

¿Qué es un servicio REST?

Desarrollo de un API REST con Spring Boot

Arquitectura de aplicaciones empresariales

- Independientemente de la tecnología, durante muchos años se ha tendido a la construcción de monolitos

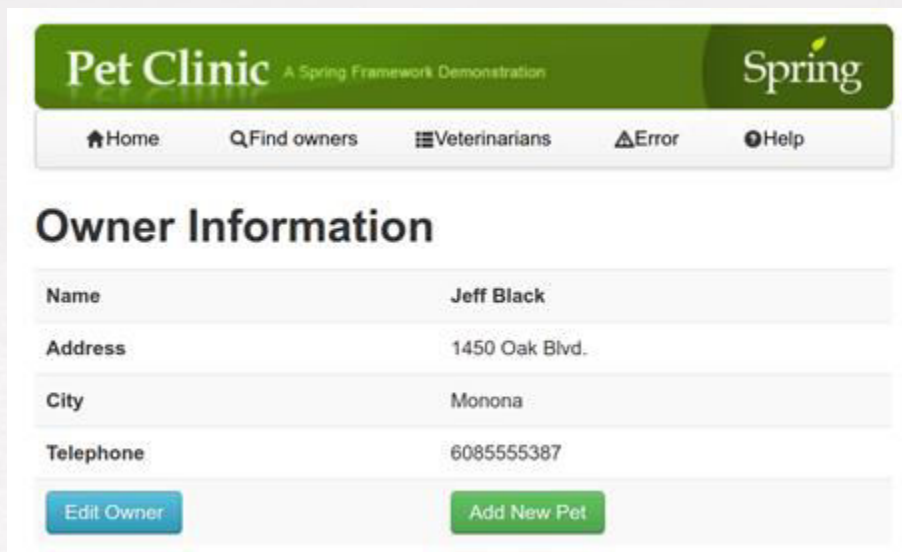


Aplicación *monolítica*

- Ventajas
 - *Uniformidad en el desarrollo*. Todo en un solo lenguaje de programación (o un conjunto pequeño si es una aplicación web).
 - *Despliegue más sencillo* (incluso con arquitecturas replicadas, balanceadas, ...).
 - *Fácil* para proyectos pequeños o al inicio de proyectos grandes.

Ejemplo aplicación *monolítica*

- <https://github.com/thymeleaf/thymeleafexamples-petclinic>
- Clásico ejemplo pet clinic (Spring + Thymeleaf)



The screenshot displays the 'Pet Clinic' application, a Spring Framework demonstration. The header is green with the 'Pet Clinic' logo and the Spring logo. Below the header is a navigation bar with links: Home, Find owners, Veterinarians, Error, and Help. The main content area is titled 'Owner Information' and shows a form with the following data:

Name	Jeff Black
Address	1450 Oak Blvd.
City	Monona
Telephone	6085555387

At the bottom of the form are two buttons: 'Edit Owner' (blue) and 'Add New Pet' (green).

Aplicación *monolítica*

- Desventajas
 - *Escenarios heterogéneos*. ¿Qué pasa si quiero integrar mi proyecto con aplicaciones móviles nativas? Desacoplamiento de la *vista*.
 - Integración proyectos *legacy*. ¿La elección tecnológica marcará definitivamente un proyecto de largo recorrido?
 - *Problemas de escalado*. Si mi aplicación debe escalar, debe hacerlo por completo (vs *microservicios*).

Servicios distribuidos

- Concepto no nuevo (COM, CORBA, RPC, RMI, SOAP).
- Intento de desacoplar algunos elementos de nuestra aplicación (lógica de negocio).
- Todos ofrecen algunas dificultades comunes
 - Interfaces frágiles y difícilmente actualizables
 - Tecnologías propietarias
 - Problemas de comunicación (incluso a nivel de red)

El auge de la web

- Pasó a ser la plataforma *preferida* para el desarrollo de mucho software (como comercio electrónico).
- Uso de muchos pequeños estándares (RFC)
- Aporte de gran flexibilidad (con respecto a los cambios)
- Caldo de cultivo para el surgimiento de REST (motivado por el protocolo HTTP).

REST

- Nuevo enfoque propuesto por Roy Fielding en su tesis doctoral.
- *Características*
 - Basado en el protocolo HTTP
 - Sin estados
 - Representados por una URI
 - Interfaz uniforme
 - Sistema de capas

REST vs RPC

- RPC (remote procedure call) orientado a ofrecer una funcionalidad, un servicio.
- REST orientado a ofrecer recursos.

RPC

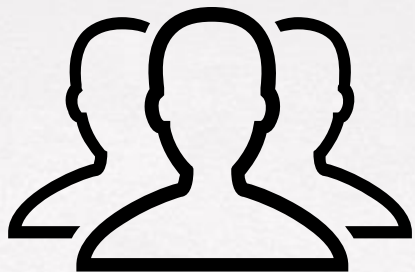
/myapi/beerService/getAll
/myapi/beerService/getById

REST

/myapi/beers
/myapi/beer/123

Recursos vs. Representación

- REST está orientado al concepto de recurso
- Cada recurso debe ser accesible a través de una URI
- El servidor puede ofrecer diferentes representaciones de un mismo recurso (por ejemplo en XML, JSON o HTML).



Representación 1:
JSON

Representación 2:
XML

Recurso
(URI)

Ventajas del uso de REST

- Separación cliente - servidor
- Visibilidad, fiabilidad y escalabilidad
- Heterogeneidad
- Variedad de formatos: JSON, XML, ...
- En general, es más rápido y utiliza menos ancho de banda.

REST en un gráfico

