`
5. Iniciar el servicio MongoDB: `sudo systemctl start mongod`
6. Habilitar MongoDB para que se inicie con el sistema: `sudo systemctl enable mongod`

2. Servidores de bases de datos

Ejemplo: administración de MongoDB (MV Ubuntu) desde MongoDB Compass

Igual que hemos hecho para MySQL, debemos crear un usuario y una contraseña para poder acceder de manera remota y segura. En este caso, vamos a configurarlo también desde la consola que lleva MongoDB.

Para acceder a la consola, desde el terminal introducir el comando: `mongo`

```
> use admin
> db.createUser({ user: "roberto" , pwd: "roberto", roles:
["userAdminAnyDatabase", "dbAdminAnyDatabase",
"readWriteAnyDatabase"]})
> exit
```

Editar el fichero de configuración: `sudo nano /etc/mongod.conf`

```
net:
  port: 27017
  bindIp: 0.0.0.0
security
  authorization: enabled
```

- Aceptar conexiones de cualquier IP
- Activar autorización

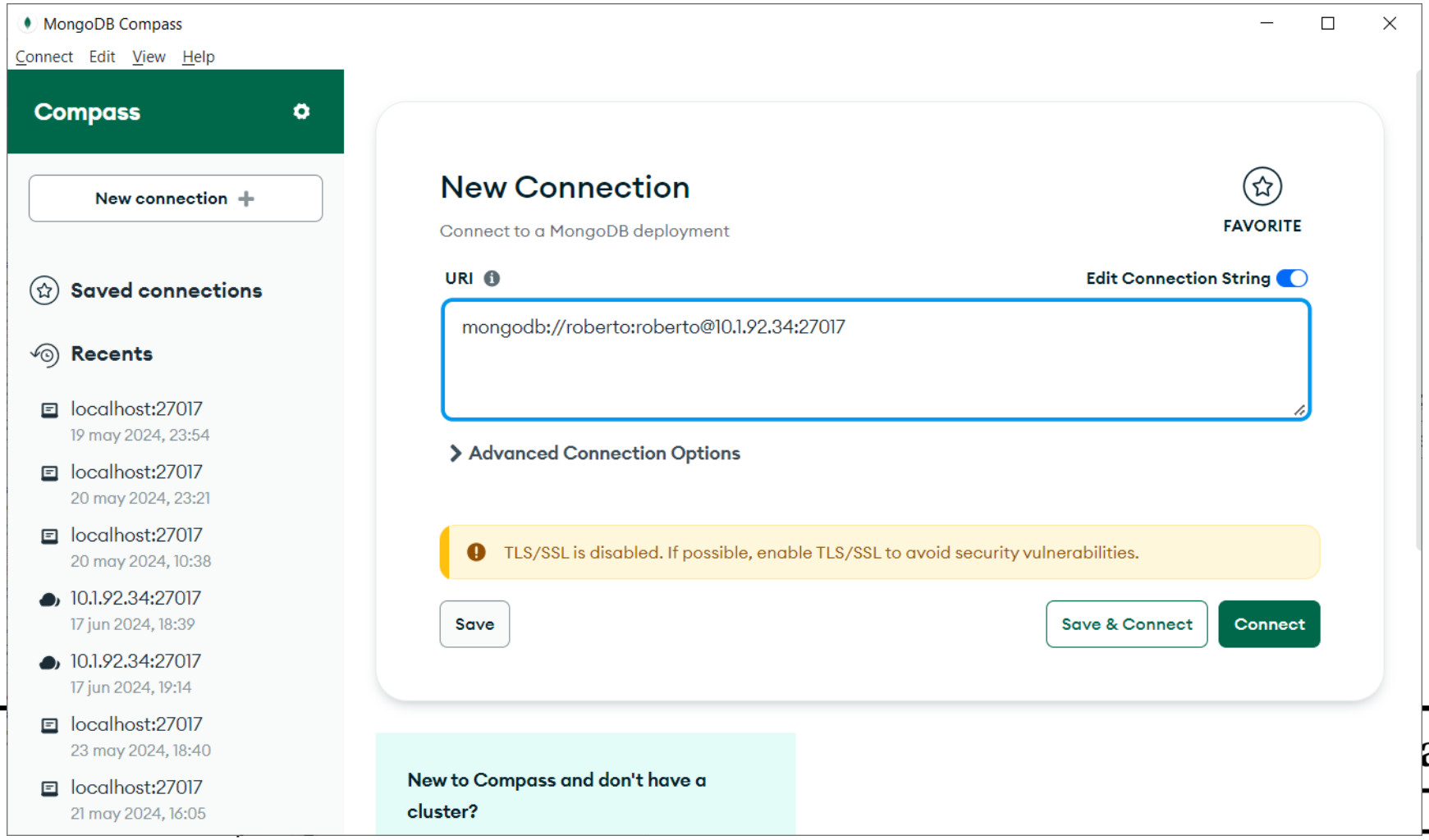
NOTA: atención a los 2 espacios de sangría

Reiniciar el servidor: `sudo systemctl restart mongod`

2. Servidores de bases de datos

Ejemplo: administración de MongoDB (MV Ubuntu) desde MongoDB Compass

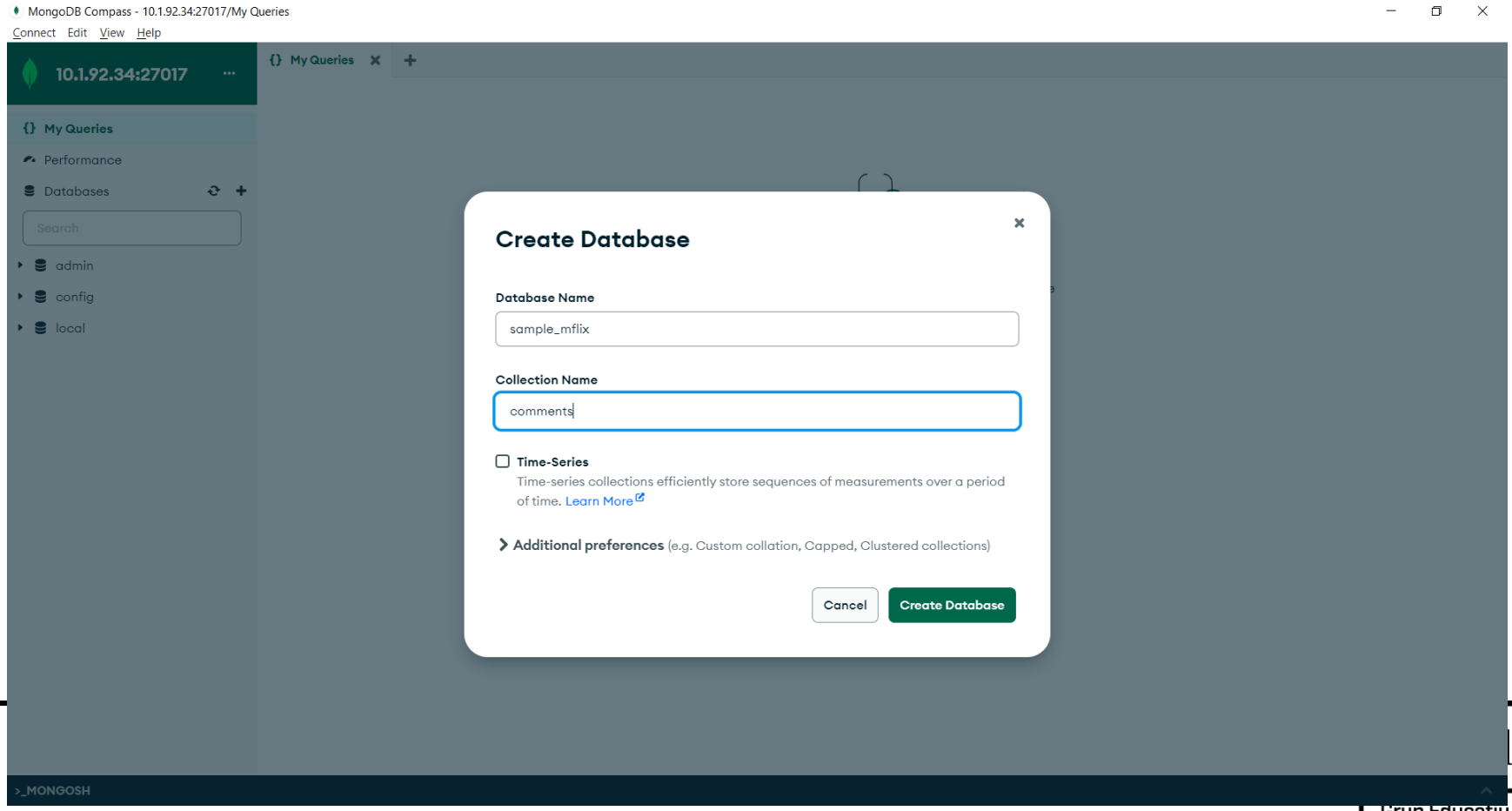
Desde MongoDB Compass (aplicación de escritorio), introducir la cadena de conexión:



2. Servidores de bases de datos

Ejemplo: administración de MongoDB (MV Ubuntu) desde MongoDB Compass

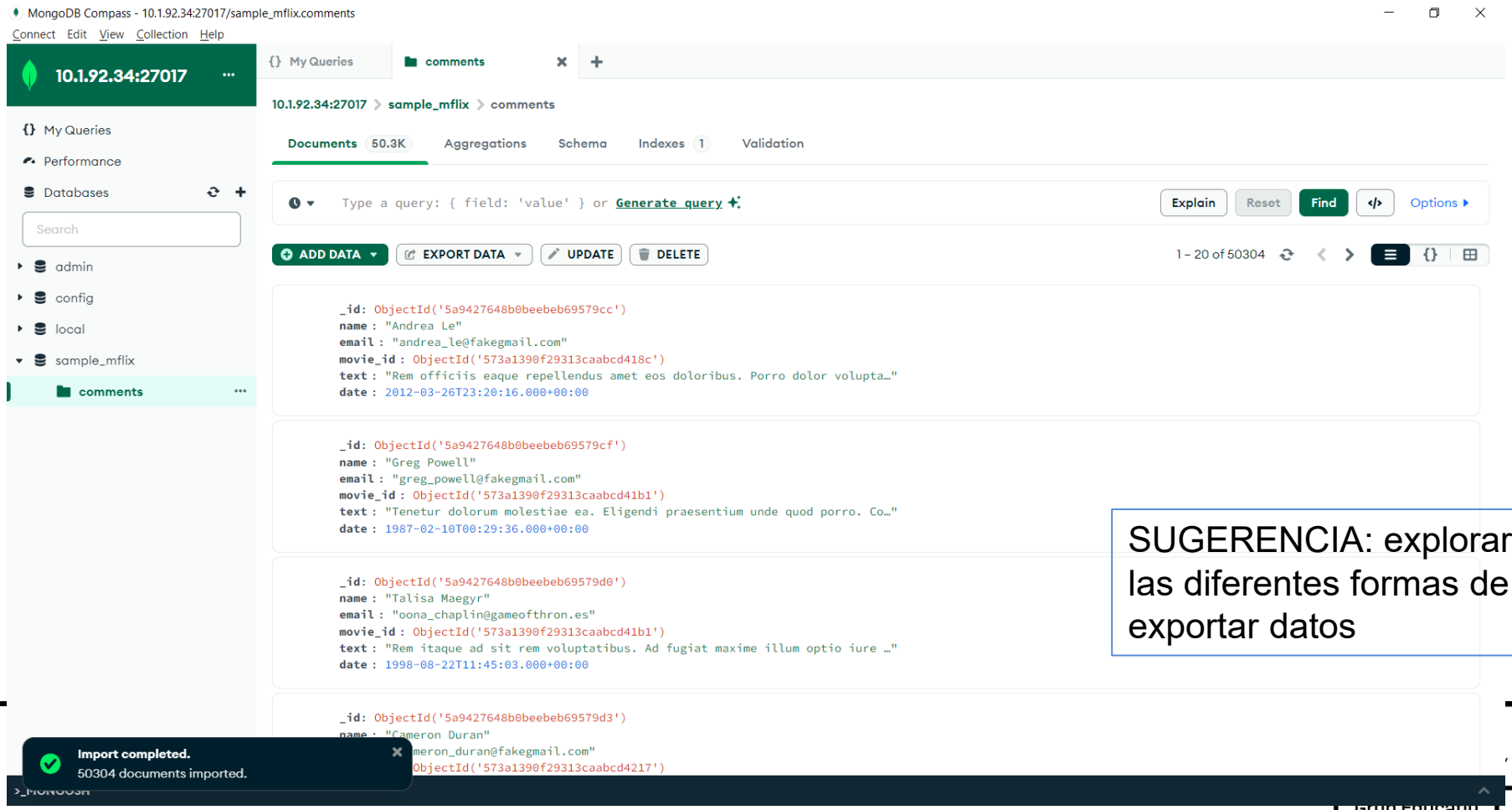
De igual forma que antes, podemos importar colecciones a la base de datos desde MongoDB Compass: primero crear base de datos y colección:



2. Servidores de bases de datos

Ejemplo: administración de MongoDB (MV Ubuntu) desde MongoDB Compass

A continuación, "ADD DATA" e "Import JSON" (en este caso "sample_mflix.comments.json"). Repetir el procedimiento para el resto de los ficheros JSON.



SUGERENCIA: explorar las diferentes formas de exportar datos

2. Servidores de bases de datos

mongoDB

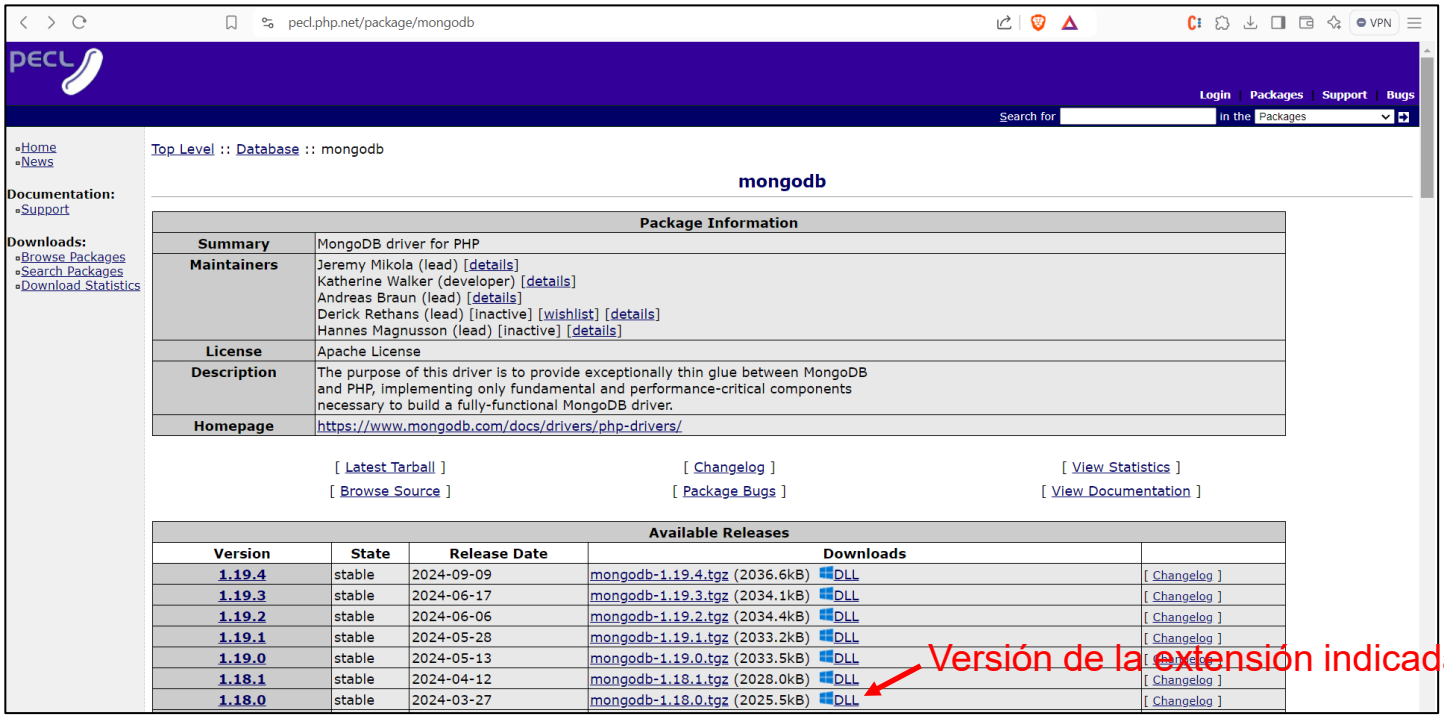
Ejemplo: script PHP para CRUD básico en MongoDB

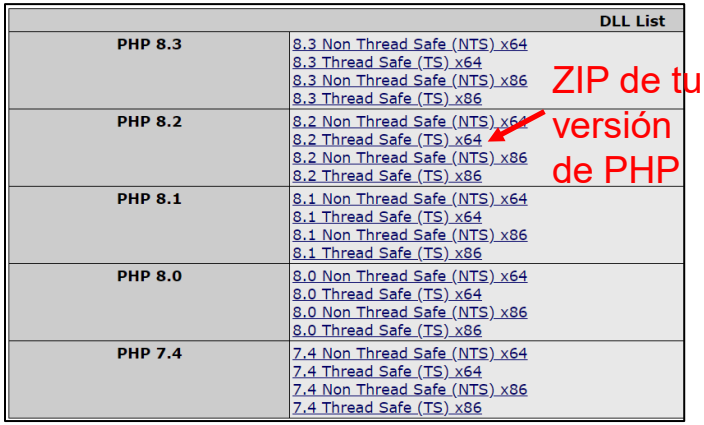
Requisitos:
- PHP
- Extensión MongoDB para PHP (fichero DLL para Windows)
- Composer
- MongoDB Community Server

Instalación de la extensión MongoDB para PHP

Para ver qué versión de la extensión necesitas, desde el directorio del proyecto, introduce la instrucción: `composer require mongodb/mongodb`

Te dará un error similar a este: `Cannot use mongodb/mongodb's latest version 1.19.1 as it requires ext-mongodb ^1.18.0 which is missing from your platform.`
En este ejemplo de error, la version de la extension que pide es la 1.18.0, que puedes decargar de https://pecl.php.net/package/mongodb haciendo clic en el icono DLL correspondiente. A continuación, descarga el fichero de tu version de PHP (puedes consultarla con el comando `php -v`). De las cuatro opciones que da para cada versión, lo habitual es que sea la que viene indicada como "Thread Safe (TS) x64".





- Descomprime el ZIP
- Copia el fichero `php_mongodb.dll` a la carpeta `C:\xampp\php\ext`
- Abre el fichero `C:\xampp\php\php.ini` y añade la línea `extension=mongodb`
- Reinicia Apache si estaba en marcha

2. Servidores de bases de datos

mongoDB

Ejemplo: script PHP para CRUD básico en MongoDB

Introduce de nuevo la instrucción para instalar las dependencias de MongoDB para PHP: `composer require mongodb/mongodb`

Esta instrucción creará una carpeta `vendor` y los ficheros `composer.json` y `composer.lock`.

A continuación, crea un fichero `index.php` que incluya la conexión a la base de datos MongoDB y ejemplos de operaciones CRUD (crear, leer, actualizar y borrar).

Recuerda que la carpeta que has creado para este ejemplo (p.ej. `php_mongodb`), debe estar en el directorio `C:\xampp\htdocs` (o su equivalente en Linux) para que se ejecute en el servidor Apache.

NOTA: para probar la aplicación, también puedes arrancar el servidor embebido de PHP en el directorio de trabajo (`php -S ...`), sin necesidad de tener el código en la carpeta `htdocs`.

2. Servidores de bases de datos

Ejemplo: script PHP para CRUD básico en MongoDB

```php
<?php
require 'vendor/autoload.php';

$client = new MongoDB\Client("mongodb://localhost:27017");
$database = $client->cine;
$collection = $database->peliculas;

echo "<h1>Operaciones CRUD PHP - MongoDB</h1>";

echo "<h2>Leer colección</h2>";
$documentos = $collection->find();
echo "<table><tr><th>Titulo</th><th>Director</th><th>Nota</th><th>Año</th><th>Presupuesto</th><th>Cartel</th><th>Trailer<th></tr>";
foreach ($documentos as $documento) {
    echo "<tr><td>" . $documento['titulo'] . "</td><td>" . $documento['director'] . "</td><td>" . $documento['nota'] . "</td>
    <td>" . $documento['anyo'] . "</td><td>" . $documento['presupuesto'] . "</td>
    <td><img style='width: 50px;' src='" . $documento['img_base64'] . "'/></td>
    <td><a href='" . $documento['url_trailer'] . "'>Ver</td></tr>";
}
echo "</table>";

echo "<h2>Leer documento</h2>";
echo "<table><tr><th>Titulo</th><th>Director</th><th>Nota</th><th>Año</th><th>Presupuesto</th><th>Cartel</th><th>Trailer<th></tr>";
$documento = $collection->findOne(['titulo' => 'Avengers Endgame']);
echo "<tr><td>" . $documento['titulo'] . "</td><td>" . $documento['director'] . "</td><td>" . $documento['nota'] . "</td>
    <td>" . $documento['anyo'] . "</td><td>" . $documento['presupuesto'] . "</td>
    <td><img style='width: 50px;' src='" . $documento['img_base64'] . "'/></td>
    <td><a href='" . $documento['url_trailer'] . "'>Ver</td></tr>";
echo "</table>";
```

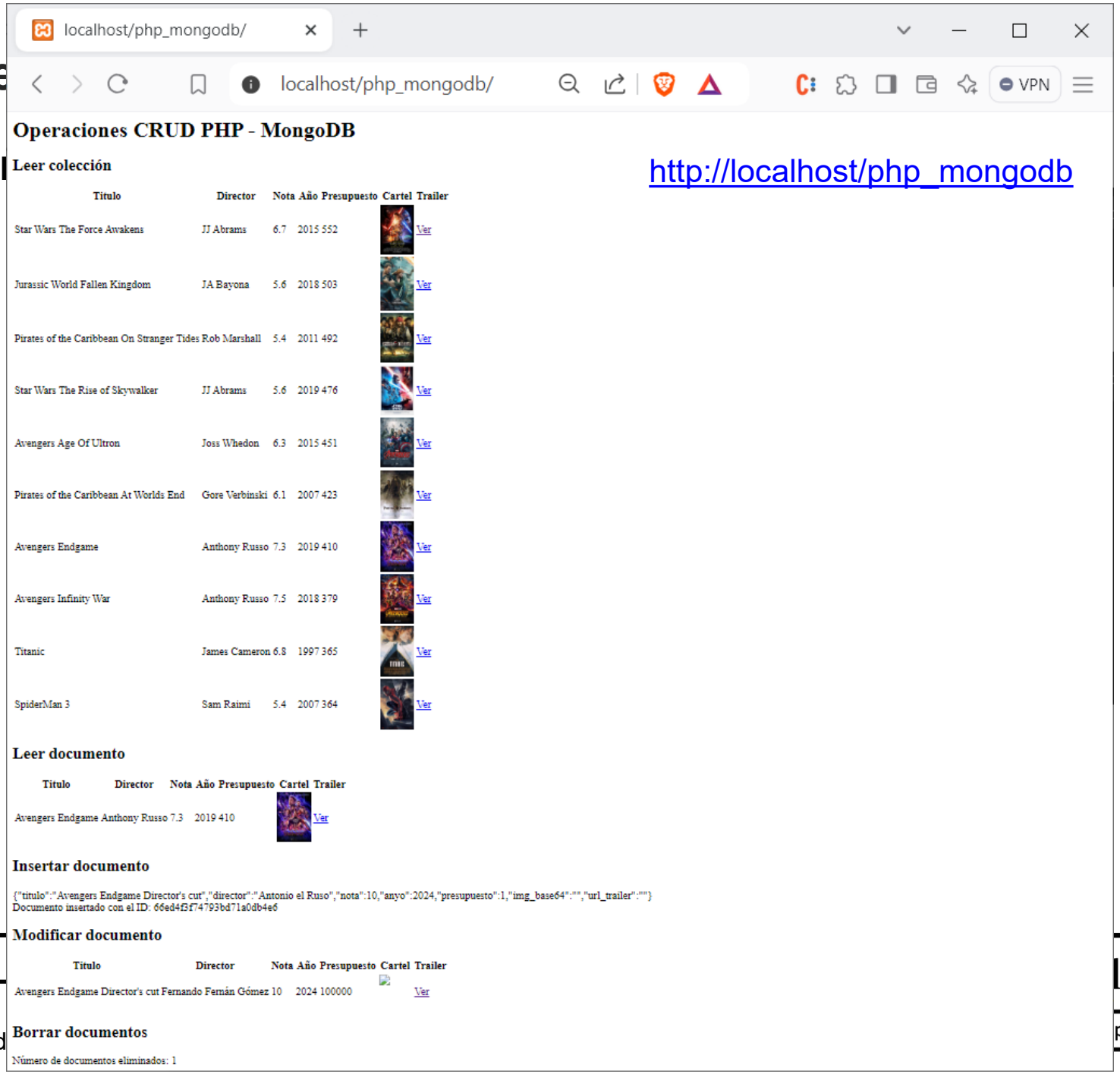
La cadena de conexión a un servidor MongoDB remoto protegido con contraseña sería:
`mongodb://usuario:contraseña@IP:puerto`

2. Servidores de bases de datos

mongoDB

Ejemplo: script PHP para CRUD básico en MongoDB

```
echo "<h2>Insertar documento</h2>";
$documento = [
    'titulo' => 'Avengers Endgame Director\'s cut',
    'director' => 'Antonio el Ruso',
    'nota' => 10,
    'anyo' => 2024,
    'presupuesto' => 1,
    'img_base64' => '',
    'url_trailer' => ''
];
echo json_encode($documento) . "<br>";
$insertOneResult = $collection->insertOne($documento);
echo "Documento insertado con el ID: " . $insertOneResult->getInsertedId();

echo "<h2>Modificar documento</h2>";
$updateResult = $collection->updateOne(
    ['titulo' => 'Avengers Endgame Director\'s cut'],  // Condición para encontrar el documento
    ['$set' => ['director' => 'Fernando Fernán Gómez', 'presupuesto' => 100000]]  // Valores a actualizar
);
echo "<table><tr><th>Titulo</th><th>Director</th><th>Nota</th><th>Año</th><th>Presupuesto</th><th>Cartel</th><th>Trailer<th></tr>";
$documento = $collection->findOne(['titulo' => 'Avengers Endgame Director\'s cut']);
echo "<tr><td>" . $documento['titulo'] . "</td><td>" . $documento['director'] . "</td><td>" . $documento['nota'] . "</td>
    <td>" . $documento['anyo'] . "</td><td>" . $documento['presupuesto'] . "</td>
    <td><img style='width: 50px;' src='" . $documento['img_base64'] . "'/></td>
    <td><a href='" . $documento['url_trailer'] . "'>Ver</td></tr>";
echo "</table>";

echo "<h2>Borrar documentos</h2>";
$deleteResult = $collection->deleteMany(['titulo' => 'Avengers Endgame Director\'s cut']);
echo "Número de documentos eliminados: " . $deleteResult->getDeletedCount();
```



2. Se

Ejempl

http://localhost/php_mongodb

mongoDB

DAW -

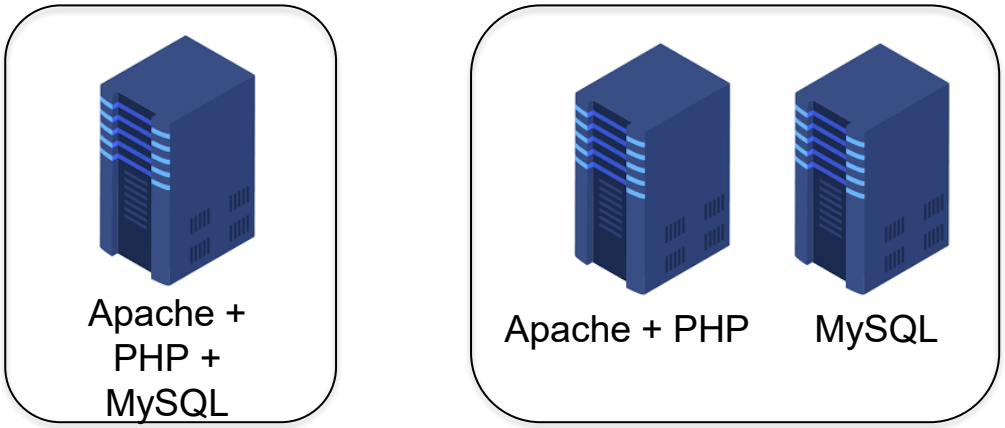
lorida

p Educatiu

3. Servid

2. Servidores de bases de datos

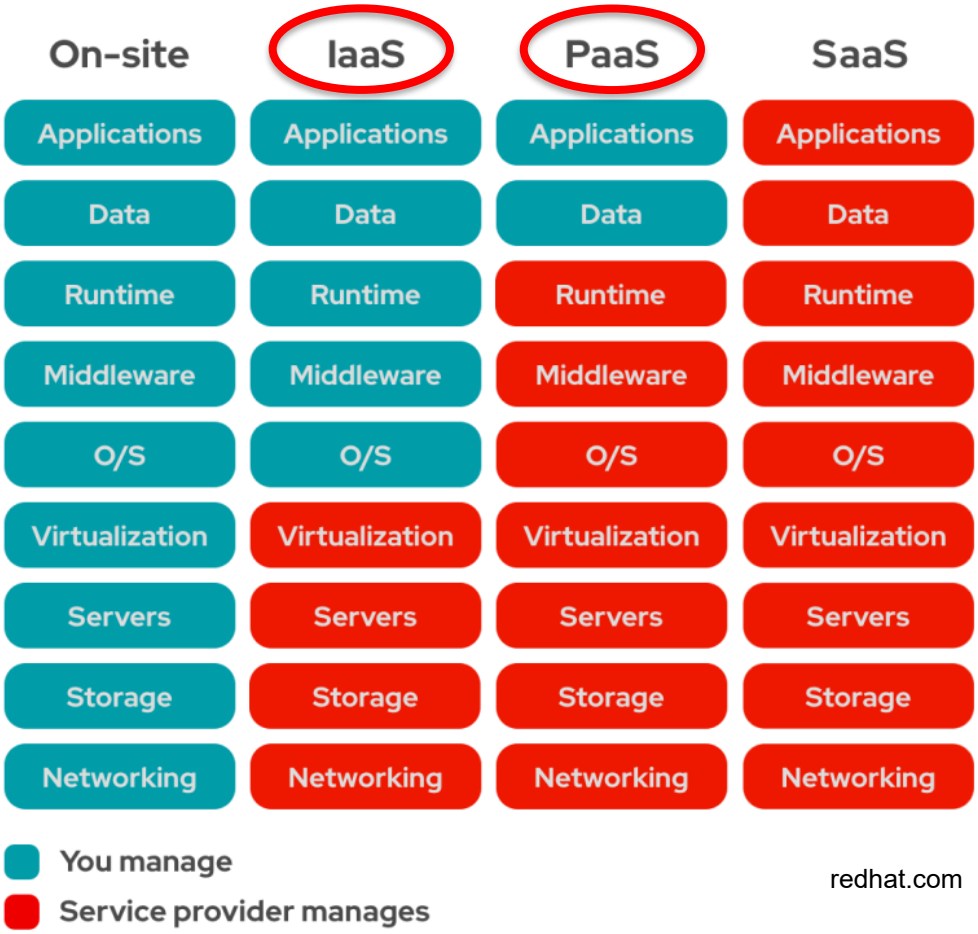
Recordar que los servidores de datos pueden estar alojados en la misma máquina donde estén ejecutándose las aplicaciones web o en otras máquinas.



La decisión de una arquitectura u otra dependerá de nuestras necesidades y/o limitaciones. En cualquier caso, es importante que la administración de la base de datos sea eficiente y segura.

3. IaaS/PaaS: *infrastructures/platforms as a service*

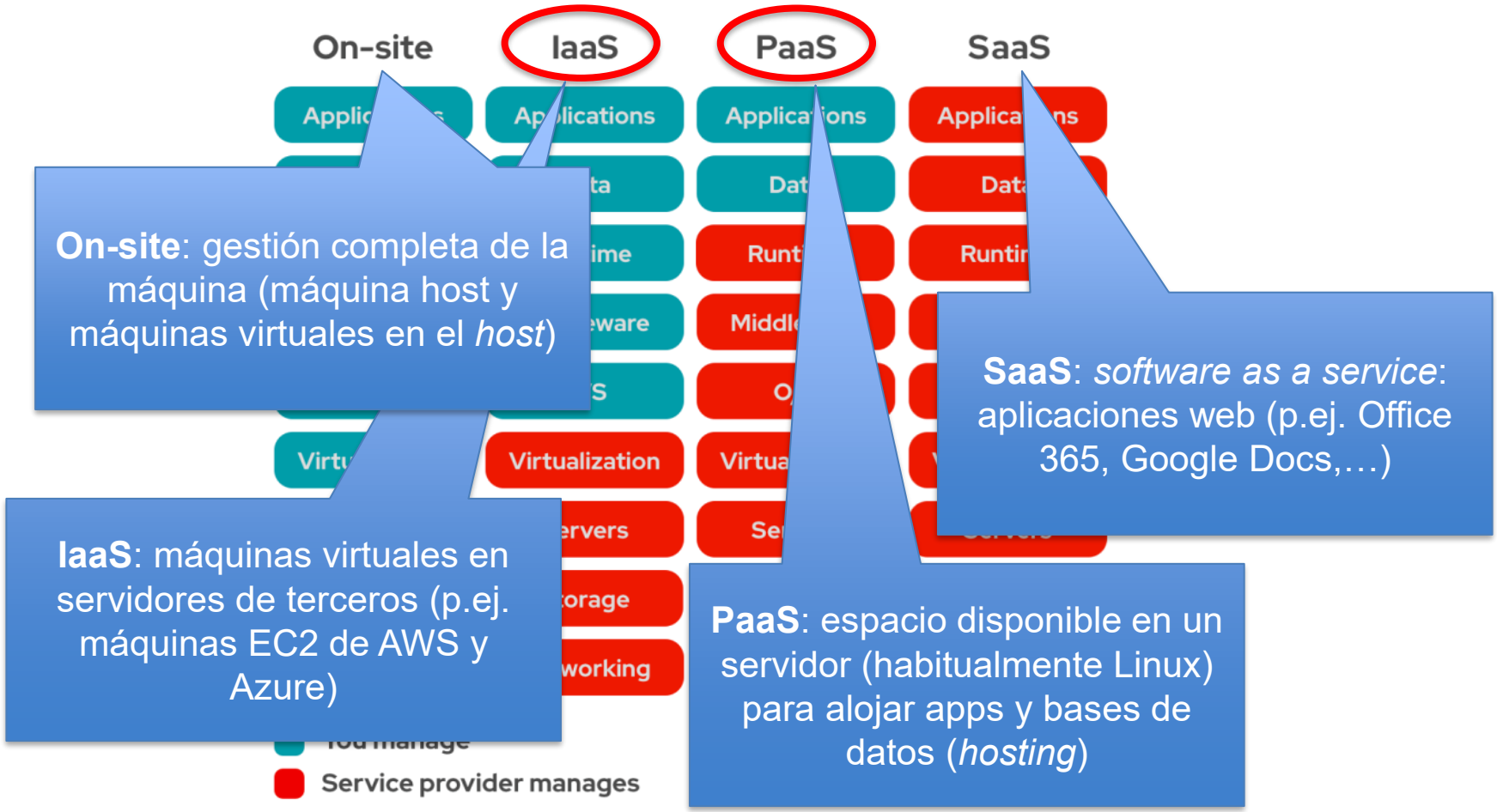
Infraestructuras y plataformas como servicios (IaaS y PaaS)



redhat.com

3. IaaS/PaaS: *infrastructures/platforms as a service*

Infraestructuras y plataformas como servicios (IaaS y PaaS)



3. IaaS/PaaS: *infrastructures/platforms as a service*

IaaS: Infrastructure as a Service 

AWS (Amazon Web Services) proporciona entre sus servicios la posibilidad de crear máquinas virtuales en la nube (EC2).

EC2 (*Elastic Compute Cloud*): permite a los usuarios alquilar computadores virtuales en los cuales pueden ejecutar sus propias aplicaciones.

AWS Academy: permite disponer de recursos AWS gratuitos para estudiantes.

Learner Lab: entorno de desarrollo que permite a los estudiantes utilizar diferentes servicios de AWS, como máquinas EC2 y contenedores.

En este tema nos vamos a centrar en la virtualización con máquinas EC2, por lo que crearemos una máquina virtual en Linux (Ubuntu) y otra en Windows.

- Creación y configuración de una instancia EC2 con Ubuntu
- Creación y configuración de una instancia EC2 con Windows Server

3. IaaS/PaaS: *infrastructures/platforms as a service*

IaaS: Configuración de la máquina virtual como servidor de aplicaciones web

- **Windows**: acceso por cliente de escritorio remoto (RDP) o por SSH
- **Ubuntu**: acceso por SSH

- Instalación en el equipo de los servicios necesarios:
  - ✓ XAMPP/LAMP (Apache + MySQL + PHP)
  - ✓ Otros servidores de bases de datos: PostgreSQL, MongoDB, etc.
  - ✓ JRE: entorno de ejecución de Java (JVM, Java Virtual Machine)
  - ✓ Node.js / Express
  - ✓ …

NOTA: es conveniente haber configurado la IP como elástica para que no varíe entre un inicio de la máquina virtual y el siguiente. Sin embargo, recuerda que tras cierto tiempo (unas pocas horas), la máquina virtual se desactiva (no se elimina) y hay que volver a arrancarla manualmente desde la consola del AWS Academy Learner Lab.

3. IaaS/PaaS: *infrastructures/platforms as a service*

PaaS: *Platform as a Service*

GitHub Pages https://pages.github.com/

Ofrece un servidor web para contenido estático, ideal para complementar la descripción de proyectos.

Servicio gratuito:

- ✓ 1GB
- ✓ No permite ejecutar código de aplicaciones (PHP, JS, Java,…)
- ✓ Subdominios (DNS)
- ✓ Certificados SSL (HTTPS)
- ✓ Permite redireccionar un dominio propio

3. IaaS/PaaS: *infrastructures/platforms as a service*

PaaS: *Platform as a Service*

InfinityFree https://www.infinityfree.com/

Ofrece una pila LAMP para desplegar aplicaciones web basadas en PHP y MySQL.

Servicio gratuito:

- ✓ 5GB
- ✓ PHP 8.2
- ✓ MySQL 5.7 y MariaDB 10.4
- ✓ Subdominios (DNS)
- ✓ Certificados SSL (HTTPS)
- ✓ Permite redireccionar un dominio propio

➢ Despliegue y configuración de una aplicación web con InfinityFree

3. IaaS/PaaS: *infrastructures/platforms as a service*

DBaaS: *Database as a Service*

Existe también la opción de tener nuestra base de datos separada de la aplicación en un servidor dedicado a bases de datos. Todas las tecnologías de bases de datos admiten su versión *cloud*.

Ejemplos de proveedores:

- ✓ Aruba Cloud (MySQL, PostgreSQL y MS SQL Server): de pago
- ✓ Amazon RDS (*Relational Database Service*): incluido en AWS Academy Learner Lab
- ✓ Aiven (MySQL, PostgreSQL): gratuito hasta 5 GB
- ✓ MongoDB Atlas (MongoDB): clúster gratuito de 512 MB

El funcionamiento del DBaaS es equivalente al de una conexión local o a una máquina virtual, como hemos visto con anterioridad. Cabe resaltar la necesidad de que el acceso esté debidamente protegido con una contraseña fuerte, ya que los ataques de tipo *ransomware* son frecuentes.

3. IaaS/PaaS: *infrastructures/platforms as a service*

DBaaS: *Database as a Service* - Sugerencias de trabajo

- Creación de una base de datos **Aiven**, gestión desde phpMyAdmin y consumo desde una app propia.
- Creación de una base de datos en **MongoDB Atlas**, gestión desde MongoDB Compass y consumo desde una app propia.
- Creación de un servicio **Amazon RDS** en AWS y conexión desde app.

4. Conclusiones

- Servidores de aplicaciones: más complejo que un servidor web (acceso a bases de datos y múltiples aplicaciones relacionadas con la lógica empresarial, no solo páginas web).

- Cada lenguaje de programación tiene el suyo, acompañado con frecuencia de un *framework* que facilita el desarrollo (Tomcat/SpringBoot para Java o Node.js/Express para Javascript).

- Los servidores de bases de datos complementan las aplicaciones web y varían en arquitectura y características (SQL: MySQL/MariaDB, PostgreSQL y NoSQL: MongoDB).

- Suelen gestionarse con herramientas tipo aplicación web (phpMyAdmin) o de escritorio (MySQL Workbench, HeidiSQL, Compass, etc.).

- Las IaaS y las PaaS facilitan el despliegue de aplicaciones en la nube, ya que proporcionan diversos elementos HW y SW configurables que se pueden ajustar a las necesidades de la aplicación, optimizando recursos.