Programación de Aplicaciones Telemáticas

TEMA 7: SPRING BOOT

AGENDA

SESSIÓN 1

- Spring Framework
- Spring Boot
- Mi primera aplicación Spring
- Scaffolding de un proyecto
- References

AGENDA

SESSIÓN 2

- Especificaciones Jakarta EE
- Spring Core
- Spring Web
- Modelos de Concurrencia

AGENDA

SESSIÓN 3

- Error Handling
- Consumiendo HTTP Endpoints
- Validacion de Bean
- Configuracion

AGENDA

SESSIÓN 4

- Logging
- Scheduling
- Actuator
- Spring Security

AGENDA

SESSIÓN 5

- Arquitectura Neflix
- Arquitectura K8S
- Servicios auxiliares

SESSIÓN 1

SPRING FRAMEWORK

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications on any kind of deployment platform.

SPRING FRAMEWORK

Why Spring?

Spring came into being in 2003 as a response to the complexity of the early J2EE specifications.

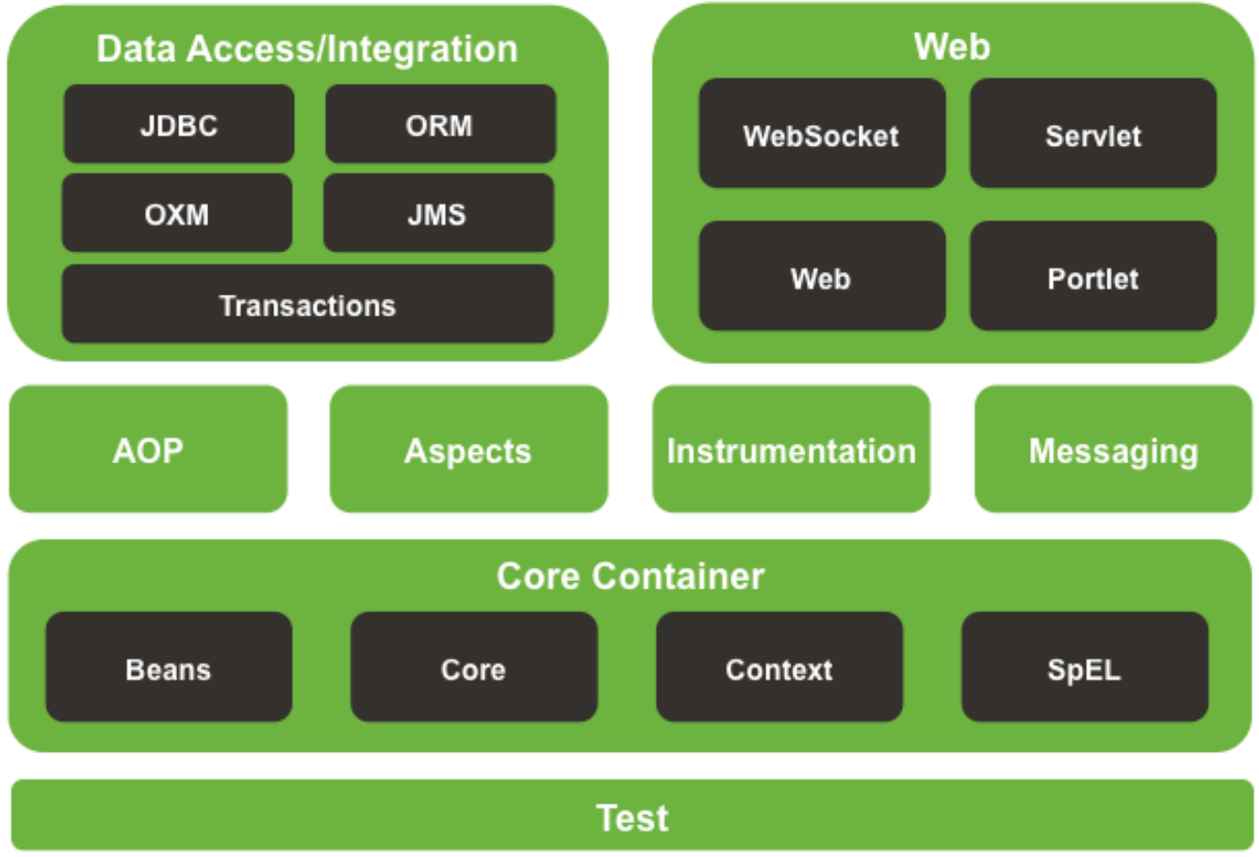
SPRING FRAMEWORK

The Spring programming model does not embrace the Java EE platform specification; rather, it integrates with carefully selected individual specifications from the EE umbrella

SPRING FRAMEWORK

- Servlet API (JSR 340)
- Concurrency Utilities (JSR 236)
- JSON Binding API (JSR 367)
- Bean Validation (JSR 303)
- JPA (JSR 338)
- JMS (JSR 914)
- Dependency Injection (JSR 330)
- Common Annotations (JSR 250)

SPRING FRAMEWORK



SPRING FRAMEWORK

¿Cual es el patrón de diseño detras de Spring Framework?

SPRING BOOT

Spring Boot helps you to create stand-alone, production-grade Spring-based Applications that you can run. We take an opinionated view of the Spring platform and third-party libraries, so that you can get started with minimum fuss.

SPRING BOOT

ECOSISTEMA SPRING

https://spring.io/projects

SPRING BOOT

SPRING INITIALIZR

https://start.spring.io/

SPRING BOOT

SPRING BOOT STARTERS

Starters are a set of convenient dependency descriptors that you can include in your application. You get a one-stop shop for all the Spring and related technologies that you need without having to hunt through sample code and copy-paste loads of dependency descriptors.

https://docs.spring.io/spring-boot/docs/2.3.3.RELEASE/reference/htmlsingle/#using-boot-starter

SPRING BOOT

MI PRIMERA APLICACIÓN SPRING

```
curl https://start.spring.io/starter.zip \
    -d dependencies=web,actuator,devtools \
    -d bootVersion=2.4.1 \
    -o my-project.zip
```

- https://start.spring.io/
- https://docs.spring.io/initializr/docs/current/reference/htm
  line

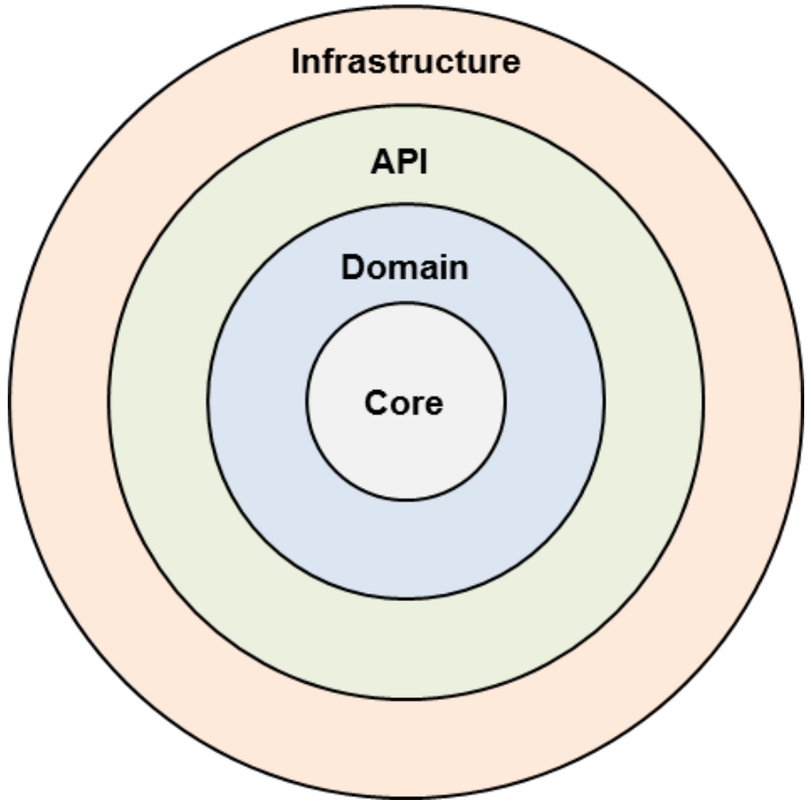
SPRING BOOT

MI PRIMERA APLICACION SPRING

```
@SpringBootApplication
public class DemoApplication {

        private static ConfigurableApplicationContext applicat

        public static void main(String[] args) {
                applicationContext = SpringApplication.run(Dem
                displayAllBeans();
        }

        public static void displayAllBeans() {
                String[] allBeanNames = applicationContext.get
                AtomicInteger counter = new AtomicInteger();
                Arrays.asList(allBeanNames).stream()
                                .map(bean -> counter.incrementAndGet()
```

SPRING BOOT

SCAFFOLDING DE UN PROYECTO



REFERENCIES

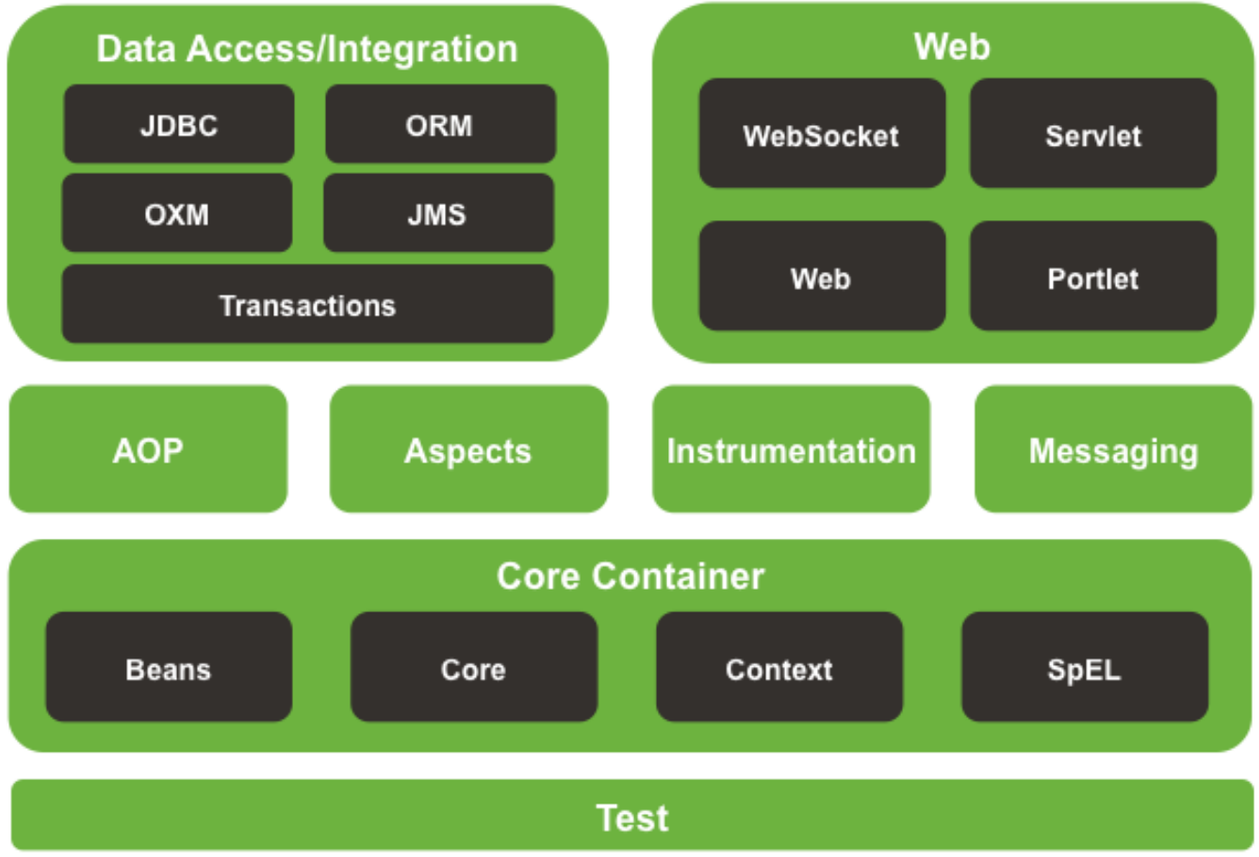
- https://docs.spring.io/spring-framework/docs/current/refe
- https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#spring
- https://docs.spring.io/spring-boot/docs/current/reference/
- https://docs.spring.io/spring-framework/docs/current/reference/html/web.html
- https://docs.spring.io/spring-boot/docs/2.3.3.RELEASE/reference/htmlsingle/#using-boo
- https://docs.spring.io/initializr/docs/current/reference/htm line

SESSIÓN 2

ESPECIFICACIONES JAKARTA EE

- Dependency Injection (JSR 330)
- Common Annotations (JSR 250)
- Servlet API (JSR 340)
- Concurrency Utilities (JSR 236)
- JSON Binding API (JSR 367)
- Bean Validation (JSR 303)

ESPECIFICACIONES JAKARTA EE



ESPECIFICACIONES JAKARTA EE

Core:

- Dependency Injection (JSR 330)
- Common Annotations (JSR 250)

ESPECIFICACIONES JAKARTA EE

Web:

- Servlet API (JSR 340)
- Concurrency Utilities (JSR 236)
- JSON Binding API (JSR 367)
- Bean Validation (JSR 303)

SPRING CORE

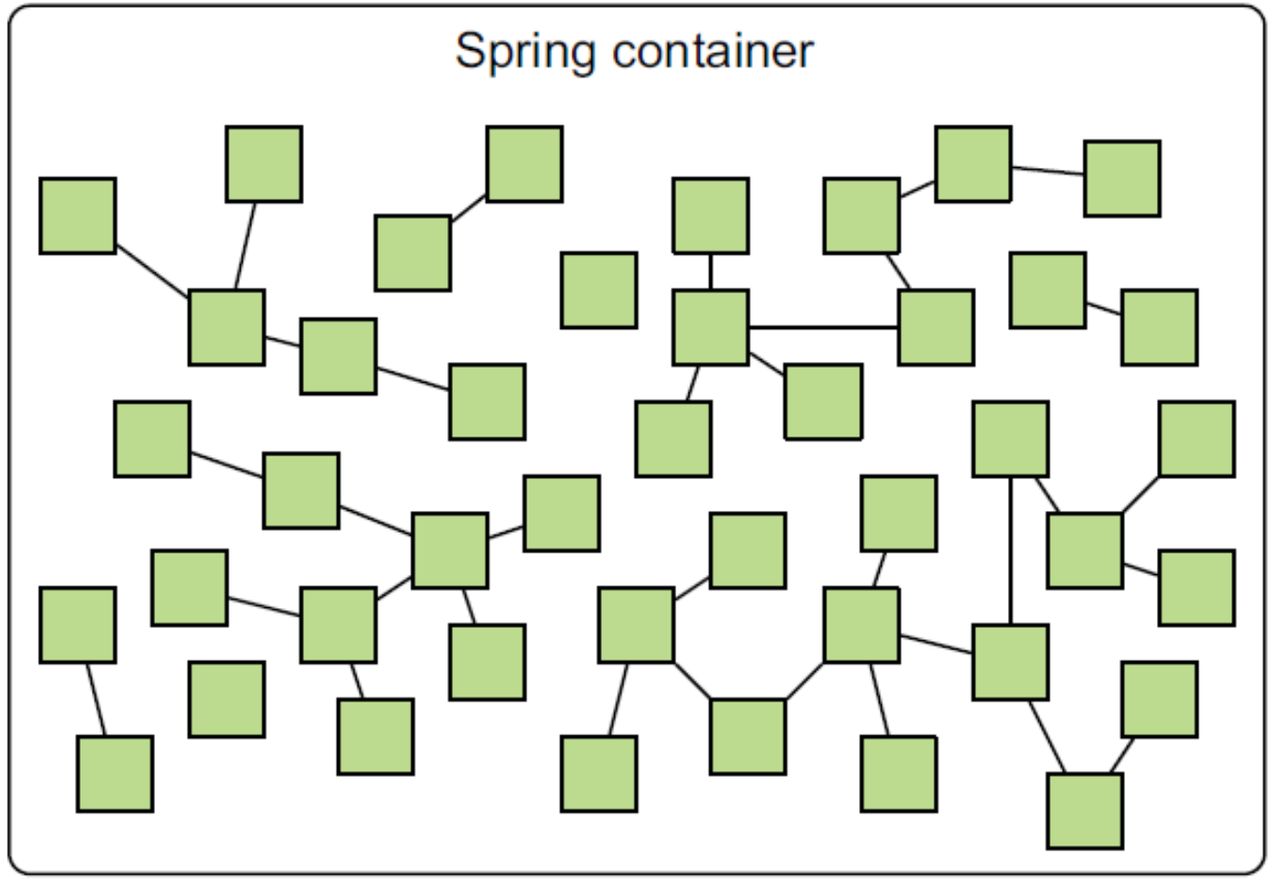
SPRING CORE

ESPECIFICACIONES JAKARTA EE

- Dependency Injection (JSR 330)
- Common Annotations (JSR 250)

SPRING CORE

SPRING CONTAINER



SPRING CORE

SPRING CONTAINER

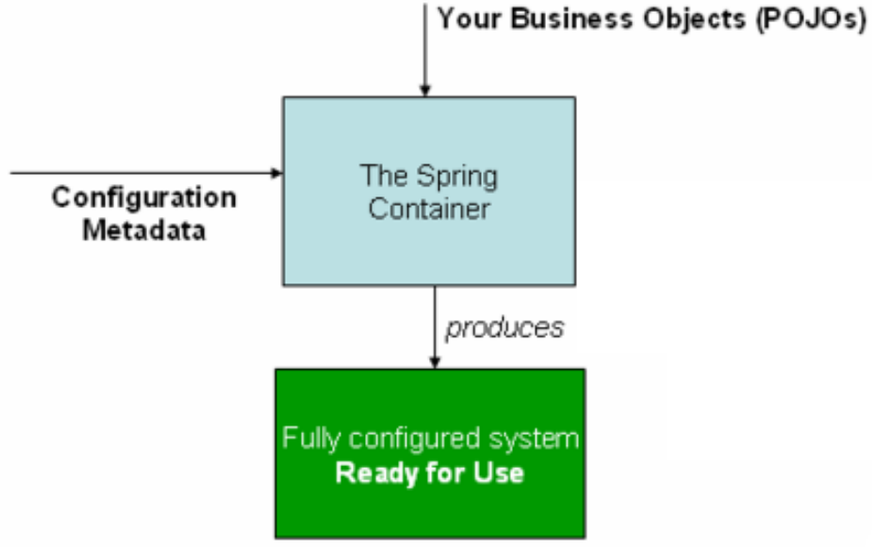


Figure 1. The Spring IoC container

SPRING CORE

SPRING CONTAINER

In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container.

SPRING WEB

ESPECIFICACIONES JAKARTA EE

- Servlet API (JSR 340)
- Concurrency Utilities (JSR 236)
- JSON Binding API (JSR 367)
- Bean Validation (JSR 303)

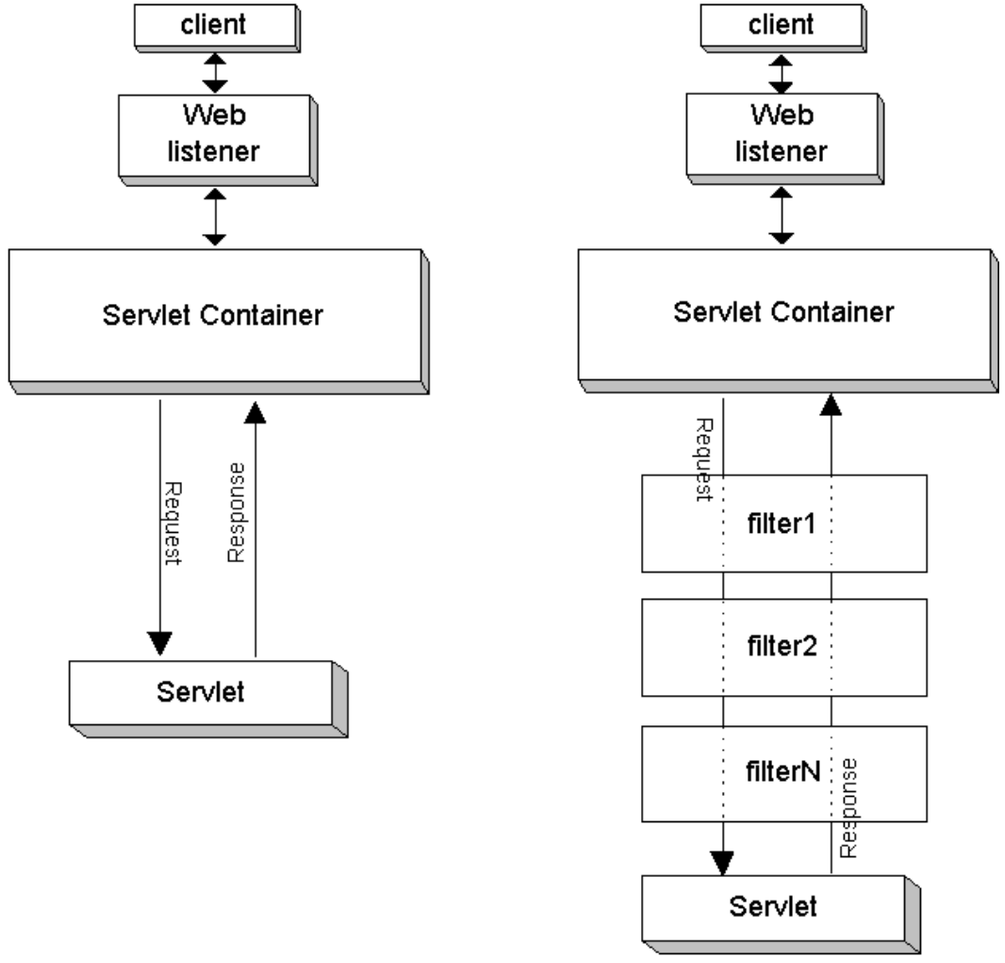
SPRING WEB

SERVLET API (JSR 340)

A Jakarta Servlet (formerly Java Servlet) is a Java software component that extends the capabilities of a server. Although servlets can respond to many types of requests, they most commonly implement web containers for hosting web applications on web servers and thus qualify as a server-side servlet web API.

SPRING WEB

SERVLET API (JSR 340)



SPRING WEB

https://docs.spring.io/spring-framework/docs/current/reference/html/web.html#spring-web

SPRING WEB

PACKAGE ORG.SPRINGFRAMEWORK.WEB.SERVLET

https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/servlet/package-summary.html

SPRING WEB

DISPATCHERSERVLET

Spring MVC, as many other web frameworks, is designed around the front controller pattern where a central Servlet, the DispatcherServlet, provides a shared algorithm for request processing, while actual work is performed by configurable delegate components.

SPRING WEB

JAX-RS EXAMPLE

```
@GET
@Path("/{id}")
@Produces("application/json")
public Response getUserById(@PathParam("id") int id) throws UR
{
        User user = DB.get(id);
        if(user == null) {
                return Response.status(404).build();
        }
        return Response
                        .status(200)
                        .entity(user)
                        .contentLocation(new URI("/user-manage
}
```

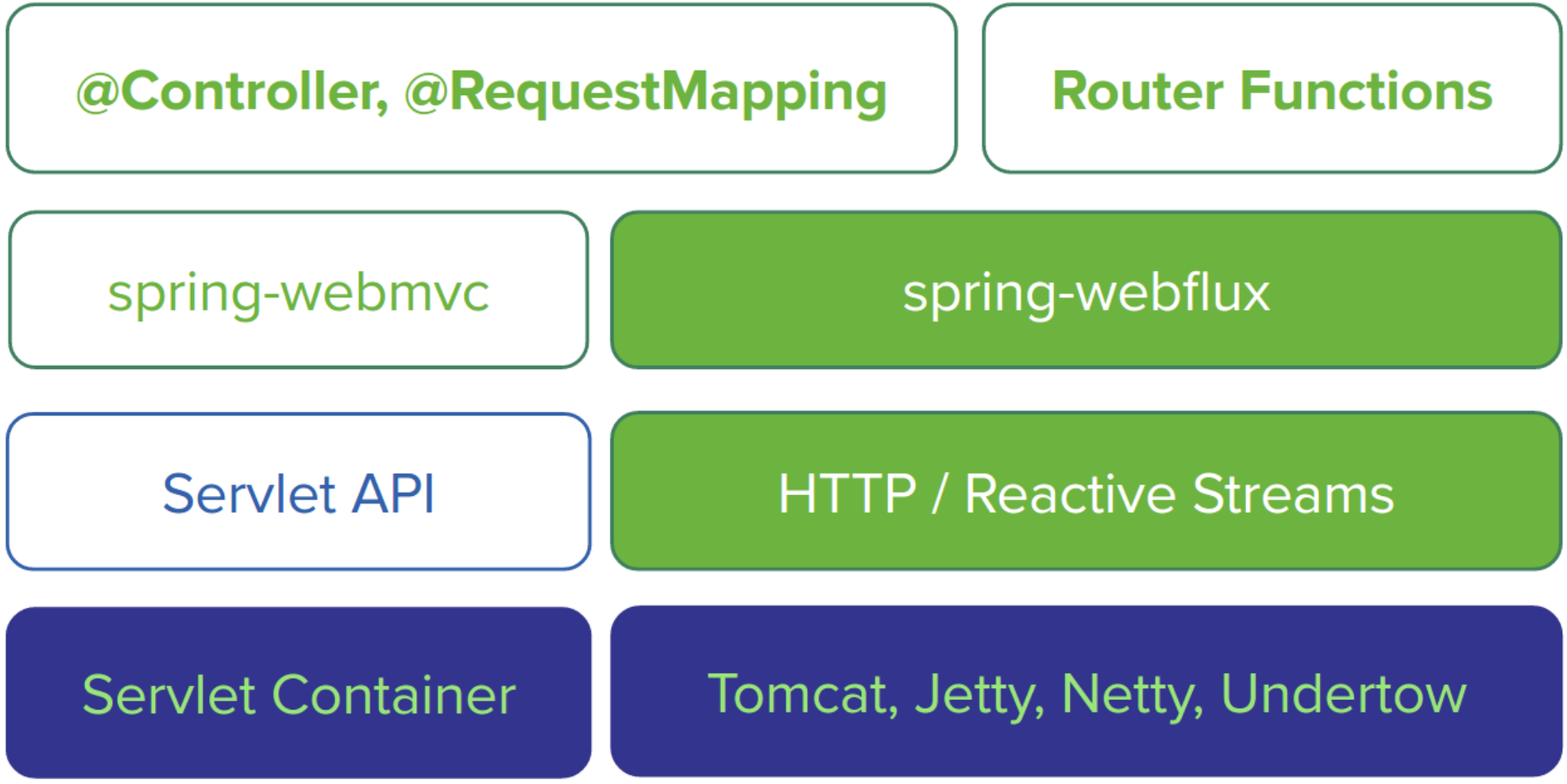
SPRING WEB

SPRING WEB EXAMPLE

```
@GetMapping("/{id}")
public Stormtrooper getTrooper(@PathVariable("id") String id)

        Stormtrooper stormtrooper = trooperDao.getStormtrooper
        if (stormtrooper == null) {
                throw new NotFoundException();
        }
        return stormtrooper;
}
```


EXAMPLE

MODELOS DE CONCURRENCIA



REFERENCIES

- https://spring.io/blog/2018/12/12/how-fast-is-spring
- http://tomcat.apache.org/tomcat-10.0-doc/architecture/requestProcess.html
- https://medium.com/javarevisited/spring-beans-in-depth-a6d8b31db8a1
- https://medium.com/javarevisited/top-spring-annotations-4f691babe458
- https://martinfowler.com/eaaCatalog/frontController.html

SESSIÓN 3

SPRING WEB

ERROR HANDLING

- ExceptionHandler
- ControllerAdvice

SPRING WEB

ERROR HANDLING

Define a method to handle exceptions and annotate that with @ExceptionHandler at @Controller level

```
@ResponseStatus(value=HttpStatus.CONFLICT,
        reason="Data integrity violation")  // 409
@ExceptionHandler(DataIntegrityViolationException.class)
public void conflict() {
        // Nothing to do
}
```

SPRING WEB

ERROR HANDLING

Spring brings support for a global @ExceptionHandler with the @ControllerAdvice annotation

```
@ControllerAdvice
public class RestResponseEntityExceptionHandler
  extends ResponseEntityExceptionHandler {

    @ExceptionHandler(value= { IllegalArgumentException.class}
    protected ResponseEntity< Object > handleConflict(
      RuntimeException ex, WebRequest request) {
        String bodyOfResponse = "This should be application sp
        return handleExceptionInternal(ex, bodyOfResponse,
          new HttpHeaders(), HttpStatus.CONFLICT, request);
    }
}
```

SPRING WEB

CONSUMIENDO HTTP ENDPOINTS

```
RestTemplate restTemplate = new RestTemplate();
String fooResourceUrl = "http://localhost:8080/api/foos";
ResponseEntity< String > response = restTemplate
        .getForEntity(fooResourceUrl + "/1", String.class);
```

https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-resttemplate

https://docs.spring.io/spring-framework/docs/current/javadoc-api/index.html?org/springframework/web/client/RestTemplate.html

SPRING WEB

VALIDACIÓN DE BEAN

```java
public class PersonForm {

        @NotNull
        @Size(min=2, max=30)
        private String name;

        @NotNull
        @Min(18)
        private Integer age;
}
```

https://hibernate.org/validator/

https://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/#validator-defineconstraints-spec

```java
@Controller
public class WebController {

        @PostMapping("/")
        public String checkPersonInfo(@Valid PersonForm person
                BindingResult bindingResult) {

                if (bindingResult.hasErrors()) {
                        return "KO";
                }
                return "OK";
        }
}
```

https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/validation/BindingResult.html

SPRING WEB

CONFIGURACION

Cual es el valor?

- Externalizar configuracion
- Soporte de múltiples entornos
- Evitar tener configuración en Código
- Facilitar el testing

https://docs.spring.io/spring-boot/docs/current/reference/html/appendix-application-properties.html

SPRING WEB

CONFIGURACION

```
@Configuration
public class WebConfig {

        @Bean
        public RestTemplateBuilder restTemplateBuilder() {
                return new RestTemplateBuilder()
                                .setConnectTimeout(1000)
                                .setReadTimeout(1000)
                                .customizers(rtc);
        }
        @Bean
        public RestTemplate restTemplate(final RestTemplateBui
                return restTemplateBuilder.build();
        }
}
```

SPRING WEB

CONFIGURACION

```
@RestController
@RequestMapping(value = "/api")
public class heroesController {

        @Autowired
        RestTemplate restTemplate;

        @GetMapping(value = "/service1", produces = MediaType.
        public ResponseEntity<model1[]> getHeroes() {
                final String url = http://localhost:8080/servi
                final ResponseEntity<model1[]> response = rest
                return response;
        }
}
```

REFERENCIES

- https://spring.io/blog/2013/11/01/exception-handling-in-spring-mvc
- https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-resttemplate
- https://hibernate.org/validator/

SESSIÓN 4

- Logging
- Scheduling
- Actuator
- Spring Security

LOGGING

WHY

Logging is the process of writing log messages during the execution of a program to a central place. This logging allows you to report and persist error and warning messages as well as info messages

LOGGING

```
2019-03-05 10:57:51.112  INFO 45469 --- [           main] org.
2019-03-05 10:57:51.253  INFO 45469 --- [ost-startStop-1] o.a.
2019-03-05 10:57:51.253  INFO 45469 --- [ost-startStop-1] o.s.
2019-03-05 10:57:51.698  INFO 45469 --- [ost-startStop-1] o.s.
2019-03-05 10:57:51.702  INFO 45469 --- [ost-startStop-1] o.s.
```

LOGGING

LOG FORMAT

- Date and Time: Millisecond precision and easily sortable.
- Log Level: ERROR, WARN, INFO, DEBUG, or TRACE.
- Process ID.
- A --- separator to distinguish the start of actual log messages.
- Thread name: Enclosed in square brackets
- Logger name: This is usually the source class name
- The log message.

LOGGING

LOG LEVELS

- ERROR Other runtime errors or unexpected conditions. Expect these to be immediately visible on a status console.
- WARNING Use of deprecated APIs, poor use of API, 'almost' errors, other runtime situations that are undesirable or unexpected, but not necessarily "wrong".

LOGGING

LOG LEVELS

- INFO Interesting runtime events (startup/shutdown). Expect these to be immediately visible on a console, so be conservative and keep to a minimum.
- DEBUG detailed information on the flow through the system. Expect these to be written to logs only.
- TRACE more detailed information. Expect these to be written to logs only.

LOGGING

```
logging.level.root=warn
logging.level.org.springframework.web=debug
logging.level.org.hibernate=error
```

SCHEDULING

Con Spring Boot, es possible planificar la ejecucion de Beans para un proposito concreto.

Es posible planificarlo en 3 maneras diferentes: cron(), fixedDelay(), or fixedRate()

SCHEDULING

```
@Sl4j
@Component
public class ScheduledTasks {

        private static final SimpleDateFormat dateFormat =
                new SimpleDateFormat("HH:mm:ss");

        @Scheduled(fixedRate = 5000)
        public void reportCurrentTime() {
                log.info("The time is now {}", dateFormat.form
        }
}
```

SCHEDULING

```
@Sl4j
@Component
public class ScheduledTasks {

        private static final SimpleDateFormat dateFormat =
                new SimpleDateFormat("HH:mm:ss");

        @Scheduled(cron = "0 15 10 15 * ?")
        public void scheduleTaskUsingCronExpression() {

                long now = System.currentTimeMillis() / 1000;
                log.info("schedule tasks using cron jobs - {}"
        }
}
```

ACTUATOR

DEFINITION

An actuator is a manufacturing term that refers to a mechanical device for moving or controlling something. Actuators can generate a large amount of motion from a small change.

ACTUATOR

WHY

Actuator endpoints let you monitor and interact with your application. Spring Boot includes a number of built-in endpoints and lets you add your own. For example, the health endpoint provides basic application health information.

ACTUATOR

ENDPOINTS

auditevents, beans, caches, conditions, configprops, env, flyway, health, httptrace, info, integrationgraph, loggers, liquibase, metrics, mappings, scheduledtasks, sessions, shutdown, startup, threaddump

SPRING SECURITY

SPRING SECURITY

FEATURES

- Authentication
- Cross Site Request Forgery (CSRF)
- Security HTTP Response Headers
- HTTP

REFERENCIAS:

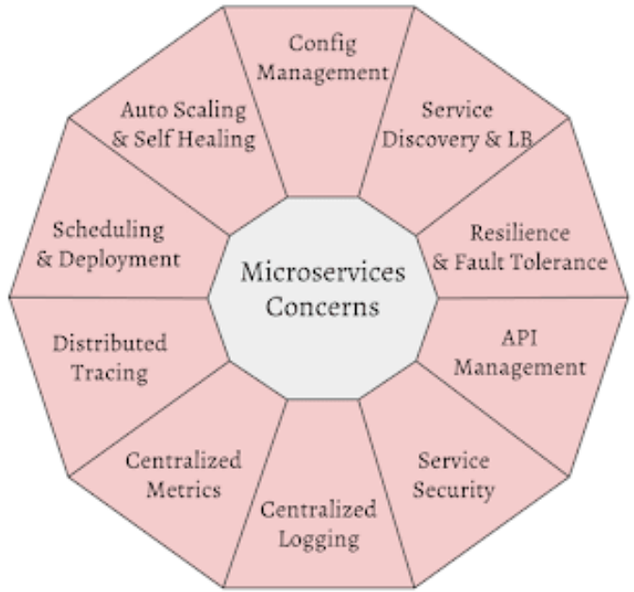
- https://en.wikipedia.org/wiki/Java_logging_framework
- https://docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.ht features-logging
- https://spring.io/guides/gs/scheduling-tasks/
- https://docs.spring.io/spring-framework/docs/current/reference/html/integration.html#

REFERENCIAS:

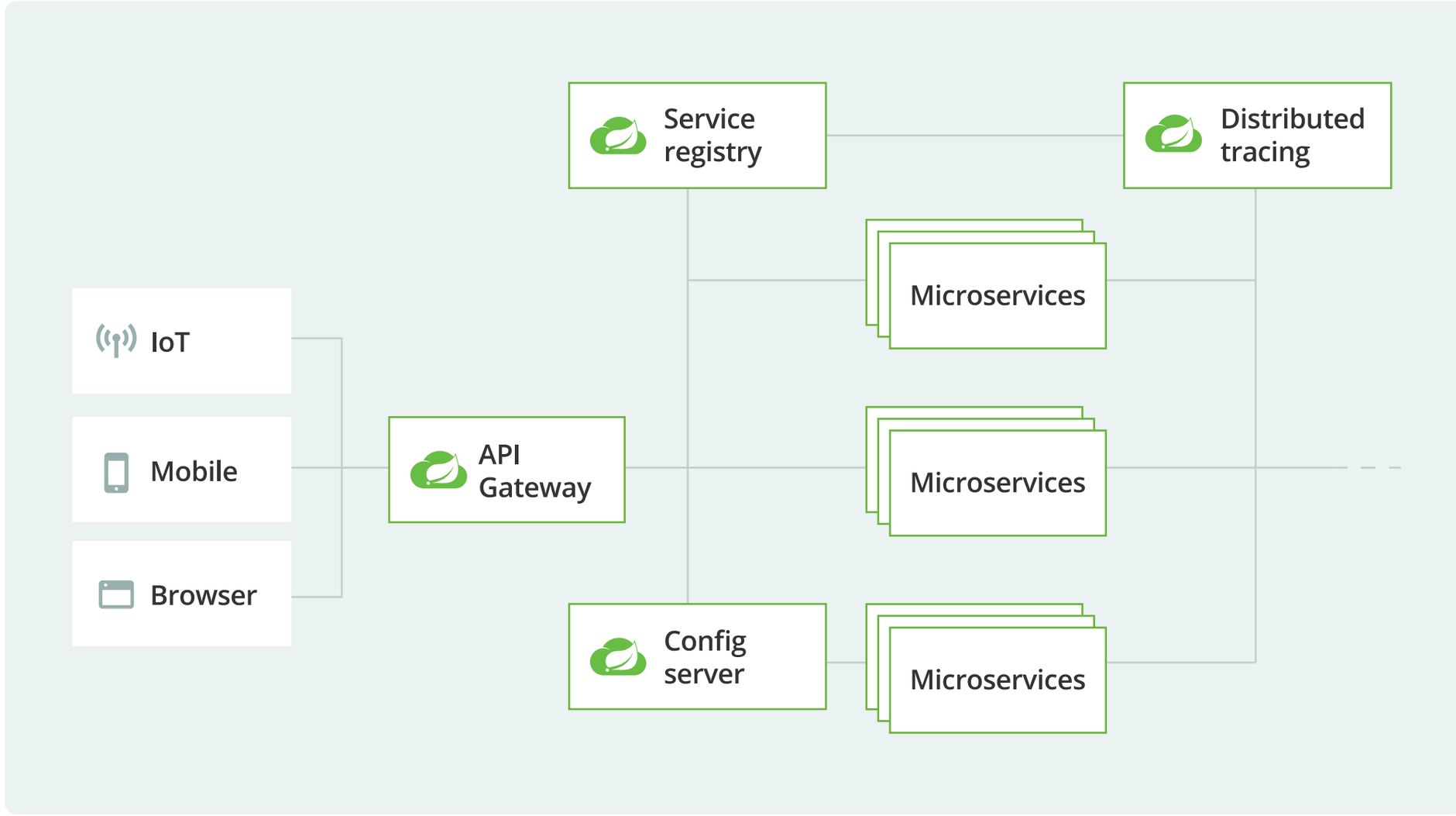
- https://docs.spring.io/spring-framework/docs/current/reference/html/integration.html#annotation-support-scheduled
- https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-featu
- https://docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.ht features-security

SESSIÓN 5

ECOSISTEMA DE MICROSERVICIOS



ARQUITECTURA CLOUD NETFLIX

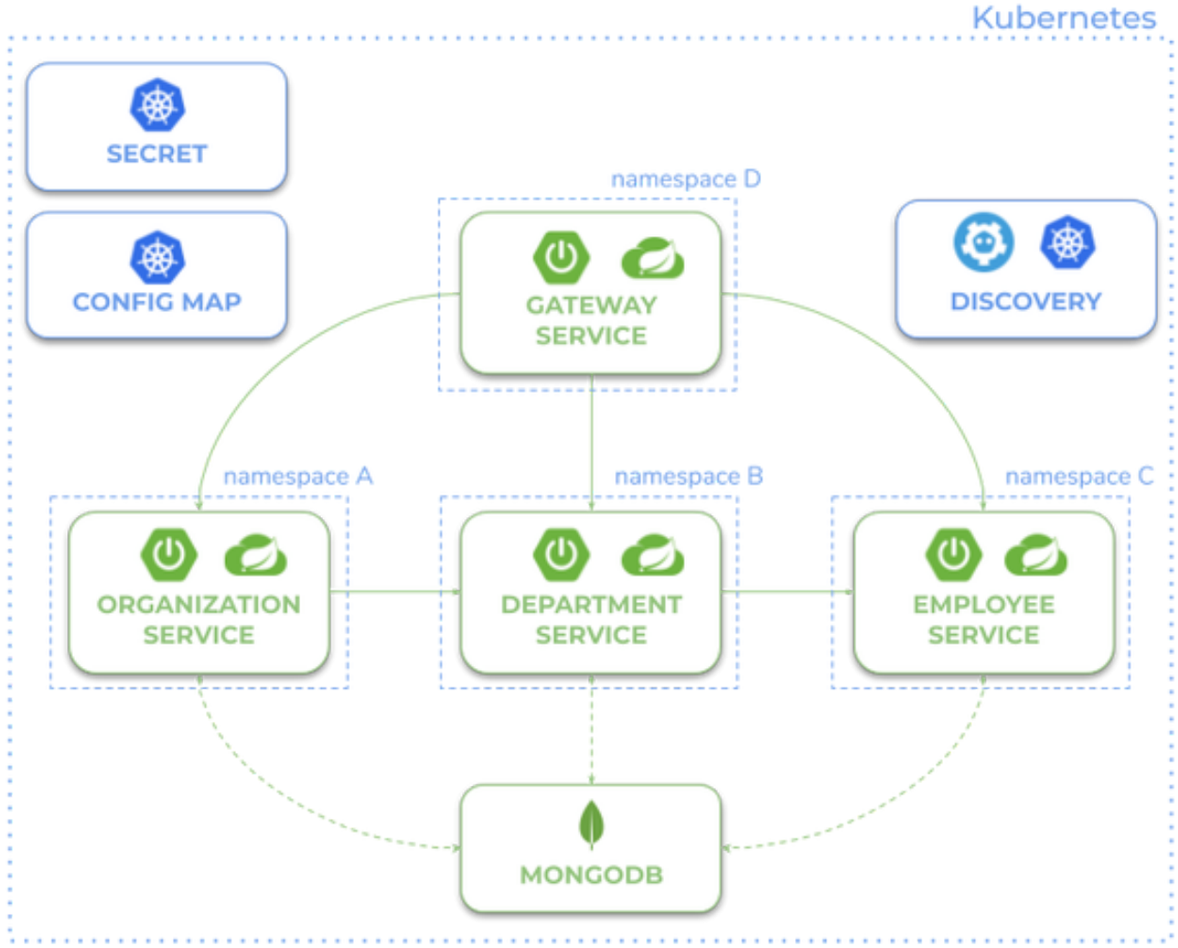


ARQUITECTURA CLOUD NETFLIX

COMPONENTES

- Gateway: Zuul
- Service Discovery: Eureka
- External Configuration: Spring Cloud Config
- Load Balancing: Ribbon

ARQUITECTURA CLOUD KUBERNETES



ARQUITECTURA CLOUD KUBERNETES

COMPONENTES

- Gateway: Spring Cloud Gateway / Ingress
- Service Discovery: K8S
- External Configuration: Config Maps
- Load Balancing: K8S

SERVICIOS AUXILIARES

- Central Logging
- Monitoring & Alerting
- Distributed Tracing

SERVICIOS AUXILIARES

CENTRAL LOGGING (EFK)

Se necesita una solucion para centralizar el Logging de todos los microservicios.

SERVICIOS AUXILIARES

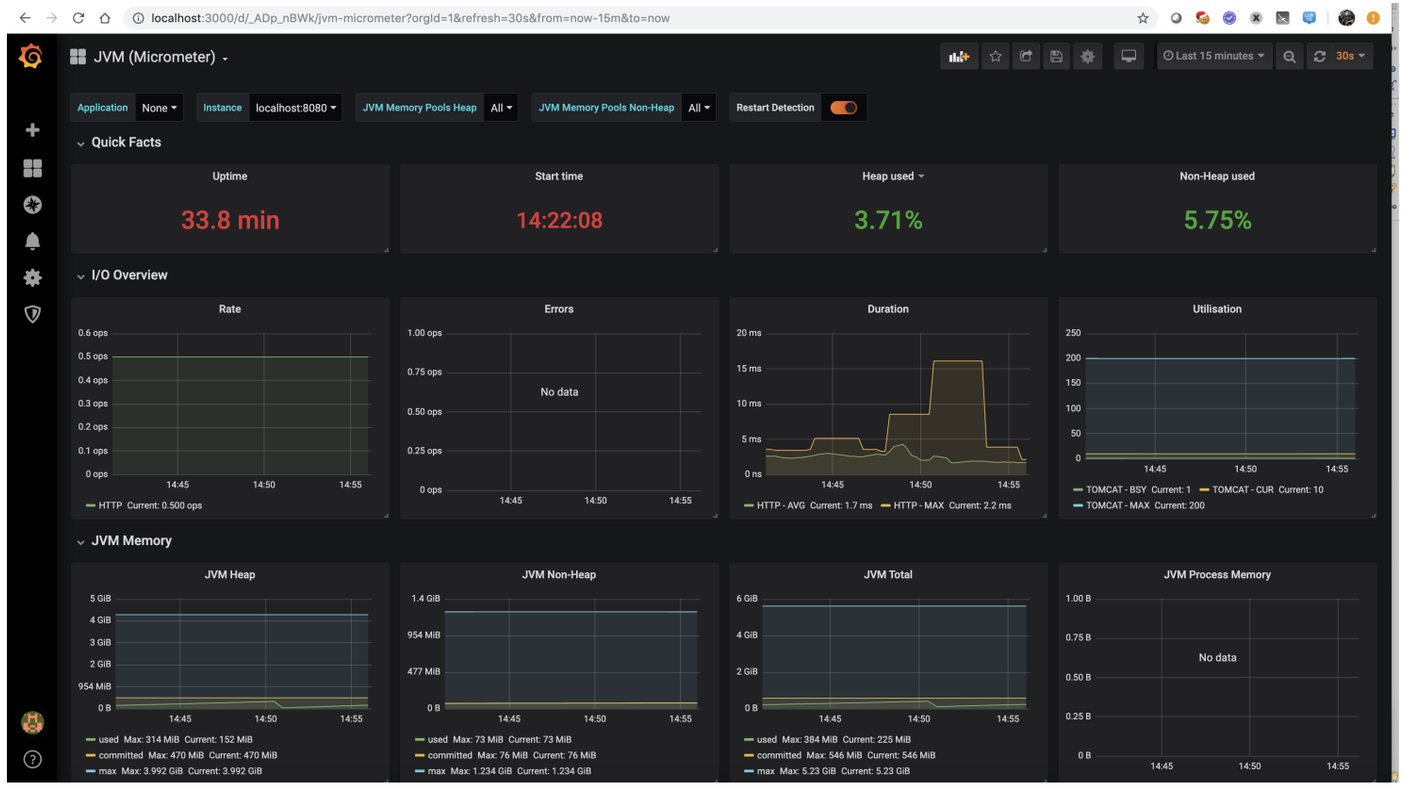
MONITORING & ALERTING

Se necesita una solucion para centralizar la monitorizacion de la salud de los microservicios.

Es necesario monitorizar: CPU, Memoria consumida, Threads, GC, otros...

SERVICIOS AUXILIARES

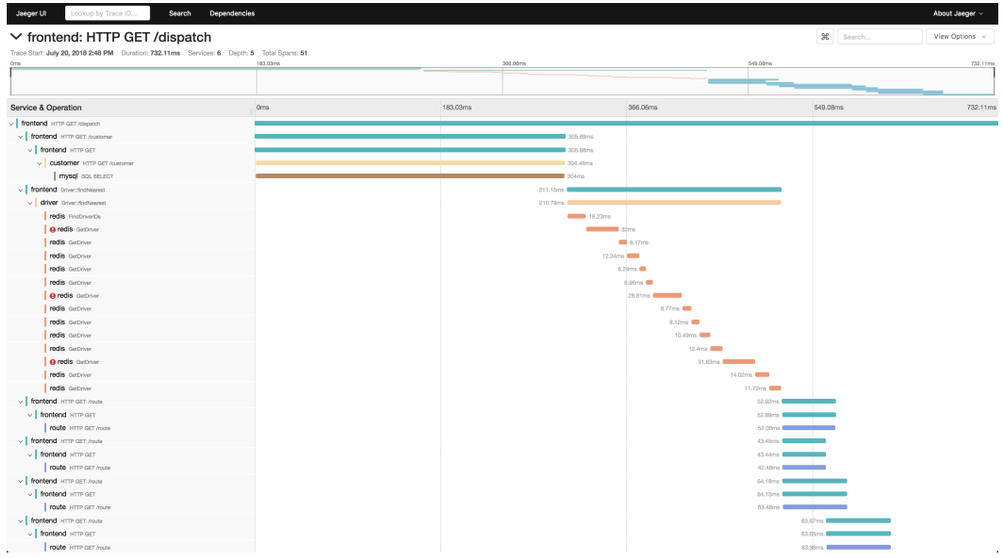
MONITORING & ALERTING



SERVICIOS AUXILIARES

DISTRIBUTED TRACING

Soluciona la necesidad de centralizar la trazabilidad de peticiones a los microservicios



REFERENCIAS

- https://12factor.net
- https://spring.io/cloud
- https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
- https://landscape.cncf.io
- https://github.com/cncf/landscape/blob/master/README.m
  map