MAINT

Microservicios

Día 5

MAINT

MongoDB: qué es, cómo funciona y cuándo podemos usarlo (o no)

- Dentro de las bases de datos NoSQL, probablemente una de las más famosas sea **MongoDB**. Con un concepto muy diferente al de las bases de datos relacionales, se está convirtiendo en una interesante alternativa.

- Pero cuándo uno se inicia en MongoDB se puede sentir perdido. No tenemos tablas, no tenemos registros y lo que es más importante, no tenemos SQL. Aun así, MongoDB es una seria candidata para almacenar los datos de nuestras aplicaciones.

¿Qué es MongoDB?

- **MongoDB** es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

- Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que **no es necesario seguir un esquema**. Los documentos de una misma colección - concepto similar a una tabla de una base de datos relacional -, pueden tener esquemas diferentes.

¿Qué es MongoDB?

- Imaginemos que tenemos una colección a la que llamamos Personas.

- Un documento podría almacenarse de la siguiente manera:

```
{
  Nombre: "Pedro",
  Apellidos: "Martínez Campo",
  Edad: 22,
  Aficiones: ["fútbol","tenis","ciclismo"],
  Amigos: [
  {
    Nombre:"María",
    Edad:22
  },
  {
    Nombre:"Luis",
    Edad:28
  }
  ]
}
```

¿Qué es MongoDB?

- El documento anterior es un clásico documento JSON. Tiene strings, arrays, subdocumentos y números. En la misma colección podríamos guardar un documento como este:

```
{
  Nombre: "Luis",
  Estudios: "Administración y Dirección de Empresas",
  Amigos:12
}
```

¿Qué es MongoDB?

- Este documento no sigue el mismo esquema que el primero. Tiene menos campos, algún campo nuevo que no existe en el documento anterior e incluso un campo de distinto tipo.

- Esto que es algo impensable en una base de datos relacional, es algo totalmente válido en **MongoDB**.

¿Cómo funciona MongoDB?

- MongoDB está escrito en C++, aunque las consultas se hacen pasando objetos JSON como parámetro. Es algo bastante lógico, dado que los propios documentos se almacenan en BSON. Por ejemplo:

```
db.Clientes.find({Nombre:"Pedro"});
```

- La consulta anterior buscará todos los clientes cuyo nombre sea Pedro.

¿Cómo funciona MongoDB?

- MongoDB está escrito en C++, aunque las consultas se hacen pasando objetos JSON como parámetro. Es algo bastante lógico, dado que los propios documentos se almacenan en BSON. Por ejemplo:

```
db.Clientes.find({Nombre:"Pedro"});
```

- La consulta anterior buscará todos los clientes cuyo nombre sea Pedro.

¿Cómo funciona MongoDB?

- MongoDB viene de serie con una consola desde la que podemos ejecutar los distintos comandos. Esta consola está construida sobre JavaScript, por lo que las consultas se realizan utilizando ese lenguaje. Además de las funciones de MongoDB, podemos utilizar muchas de las funciones propias de JavaSciprt. En la consola también podemos definir variables, funciones o utilizar bucles.

¿Cómo funciona MongoDB?

- Si queremos usar nuestro lenguaje de programación favorito, existen drivers para un gran número de ellos. Hay drivers oficiales para C#, Java, Node.js, PHP, Python, Ruby, C, C++, Perl o Scala. Aunque estos drivers están soportados por MongoDB, no todos están en el mismo estado de madurez. Por ejemplo el de C es una versión alpha. Si queremos utilizar un lenguaje concreto, es mejor revisar los drivers disponibles para comprobar si son adecuados para un entorno de producción.

¿Dónde se puede utilizar MongoDB?

- Aunque se suele decir que las bases de datos NoSQL tienen un ámbito de aplicación reducido, **MongoDB se puede utilizar en muchos de los proyectos que desarrollamos en la actualidad**.

- Cualquier aplicación que necesite almacenar datos semi estructurados puede usar **MongoDB**. Es el caso de las típicas aplicaciones CRUD o de muchos de los desarrollos web actuales.

¿Dónde se puede utilizar MongoDB?

- Eso sí, aunque las colecciones de **MongoDB** no necesitan definir une esquema, es importante que diseñemos nuestra aplicación para seguir uno.

- Tendremos que pensar si necesitamos normalizar los datos, denormalizarlos o utilizar una aproximación híbrida. Estas decisiones pueden afectar al rendimiento de nuestra aplicación. En definitiva el esquema lo definen las consultas que vayamos a realizar con más frecuencia.

¿Dónde se puede utilizar MongoDB?

- MongoDB es especialmente útil en entornos que requieran escalabilidad. Con sus opciones de replicación y sharding, que son muy sencillas de configurar, podemos conseguir un sistema que escale horizontalmente sin demasiados problemas.

¿Dónde no se debe usar MongoDB?

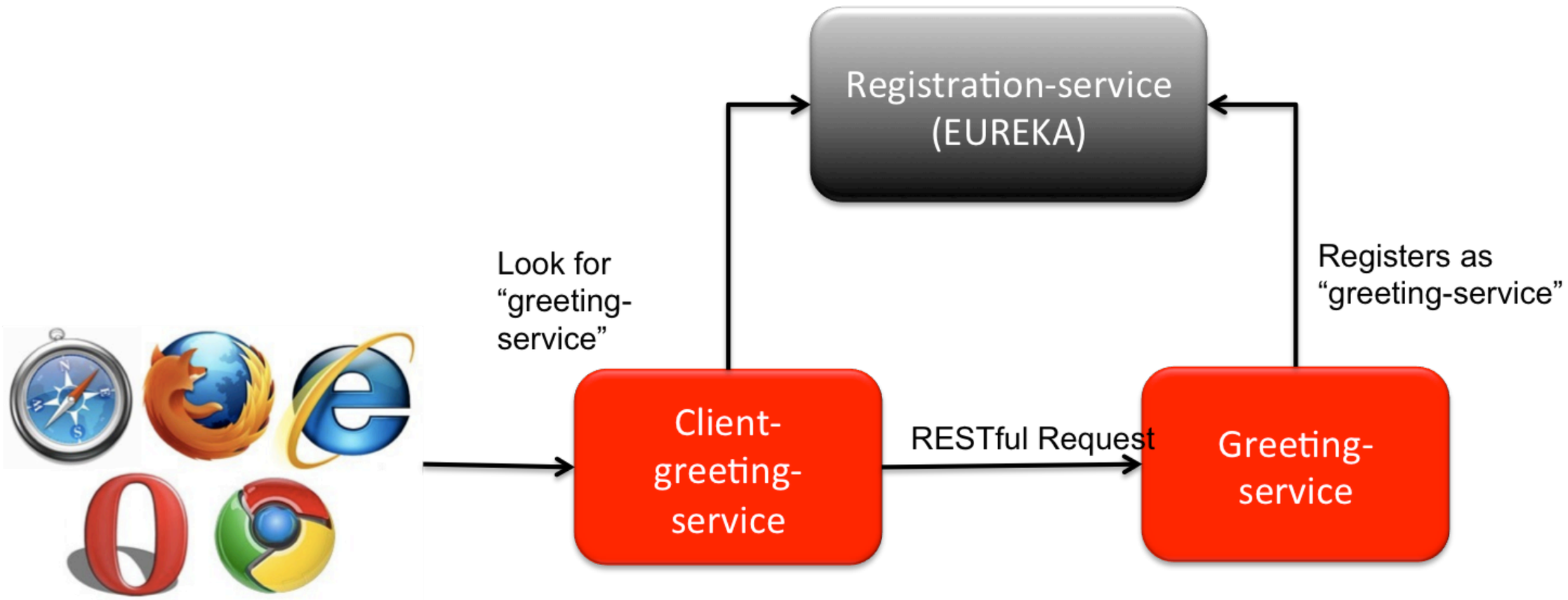
- En esta base de datos **no existen las transacciones**. Aunque nuestra aplicación puede utilizar alguna técnica para simular las transacciones, **MongoDB** no tiene esta capacidad. Solo garantiza operaciones atómicas a nivel de documento. Si las transacciones son algo indispensable en nuestro desarrollo, deberemos pensar en otro sistema.

¿Dónde no se debe usar MongoDB?

- **Tampoco existen los JOINS**. Para consultar datos relacionados en dos o más colecciones, tenemos que hacer más de una consulta. En general, si nuestros datos pueden ser estructurados en tablas, y necesitamos las relaciones, es mejor que optemos por un RDBMS clásico.

¿Dónde no se debe usar MongoDB?

- Y para finalizar, están las consultas de agregación. **MongoDB** tiene un framework para realizar consultas de este tipo llamado Aggregation Framework. También puede usar Map Reduce. Aún así, estos métodos no llegan a la potencia de un sistema relacional. Si vamos a necesitar explotar informes complejos, deberemos pensar en utilizar otro sistema. Eso sí, esta es una brecha que MongoDB va recortando con cada versión. En poco tiempo esto podría dejar de ser un problema.



Spring Data MongoDB: Acceso a datos – Cómo construir microservicios con Spring Boot

Laboratorio 9

MAINT

Swagger

¿Qué pasa cuando dos personas no hablan un mismo idioma?

Las APIs no comparten un idioma común en demasiadas ocasiones y Swagger y Swagger UI trabajan para solucionarlo.

Las APIs nos permiten conectar y compartir datos, algo esencial para las empresas más vanguardistas.

MAINT.

Swagger

Pero surge un gran problema, las APIs no cuentan con un estándar de diseño común y ni si quiera existen unos parámetros comunes para documentarlas.

Es decir, no solo no hablan el mismo idioma, sino que no hay un diccionario que nos ayude a descifrarlas.

¿Tiene sentido en una plataforma cuyo objetivo es que colaboren diversas personas para conseguir los mejores desarrollos?

MAINT

“Una API pierde su sentido sino es accesible y si no tenemos una documentación que nos ayude a entenderla.”

Swagger

Una API definitivamente pierde su sentido sino es accesible y si no tenemos una documentación que nos ayude a entenderla.

¿Cómo van a ser accesibles?

Uno de los mayores problemas de las APIs es que en muchos casos, la documentación que les acompaña es inútil.

Swagger nace con la intención de solucionar este problema. Su objetivo es estandarizar el vocabulario que utilizan las APIs. Es el diccionario API.

Swagger

Cuando hablamos de Swagger nos referimos a una serie de reglas, especificaciones y herramientas que nos ayudan a documentar nuestras APIs.

De esta manera, podemos realizar documentación que sea realmente útil para las personas que la necesitan. Swagger nos ayuda a crear documentación que todo el mundo entienda.

MAINT

“Swagger es una serie de reglas, especificaciones y herramientas que nos ayudan a documentar nuestras APIs.”

Swagger

Aunque existen otras plataformas que ofrecen frameworks diferentes, Swagger es la más extendida.

¿Por qué? Porque ofrece otros beneficios.

El principal de ellos es que todo el mundo entiende Swagger, o casi todos, seas desarrollador o tengas pocos conocimientos sobre desarrollo podrás entenderla gracias al Swagger UI. Incluso las máquinas pueden leerlo, convirtiéndose en otra ventaja muy atractiva.

Swagger

Con Swagger la documentación puede utilizarse directamente para automatizar procesos dependientes de APIs.

Otra de las ventajas, es que lo podemos encontrar integrado en herramientas populares y potentes.

Por último, tampoco nos podemos olvidar de las herramientas que ofrece.

Swagger UI – La interfaz de usuario de Swagger

Swagger UI es una de las herramientas atractivas de la plataforma.

Para que una documentación sea útil necesitaremos que sea navegable y que esté perfectamente organizada para un fácil acceso. Por esta razón, realizar una buena documentación puede ser realmente tedioso y consumir mucho tiempo a los desarrolladores.

MAINT.

“Utilizando el Swagger UI para exponer la documentación de nuestra API, podemos ahorrarnos mucho tiempo.”

Swagger UI – La interfaz de usuario de Swagger

Con el Swagger UI podremos organizar nuestros métodos e incluso poner los ejemplos que necesitemos.

Swagger UI utiliza un documento JSON o YAML existente y lo hace interactivo.

Crea una plataforma que ordena cada uno de nuestros métodos (get, put, post, delete) y categoriza nuestras operaciones. Cada uno de los métodos es expandible, y en ellos podemos encontrar un listado completo de los parámetros con sus respectivos ejemplos. Incluso podemos probar valores de llamada.

MAINT

Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key `special-key` to test the authorization filters.

Find out more about Swagger

http://swagger.io
Contact the developer
Apache 2.0

pet : Everything about your Pets

Show/Hide | List Operations | Expand Operations

POST /pet — Add a new pet to the store

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	(required)	**Pet object that needs to be added to the store**	body	Model | Example Value

Parameter content type: application/json

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "doggie",
  "photoUrls": [
    "string"
  ],
  "tags": [
```

Response Messages

HTTP Status Code	Reason	Response Model	Headers
405	Invalid input		

Try it out!

PUT /pet — Update an existing pet

GET /pet/findByStatus — Finds Pets by status

GET ~~/pet/findByTags~~ — Finds Pets by tags

DELETE /pet/{petId} — Deletes a pet

GET /pet/{petId} — Find pet by ID

POST /pet/{petId} — Updates a pet in the store with form data

POST /pet/{petId}/uploadImage — uploads an image

store : Access to Petstore orders

Show/Hide | List Operations | Expand Operations

user : Operations about user

Show/Hide | List Operations | Expand Operations

[BASE URL: /v2 , API VERSION: 1.0.0]

VALID {...}

Swagger UI – La interfaz de usuario de Swagger

Esta no es la única herramienta útil con la que cuenta Swagger, “Editor” es otra herramienta muy interesantes que nos ayudará a identificar los errores de nuestras documentación en YAML o JSON.

Simplemente subiendo nuestra documentación a la plataforma la comparará con las especificaciones.

Swagger Editor, no solo identificará los errores que hemos cometido, sino que nos planteará sugerencias y nos dará alternativas para que nuestra documentación sea perfecta.

MAINT.

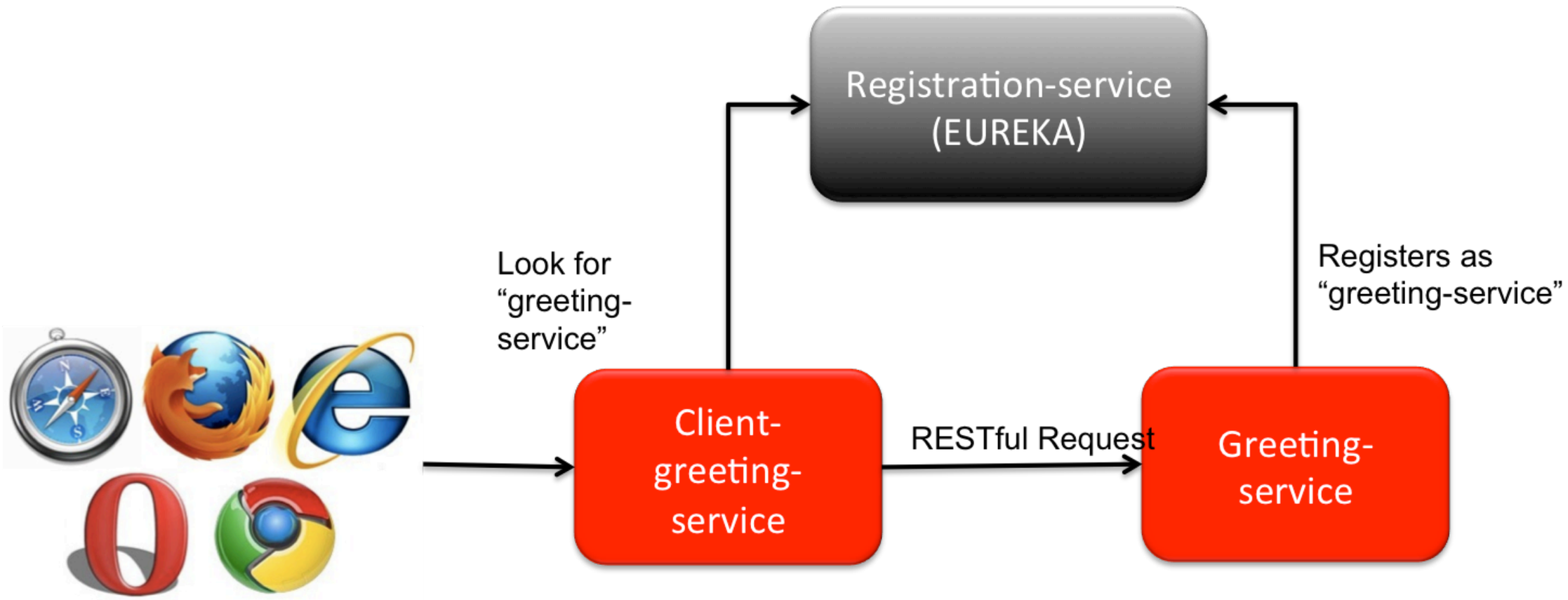
“Una API bien documentada significa que es accesible por otros desarrolladores, y haciendo nuestras APIs más accesibles podremos mejorarlas.”

Swagger UI – La interfaz de usuario de Swagger

Sin duda, Swagger, Swagger UI y todas sus herramientas, son capaces de hacer el trabajo de nuestros desarrolladores mucho más fácil a la hora de documentar nuestras APIs.

Swagger, es un framework tan establecido que hasta está integrado en herramientas tan populares para la gestión de APIs como CA API Manager.

Gracias a su popularidad y sus resultados, Swagger hace posible que cada API tenga el mejor diccionario para entenderla.



Swagger: Documenta APIs REST – Cómo construir microservicios con Spring Boot

Laboratorio 10

MAINT

Gracias