[formación por ciclos] Desarrollo de APLICACIONES WEB

Introducción a servicios REST

Diego Iván Oliveros Acosta

@scalapp.co

Agenda

- Historia – contexto
- ¿Qué son microservicios?
- ¿Qué es REST?
- Introducción a los servicios REST
- Microservicios con Spring Boot
- Microservicios con Netflix OSS
- Referencias

Objetivos

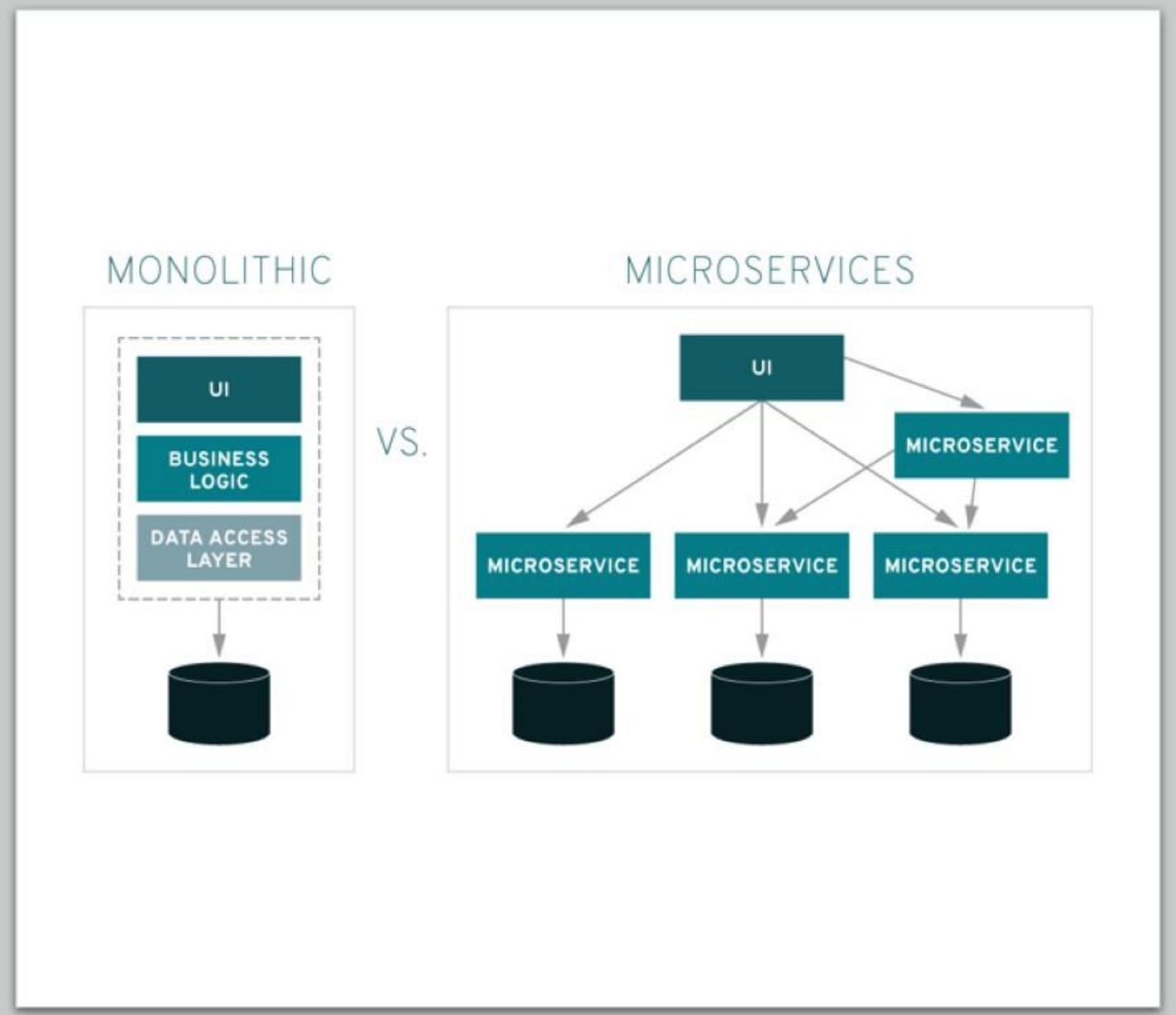
- Construir arquitecturas basadas en microservicios que nos permita atomizar las reglas del negocio.
- Dar más flexibilidad y eficiencia a la hora de implementar y desplegar artefactos funcionales dentro de la arquitectura empresarial corporativa.

Temas a desarrollar (Microservicios)

- Introducción a los servicios REST
- Microservicios con Spring Boot
- Microservicios con Netflix OSS

Vocabulario

- Arquitectura
- Sistema distribuido
- Protocolo
- Framework
- API
- Servicio – micro servicio
- Estándar



El manejo de los **microservicios** en las definiciones de las **nuevas** arquitecturas de *software*

- Ha venido creciendo vertiginosamente en los últimos tiempos y se ha hecho parte del concepto general de los sistemas distribuidos.
- Los sistemas distribuidos utilizan protocolos como REST
- Estos son más livianos y de fácil acceso, para el envío de información, conectando diferentes sistemas e interfaces con diversas arquitecturas en un sinnúmero de lenguajes de programación.

Principales ventajas de un microservicio

- **Arquitectura y lenguaje libre**
- **Se ejecutan de manera independiente y autónoma**
- **Infraestructuras IT más flexibles y adaptables.**

Principales ventajas de un microservicio

- **Multiplataforma:** permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda HTTP
- **Competitivo**: Es increíblemente más simple y convencional que otras alternativas de comunicaciones hacia y desde otras interfaces expuestas por cualquier sistema.
- Podríamos considerar **REST** como un **framework** para construir aplicaciones web respetando **HTTP**.

Desventajas

- Alto consumo de memoria
- Inversión de tiempo inicial
- Complejidad en la gestión
- Perfiles especializados
- Heterogeneidad
- Complejidad en la realización de pruebas
- Costos de implantación o implementación

Servicios REST

- **REST** (representational state transfer) es un estilo de arquitectura de software para sistemas de hipermedia distribuidos como, por ejemplo, la web.

- **REST** se refiere a una colección de principios para el diseño de arquitecturas en red. Estos principios nos dicen cómo son definidos los recursos y cómo son transmitidos desde y hacia cualquier interfaz en un dominio sobre el protocolo HTTP sin capas adicionales.

Servicios REST

- REST no es un estándar, pero se basa en los estándares:
  - HTTP
  - URL
  - XML/HTML/GIF/JPEG
  - MIME:text/html

REST

- **REST** es cualquier interface entre sistemas que usa HTTP.
- **Obtiene datos** o genera operaciones sobre los datos.
- Usa todos los formatos posibles:
  - XML
  - JSON

Stateful vs stateless

- ¿Qué es un estado?
- ¿Cuánto tiempo se esté registrando y cómo se debe almacenar esa información?

- **Apátrida**
- **Con estado**

Principios de REST

- **Escalabilidad** de la interacción con los componentes:
  - Crecimiento exponencial sin degradar el rendimiento
  - Variedad de clientes que pueden acceder
  - Uso en los dispositivos móviles

- **Generalidades de interfaces. Gracias al protocolo HTTP:**
  - Cualquier cliente puede interactuar con cualquier servidor HTTP.
  - No posee ninguna configuración adicional para lograr conexión.

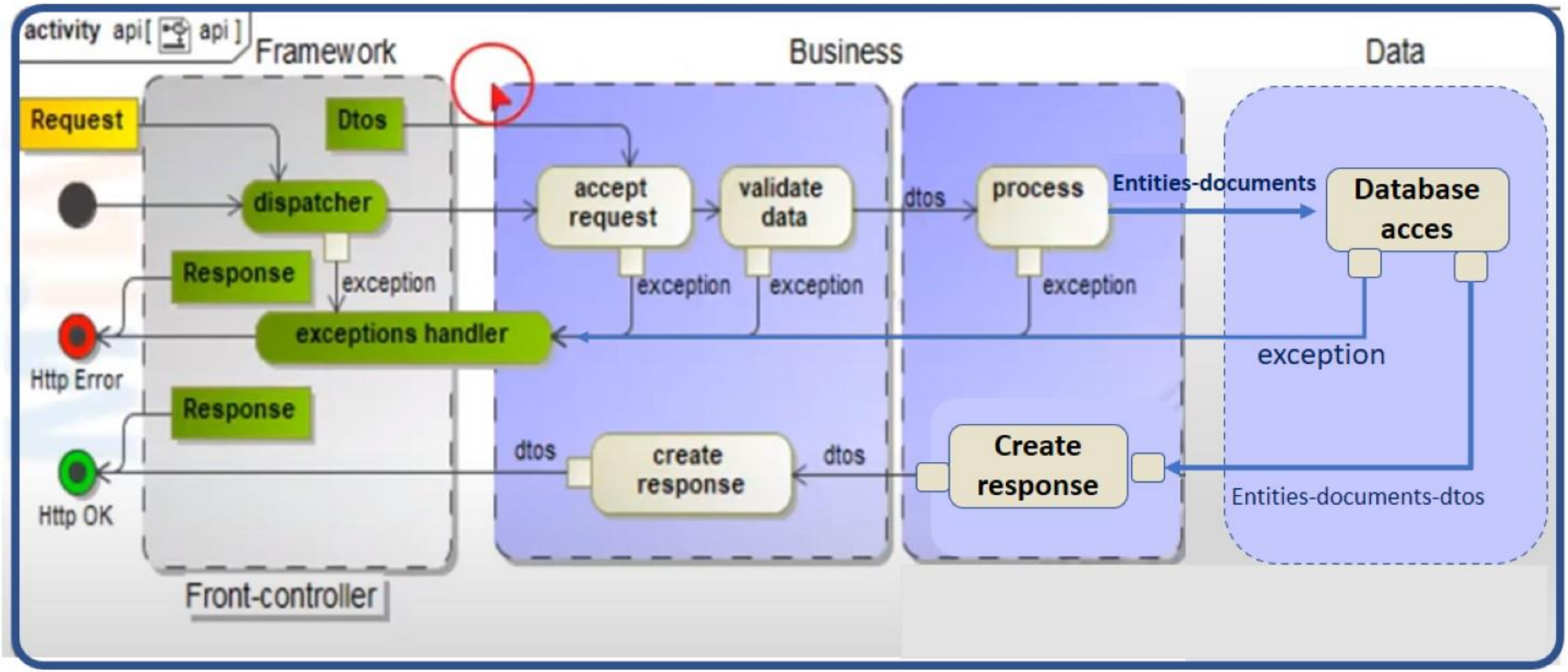
Principios de REST

- **Compatibilidad** con componentes intermedios:
  - Proxy para web
  - Cache, para mejorar el rendimiento
  - Firewalls
  - Gateway

Términos en REST

- URI (uniform resource indentifier) son recursos identificados como colecciones e interfaces que sirven de puntos de acceso e identifican los recursos que queremos operar.
- Ejemplo: **Empleado** (una URI por empleado)
  - TodosLosEmpleados(Listado de los empleados)
- Cada página, archivo, es un recurso al que accedemos o podemos modificar/eliminar

Flujo de datos API REST



Términos en REST

- **URL** (uniform resource locator) es un tipo de URI que, además de identificar de forma única el recurso, nos permite localizarlo para acceder a él.
- Estructura URL:
  - {protocolo}://{dominio o hostname}[:puerto (opcional)]/{ruta del recurso}?{consulta de filtrado}

- Ejemplo URL:
  - https://cursomintic.com:101/2021/conceptos-sobre-apis-rest/

Términos en REST

- **Formatos**. Son una representación como un documento HTML, XML, una imagen, etc.
- Ejemplo de un formato de representación en XML para **Empleado**:
  - <employee xmlns='HTTP://example.org/my-example-ns/'>
  - <name>Full name goes here.</name>
  - <title>Persons title goes here.</title>
  - <phone-number>Phone number goes here.</phone-number>
  - </employee>

Acciones REST

- Los métodos o acciones están diseñadas para operar con datos atómicos dentro de una transacción similar a las operaciones CRUD.
- REST se diseña alrededor de transferencias atómicas de un estado más complejo.
- Es la transferencia de un documento estructurado de una aplicación a otra

Acción	HTTP	SQL	Copy&Paste	Unix Shell
Create	PUT	Insert	Pegar	>
Read	GET	Select	Copiar	<
Update	POST	Update	Pegar después	>>
Delete	DELETE	Delete	Cortar	Del/rm

Métodos soportados en REST

- Cuando hablamos de los **métodos** o acciones, es la forma en que se van a referenciar los recursos (URI) para ser accedidos de diferentes maneras.
- Ejemplo:

HTTP	CRUD	Descripción
POST	CREATE	Crear un nuevo recurso
GET	RETRIEVE	Obtener la representación de una recurso
PUT	UPDATE	Actualizar un recurso
DELETE	DELETE	Eliminar un recurso

Código de estado devueltos en REST

- Son los códigos de estado del recurso que podrían ser devueltos con cualquier método o acción enviado desde el cliente.

- Los códigos son respuestas estandarizadas para informar al cliente sobre el resultado de una solicitud o acción.

2 - 4 - 5

Código de estado devueltos en REST

Códigos de estado para la finalización satisfactoria de la solicitud REST

Código de estado	Descripción
200 Correcto	La solicitud se ha completado satisfactoriamente.
201 Creado	La solicitud se ha completado satisfactoriamente; se ha creado un recurso nuevo.
204 Sin contenido	La solicitud se ha completado satisfactoriamente pero el contenido no está disponible

Código de estado devueltos en REST

Códigos de estado para situaciones de error esperado

Código de estado	Descripción
400 Solicitud incorrecta	La solicitud REST contiene parámetros que no son válidos o que están ausentes.
401 No autorizado	El interlocutor no está autorizado a realizar la solicitud.
403 Prohibido	El interlocutor no está autorizado a completar la solicitud.
404 No encontrado	El recurso solicitado no existe.
406 No aceptable	Se ha solicitado un tipo de contenido o de codificación de contenido no soportado.
409 Conflicto	Existe un conflicto con el estado actual del recurso. La acción solicitada no se puede llevar a cabo en el recurso en su estado actual.
415 Tipo de soporte no soportado	La solicitud contiene un tipo de contenido o una codificación de contenido desconocidos.

Código de estado devueltos en REST

Códigos de estado para errores inesperados

Código de estado	Descripción
500 Error interno del servidor	Se ha producido un problema; se proporciona más información en el rastreo de la pila.
501 No implementado	La solicitud no tiene soporte en la API de REST de IBM® Business Process Manager
503 Servicio no disponible	Las solicitudes federadas no se han podido entregar a todos los destinos de la federación.
504 Tiempo de espera de pasarela	Una respuesta federada no está completa ya que faltan respuestas de algunos de los destinos de la federación.

Referencias

- https://lms.misiontic2022udea.com/course/view.php?id=531
- www.scalapp.co
- https://www.redhat.com/en/topics/api/what-is-a-rest-api
- https://www.ibm.com/docs/es/bpm/8.5.6?topic=apis-status-codes
- https://datatracker.ietf.org/doc/html/rfc6838
- https://www.iana.org/assignments/media-types/media-types.xhtml
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types
- Máster en Ingeniería Web. Universidad Politécnica de Madrid.
- Arquitectura y Patrones para Aplicaciones Web.
- Capítulo 14: Arquitectura de capa de negocio. API Rest con Spring