Spring Platform

IT BOARDING

BOOTCAMP

Índice

IT BOARDING
BOOTCAMP

Introducción

IT BOARDING

BOOTCAMP

¿Que es Frontend?

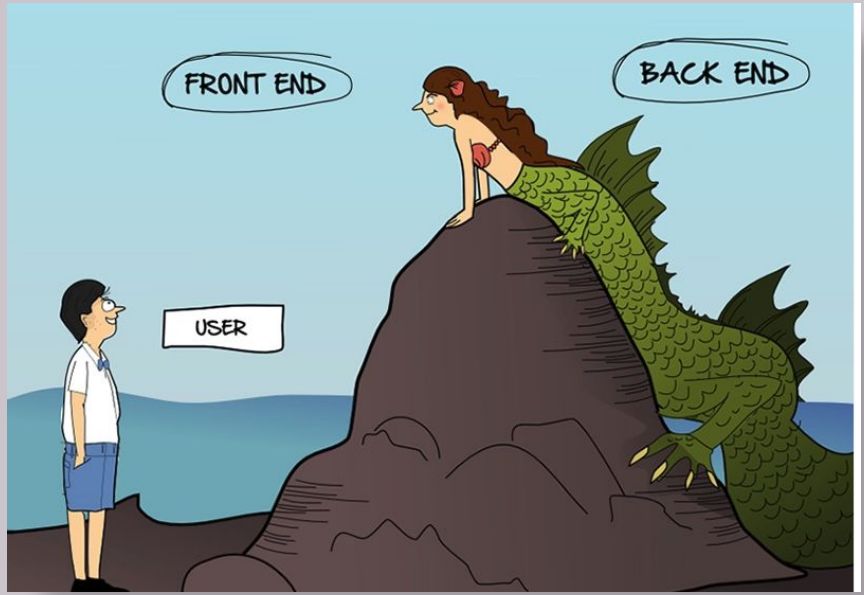
Es la parte de un software a la que un **usuario puede acceder e interactuar** directamente. (Interfaz gráfica)

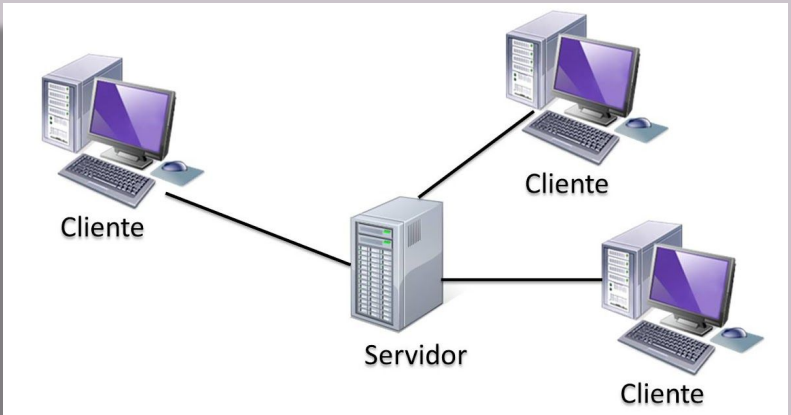
¿Que es Backend?

Capa de **acceso a datos** de un software o cualquier dispositivo, **contiene la lógica** de la aplicación que maneja dichos datos.

¿Que es FullStack?

Es el nombre que reciben los desarrolladores que pueden llevar a cabo una aplicación de **principio a fin**, es decir, tanto en su apartado frontend, como backend y todas las herramientas complementarias.





Arquitectura Cliente-Servidor

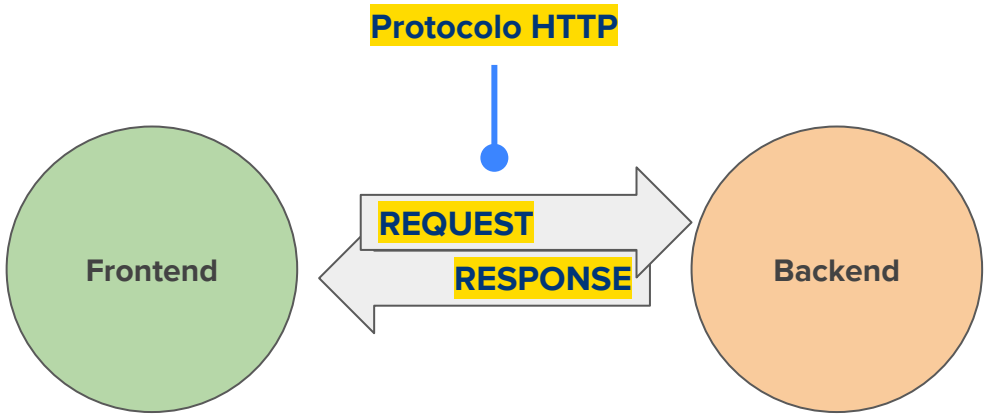
Es un **modelo** de **diseño de software** en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados **servidores**, y los demandantes, llamados **clientes**.

- Los **clientes** realizan peticiones a los servidores **(request).**
- Los **servidores** le brindan una respuesta respecto a lo que el cliente solicita **(response).**

// Comunicación

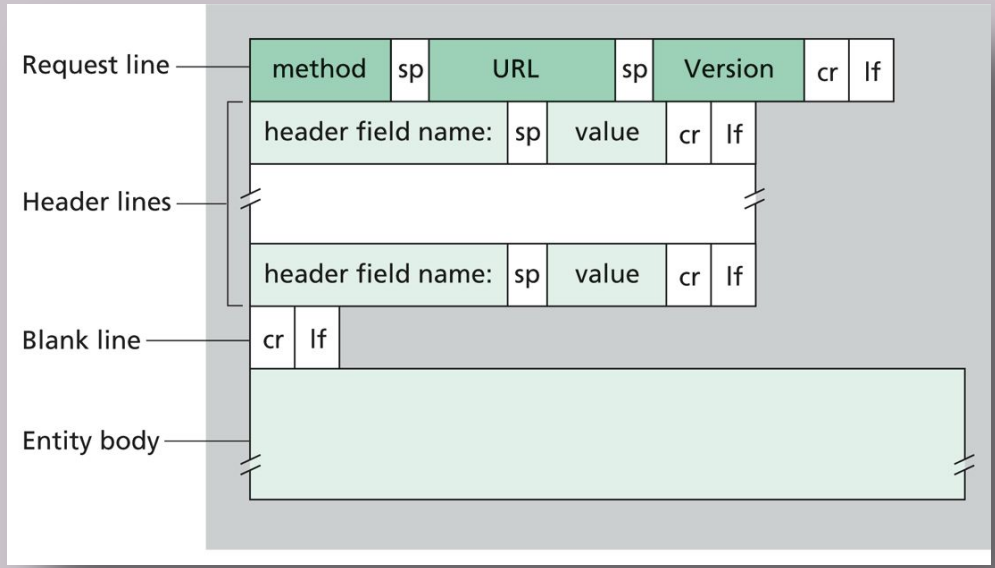
- El Backend es el encargado de exponer **APIs** mediante un protocolo llamado **HTTP** para que el frontend consuma y muestre la información según las necesidades del usuario.

- Hoy en día, se utiliza como estándar el protocolo **HTTPS**, que establece que la comunicación será **encriptada utilizando SSL** (Secure Sockets Layer)



REQUEST

- **METHOD:**
  - Los más utilizados son: GET, POST, PUT, DELETE. Ellos, hacen referencia al tipo de acción a ejecutar dentro del servidor.
- **URL:** Dirección a donde se encuentra la petición
- **HEADERS:** Atributos generales y esenciales (ejemplo: Auth)
- **BODY:** Cuerpo del mensaje utilizado en las peticiones para la transmisión de datos.



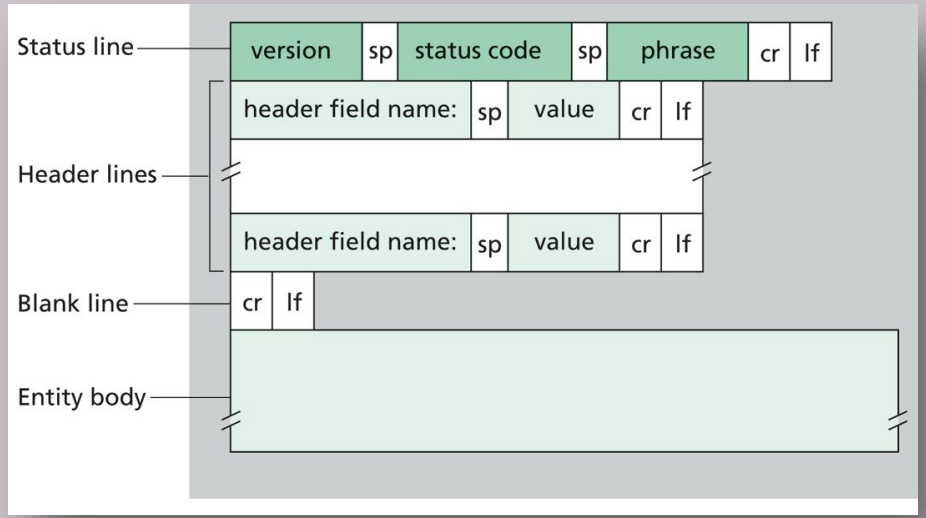
RESPONSE

STATUS: Código que representa que ha pasado con la petición

- **1xx:** Respuestas informativas. Indica que la petición ha sido recibida y se está procesando.
- **2xx:** Respuestas correctas. Indica que la petición ha sido procesada correctamente. (Ejemplo: **200 Success**
- **3xx:** Respuestas de redirección.
- **4xx:** Errores causados por el cliente. (Ejemplo: **404 Not Found**
- **5xx:** Errores causados por el servidor. (Ejemplo: **500 Internal Server Error**

HEADERS: Atributos generales y esenciales (ejemplo: Auth)

BODY: Respuesta del mensaje.



Framework y Librerías

IT BOARDING

BOOTCAMP

“Una librería/biblioteca es un conjunto de implementaciones funcionales de código en un lenguaje de programación que están a disposición para utilizar en cualquier aplicación/programa”.

IT BOARDING

BOOTCAMP

“Un framework es un entorno de desarrollo, con el cual podemos resolver problemas comunes más rápidamente, utilizando diferentes librerías para lograrlo.

IT BOARDING

BOOTCAMP

// ¿Qué es Maven?

- Es una herramienta de software para la gestión y construcción de proyectos Java.
- **Maven** nos permite instalar librerías dentro de nuestro proyecto de una forma fácil y escalable.
- **Intellij IDEA** está integrado con Maven y nos facilita el trabajo de instalación de complementos y librerías.

Spring & Spring Boot

IT BOARDING

BOOTCAMP

¿Por qué Spring?

Rendimiento

EVOLUCIONA

Cloud

GRATIS

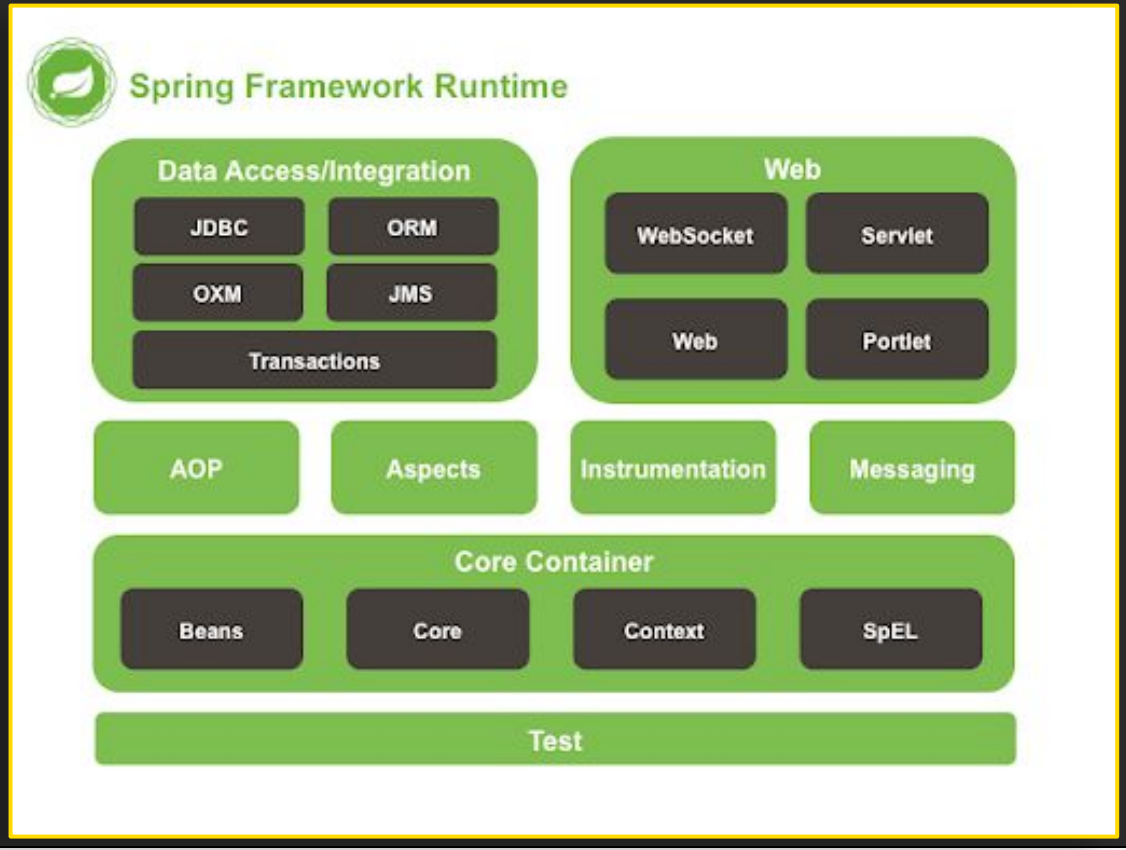
SEGURO

Gran Comunidad

Microservicios

Testing

Core de Spring



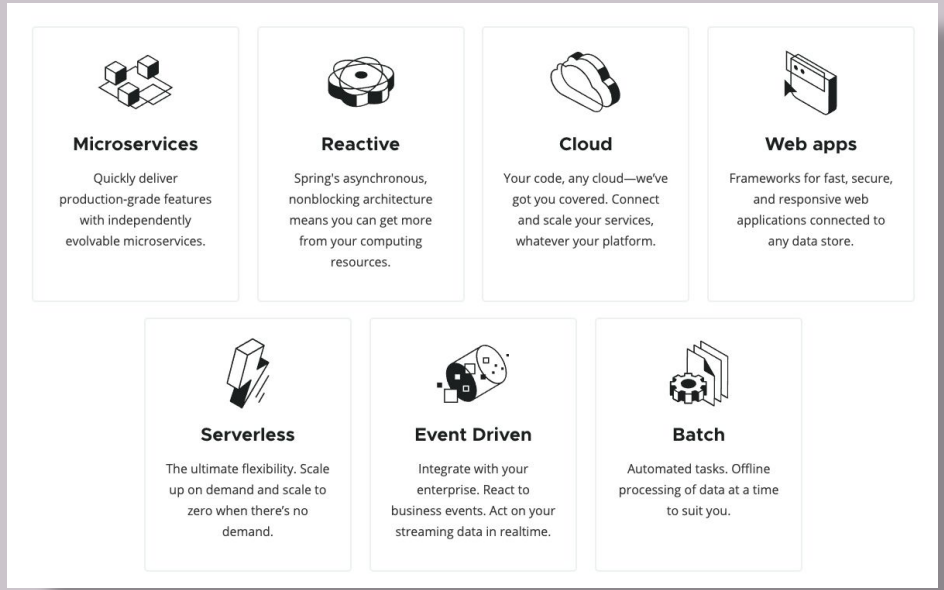
¿Qué es Spring Platform?

Es un conjunto de ***proyectos open source*** desarrollados en Java con el objetivo de agilizar el desarrollo de aplicaciones.

Cuenta con ***gran variedad de frameworks, que nos facilitan el trabajo*** desde el acceso a datos, infraestructura, creación de aplicaciones web, microservicios, etc.

https://spring.io

https://spring.io/projects



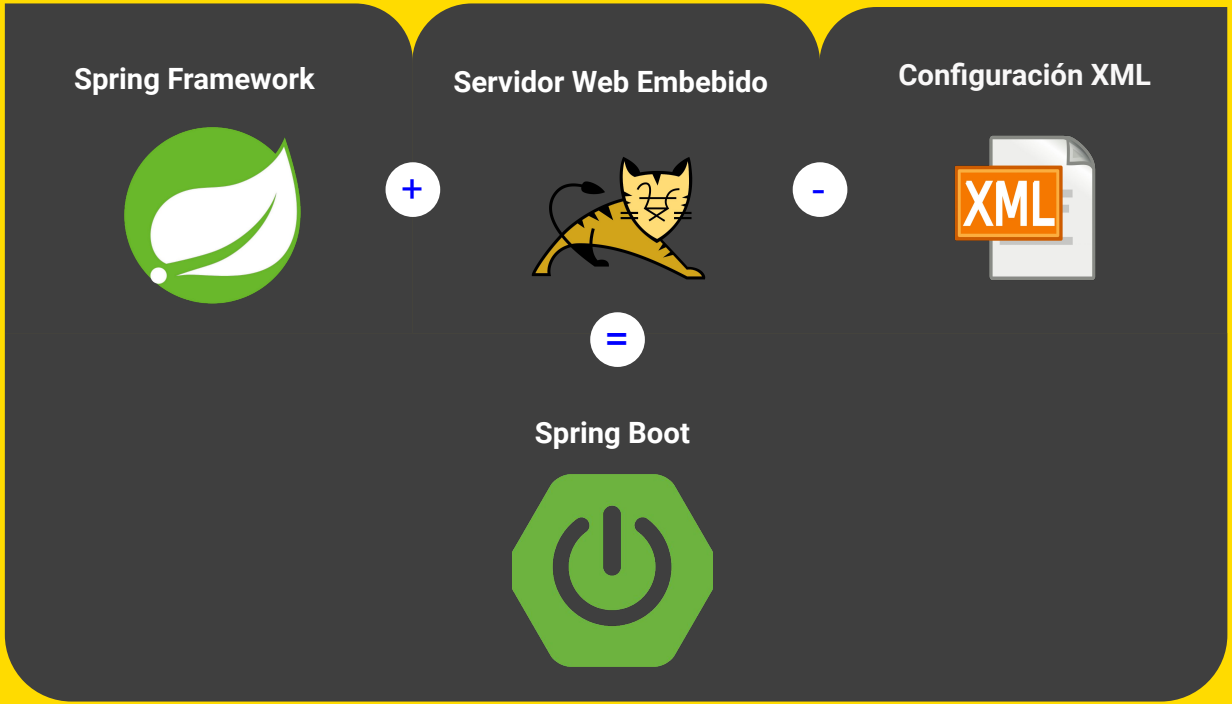
// Spring Boot

- Es una extensión de Spring framework que permite la creación **fácil y rápida de aplicaciones web** listas para producción con el concepto de “just run” (solo ejecutar).

- Requiere una **mínima configuración** y se complementa con muchos proyectos de Spring Platform y librerías de terceros.

https://spring.io/projects/spring-boot

Spring Framework vs Spring Boot



¡Creemos nuestro primer proyecto!

IT BOARDING

BOOTCAMP

// Spring Boot Initializr

- **Spring Initializr** es una pequeña utilidad Web que permite crear un esqueleto de un proyecto de **Spring Boot** con las opciones que queramos configurar.

- Permite elegir el **proveedor de dependencias** (Maven, Gradle, etc), el **lenguaje a utilizar** (Java, Kotlin, etc), el **tipo de empaquetado** (Jar, War), las **dependencias/librerías** que necesitamos, entre otras configuraciones iniciales.

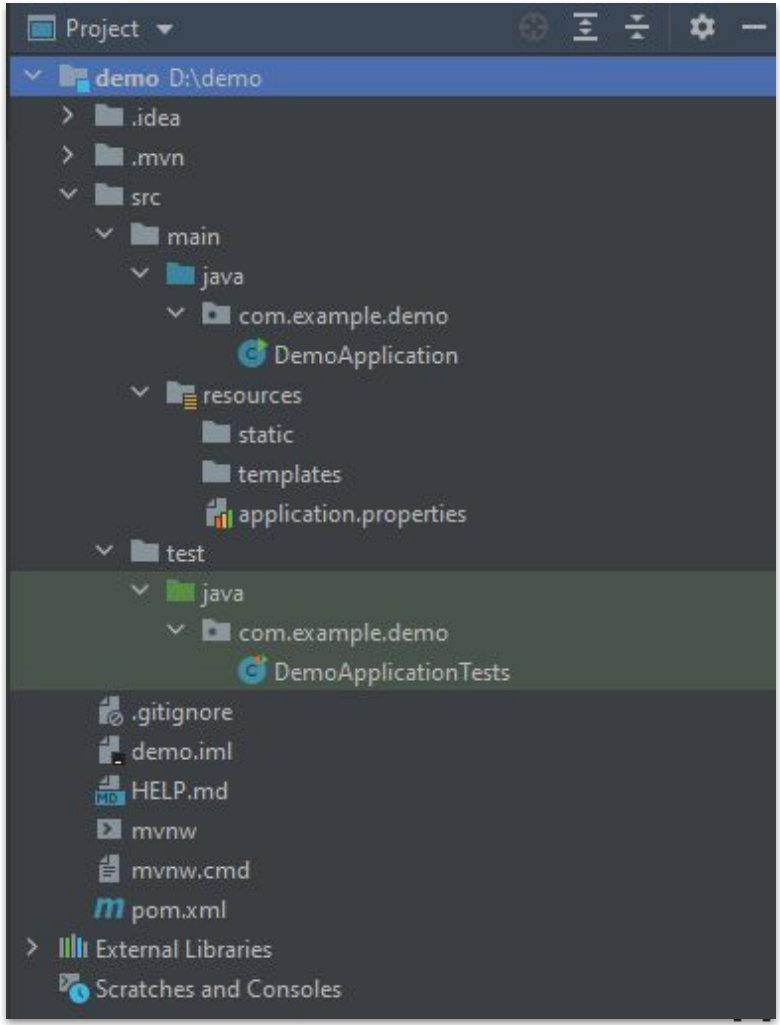
- Se puede utilizar desde su web oficial, o descargando un complemento para Intellij Idea o el IDE que estemos utilizando.

https://start.spring.io/

// Estructura del Proyecto

Se crean los siguientes archivos y directorios. Entre los más importantes se encuentran:

- pom.xml
- application.properties
- carpeta target (luego de buildear el proyecto)
- profile (definir entorno de trabajo)



// Application.java

- Spring Boot viene con un servidor web embebido que nos permite ejecutar la aplicación directamente desde el archivo main que contiene la anotación **@SpringBootApplication**, desde Intellij IDEA presionando el botón ▶

HelloWorldApplication.java

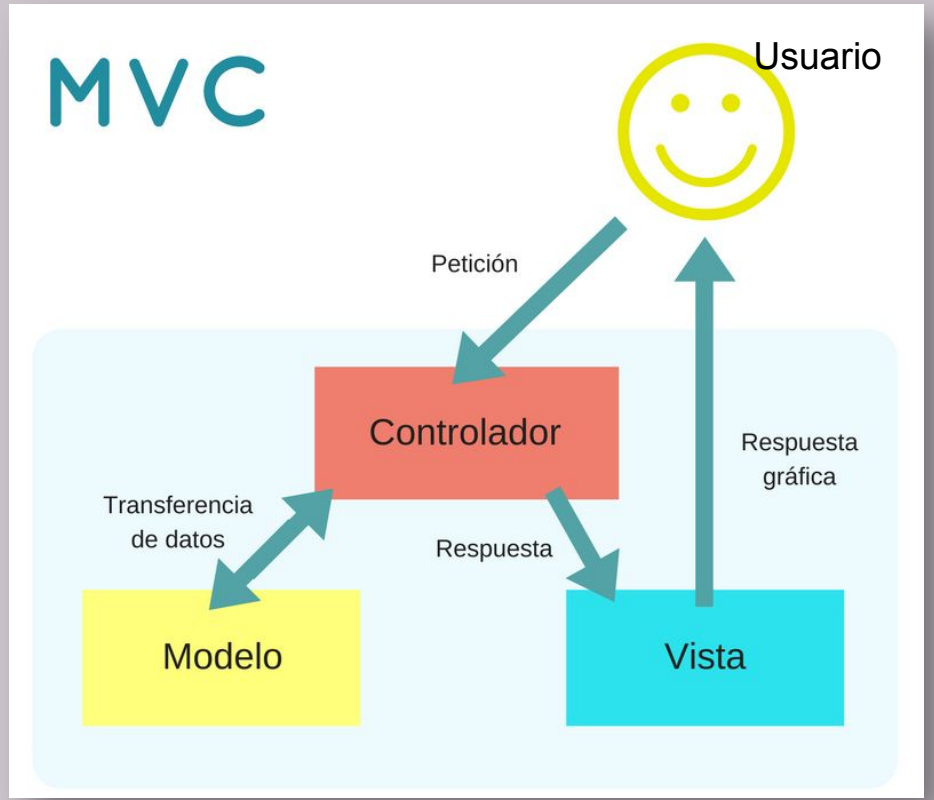
```java
package com.mercadolibre.helloworld;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class HelloWorldApplication {
  public static void main(String[] args) {
    SpringApplication.run(HelloWorldApplication.class, args);
  }
}
```

// MVC

- El **Modelo-Vista-Controlador** es un patrón de arquitectura de software que separa la l**ógica de negocio**, de la **lógica de la vista** en una aplicación.
- **Modelo:** Se encarga de los datos, generalmente (pero no obligatoriamente) consultando alguna base de datos.
- **Controlador:** Se encarga de “controlar”; recibe las órdenes del usuario, solicita los datos al modelo y se los comunica a la vista.
- **Vista:** Es la representación visual de los datos

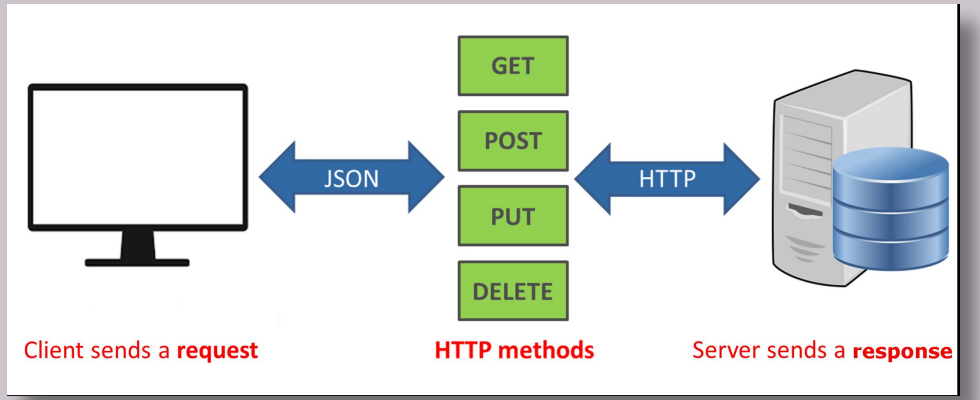


IT BOARDING
BOOTCAMP

// ¿Qué es una API REST?

- Una **API** (application programming interface) es un conjunto de funciones y procedimientos (métodos) que se usa para diseñar e integrar el software de aplicaciones.

- **REST (REpresentational State Transfer)** es un tipo de servicio que no posee estado. Es cualquier interfaz (interconexión) entre sistemas que use HTTP como protocolo para obtener datos o generar operaciones sobre los mismos en formatos como XML y JSON.



IT BOARDING
BOOTCAMP

// Creando una API

- Dentro del paquete **helloworld** creamos el archivo HelloRestController.java y en el mismo creamos el el método **sayHello()** que va a devolver un String (esto será el formato de respuesta de la API)

- **@RestController:** Esta anotación informa a Spring que la clase va a ser utilizada para generar una REST API.

- **@GetMapping:** Esta anotación configura el método sayHello() para que sea un **punto de entrada HTTP GET** en **"/"**

```java
package com.mercadolibre.helloworld;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloRestController {
  @GetMapping
  public String sayHello() {
    return "Hello World";
  }
}
```

¿Cómo obtener el parámetro {nombre}?

```
@GetMapping(path = "/{name}")
```

Vamos al RestController que necesitamos modificar, en este caso HelloRestController y editamos la anotación GetMapping de la siguiente manera:

Y en la interfaz del nuestro mensaje agregamos el parámetro “name”:

```
public String sayHello(@PathVariable String name)
```

```
@RestController
public class HelloRestController {
  @GetMapping(path = "/{name}")
  public String sayHello(@PathVariable String name) {
    return "Hello, " + name;
  }
}
```

Con esto le enseñamos a Spring a interpretar la URL y como queremos utilizar los parámetros.

// Diccionario/Resumen de Anotaciones de esta clase

- **@SpringBootApplication**: Nos permite especificar que trabajamos sobre una aplicación de Spring Boot. Habilita 3 características, la autoconfiguración del proyecto (@EnableAutoConfiguration), la búsqueda de componentes/paquetes de la aplicación (@ComponentScan) y la posibilidad de realizar configuraciones extras (@Configuration).

- **@RestController:** Anotación para identificar el controlador de un servicio REST

- **@GetMapping:** Anotación para “mapear” las peticiones mediante el método GET dentro de nuestra aplicación.

¿Dudas?

¿Preguntas?

IT BOARDING

BOOTCAMP