Desafío Mercado Libre.

PortalInmobiliario.com

Documentación

Autor: Christian Adrián Ortiz

Link al Repositorio : https://github.com/Cortiz1692/Challenge.git

Desafío:

Problemática:

Los usuarios de Portalinmobiliario.com necesitan poder ver los puntos de interés (POIs) más cercanos al inmueble que está visitando. En encuestas realizadas previamente, el 51% de los usuarios utilizan la información del entorno para tomar una decisión de comprar o arrendar una propiedad.

Solución propuesta :

Incluir en la página principal de la propiedad la información del entorno, mostrando sólo los puntos de interés que se encuentren en un radio de 2 Km.

Consideraciones generales

● Portal inmobiliario recibe + 2 MM de visitas únicas mensuales y el tráfico es aproximadamente de 5K RPM.

● La información debe mostrarse para todas las categorías disponibles. ● Las categorías de los POIs pueden variar en el futuro, dependiendo del % de adopción de la funcionalidad.

● Cuando se creen nuevos POIs, es necesario que se vean reflejados.

● La información de los POIs se encuentra en una fuente de datos sin ninguna modificación o transformación especial.

Abordaje de la Solución

Objetivos Clave

Para este desafío, es crucial abordar varios aspectos técnicos y operativos para asegurar el éxito y la escalabilidad del proyecto.

1. **Trabajo Reactivo**: Implementar un enfoque reactivo para manejar la alta concurrencia de visitas y asegurar que la aplicación pueda escalar adecuadamente frente a la demanda.
2. **Escalabilidad**: Diseñar la arquitectura de la solución con un enfoque en la escalabilidad, permitiendo la adición futura de nuevas categorías de puntos de interés (POIs) sin afectar el rendimiento del sistema.
3. **Comunicación en Tiempo Real**: Integrar mecanismos de comunicación en tiempo real para mejorar la experiencia del usuario, asegurando que los nuevos puntos de interés se reflejen de inmediato en la interfaz de usuario.
4. **Persistencia Escalable y Confiable**: Adoptar un sistema de persistencia que no solo sea escalable y confiable, sino que también maneje de manera efectiva los datos espaciales, garantizando integridad y rendimiento en las consultas geoespaciales.

Solución Propuesta

Arquitectura Orientada a Microservicios

La arquitectura propuesta para abordar esta problemática es una arquitectura orientada a microservicios. Este enfoque permite una alta capacidad de respuesta y un manejo eficiente de la velocidad, capaz de soportar la alta demanda de visitas en el portal inmobiliario.

Características Clave

1. **Alta Capacidad de Respuesta**:
    - Los microservicios permiten respuestas rápidas a las solicitudes de los usuarios, mejorando la experiencia general.
2. **Escalabilidad**:
    - Cada microservicio puede escalarse de forma independiente según la demanda. Esto asegura que el sistema puede manejar picos de tráfico sin afectar el rendimiento.
3. **Manejo de Concurrencia**:

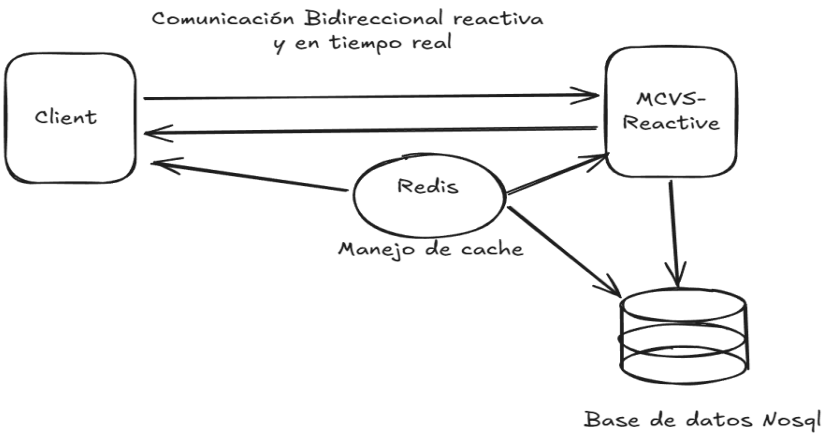
- Utilizando tecnologías como Redis para el caché y MongoDB para el almacenamiento de datos, se asegura una alta concurrencia y manejo eficiente de las consultas geoespaciales.

4. **Comunicación en Tiempo Real**:
    - Integración de WebSockets
    - para actualizar la información de POIs en tiempo real, mejorando la experiencia de usuario al ver nuevos puntos de interés.
5. **El frontend está desarrollado utilizando un enfoque basado en componentes con React, lo que permite una mayor modularidad y reutilización de código, mejorando la mantenibilidad y escalabilidad de la aplicación**

Componentes Principales

1. **Front React**:
    - Maneja la experiencia de usuario y las solicitudes ha los servicios.
2. **Microservicio de Propiedades y POIs**:
    - Servicios independientes que manejan la lógica de negocio relacionada con las propiedades y los puntos de interés.
3. **Base de Datos MongoDB**:
    - Almacena datos geoespaciales de manera eficiente, utilizando índices 2dsphere para mejorar el rendimiento de las consultas.
4. **Sistema de Cache Redis**:
    - Optimiza el rendimiento de las consultas y reduce la carga sobre la base de datos.

Diagrama de Arquitectura



Necesidad de Usar una Base de Datos Espacial (La persistencia).

En este proyecto nos encontramos con los puntos de interés (POIs) que estaban geográficamente distribuidos, por eso se optó por él uso de una base de datos espacial por varias razones:

1. Consultas Geoespaciales Eficientes

Las bases de datos espaciales están optimizadas para realizar consultas que involucran ubicación y distancia, como encontrar todos los POIs dentro de un radio de 2 km de una propiedad. Estas consultas son mucho más rápidas y eficientes en una base de datos espacial comparada con una base de datos tradicional.

2. Indexación Espacial

Las bases de datos espaciales, como MongoDB con sus índices 2dsphere, permiten la creación de índices geoespaciales que mejoran significativamente el rendimiento de las consultas espaciales. Esto es crucial para aplicaciones que necesitan responder rápidamente a las solicitudes de los usuarios.

3. Manejo de Datos Complejos

El manejo de datos espaciales incluye no solo la latitud y longitud, sino también formas geométricas complejas, rutas y áreas. Las bases de datos espaciales están diseñadas para manejar y almacenar este tipo de datos de manera eficiente.

4. Escalabilidad

A medida que se añaden más datos y la demanda de consultas aumenta, las bases de datos espaciales como MongoDB pueden escalar horizontalmente, distribuyendo la carga de trabajo a través de múltiples servidores. Esto asegura que la aplicación pueda manejar grandes volúmenes de datos y tráfico.

5. Experiencia de Usuario Mejorada

Proporcionar información precisa y rápida sobre los POIs más cercanos mejora significativamente la experiencia del usuario, permitiendo decisiones más informadas sobre la compra o el alquiler de propiedades.

Investigación y Comparación de MongoDB y PostgreSQL con PostGIS

Comparación de MongoDB y PostgreSQL con PostGIS

MongoDB:

- **Tipo de Base de Datos**: NoSQL, orientada a documentos.
- **Estructura de Datos**: Utiliza BSON (Binary JSON) para almacenar datos semi-estructurados.

- **Escalabilidad**: Ofrece escalabilidad horizontal mediante sharding.
- **Flexibilidad**: No requiere un esquema fijo, lo que permite una mayor flexibilidad en la estructura de los datos.
- **Rendimiento**: Alta eficiencia en operaciones de lectura y escritura, especialmente con grandes volúmenes de datos.
- **Comunidad y Ecosistema**: Gran comunidad activa y amplia documentación.

PostgreSQL con PostGIS:

- **Tipo de Base de Datos**: Relacional, con extensión espacial PostGIS.
- **Estructura de Datos**: Utiliza tablas con filas y columnas estructuradas.
- **Escalabilidad**: Principalmente escalabilidad vertical, aunque también puede ser escalado horizontalmente con configuraciones avanzadas.
- **Flexibilidad**: Requiere un esquema predefinido, lo que puede limitar la flexibilidad en comparación con MongoDB.
- **Rendimiento**: Excelente para consultas complejas y operaciones espaciales gracias a PostGIS.
- **Comunidad y Ecosistema**: Fuerte comunidad y amplia gama de extensiones y herramientas.
-

Razones para Elegir MongoDB

1. **Flexibilidad de Esquema**: La capacidad de MongoDB para manejar datos semi-estructurados y su esquema flexible es ideal para aplicaciones que requieren adaptabilidad y evolución constante.
2. **Escalabilidad Horizontal**: La facilidad con la que MongoDB puede escalar horizontalmente mediante sharding es crucial para manejar grandes volúmenes de tráfico y datos.
3. **Rendimiento en Operaciones de Escritura**: MongoDB ofrece un rendimiento superior en operaciones de escritura, lo que es beneficioso para aplicaciones que manejan una alta cantidad de inserciones y actualizaciones.
4. **Integración con Tecnologías Modernas**: La integración de MongoDB con tecnologías modernas y su compatibilidad con múltiples plataformas lo hacen una opción robusta para aplicaciones web y móviles.
5. **Manejo de Datos Geoespaciales**: Aunque PostGIS es una opción sólida para operaciones espaciales, MongoDB también ofrece capacidades de

indexación geoespacial que son suficientes para nuestras necesidades actuales.

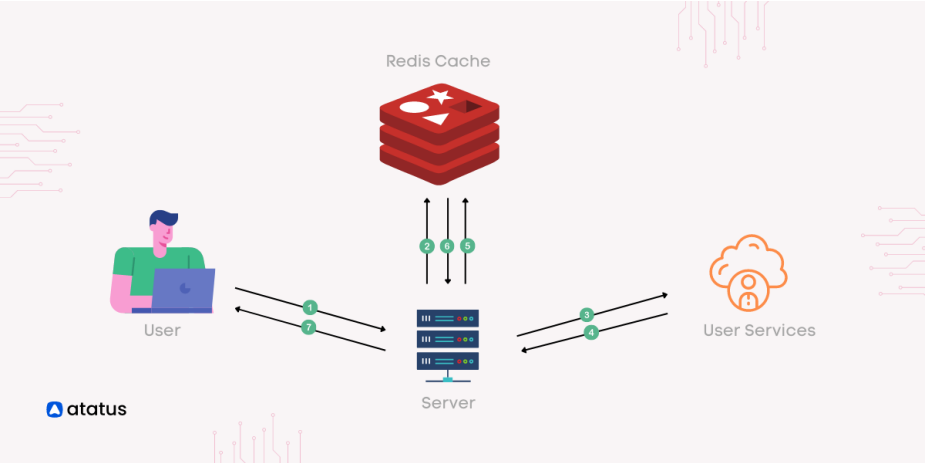
Uso de Redis y Sus Beneficios

¿Por Qué Elegimos Redis?

En este proyecto, he optado por usar Redis para mejorar el rendimiento de las consultas y manejar la alta concurrencia de solicitudes y el cacheo eficiente.

En nuestro proyecto, Redis se utiliza para:

- Almacenar en caché los puntos de interés (POIs) más consultados.
- Reducir la carga de consultas repetidas en MongoDB.



Comunicación en Tiempo Real con WebSockets.

Para el punto de las consideraciones donde se debían reflejar los puntos de interés en le mapa, se evaluaron opciones como Apache Kafka, Server-Sent Events (SSE), y WebSockets, quedándome con la última por su facilidad de integración, amplio soporte y capacidad de comunicación bidireccional a diferencia de (Server-Sent Events).

Microservicio Reactivo con WebFlux

En cuando al manejo de la lógica y endpoints de consulta opte por el uso de

Spring WebFlux. Este enfoque nos brinda varios beneficios, especialmente cuando se trata de manejar la alta concurrencia y ofrecer una experiencia de usuario más fluida.

Beneficios del Enfoque Reactivo

Manejo de Alta Concurrencia:

- Spring WebFlux permite manejar miles de solicitudes concurrentes de manera eficiente, sin bloquear los hilos. Esto es crucial para nuestro proyecto, dado el alto volumen de tráfico en Portalinmobiliario.com.

Rendimiento Mejorado:

- Al ser no bloqueante, WebFlux aprovecha mejor los recursos del servidor, lo que resulta en un rendimiento mejorado y tiempos de respuesta más rápidos.

Además Spring WebFlux se integra de manera nativa con MongoDB, lo que nos permite aprovechar las capacidades reactivos de MongoDB a través de **Reactive MongoRepository**. Esta integración facilita el manejo de datos geoespaciales de manera reactiva y eficiente lo que fue un punto clave a la hora de du elección.

Documentación de la API REST.

API 1: Obtener Todas las Propiedades

Verbo	http
Get	http://localhost:8080/geospatial/properties

API 2: Obtener Todas las Categorías

Verbo	http
Get	http://localhost:8080/geospatial/categories

API 3: Obtener Todos los puntos cercanos a un radio de 2 km.

Verbo	http
Get	http://localhost:8080/geospatial/pois-near-property?latitude= (datos)&longitude=(datos)

Modelo de Objeto Response:


```
[
    {
        "id": "674ac0c7a32a55e08106cbab",
        "name": "Centro de Salud Este",
        "category": "Salud",
        "address": "Calle Salud 1708",
        "location": {
            "type": "Point",
            "coordinates": [
                -58.3838,
                -34.6092
            ]
        }
    },
```


API 4: Crea nuevos puntos de interes

Verbo	http
Post	http://localhost:8080/create/newPois

Request:

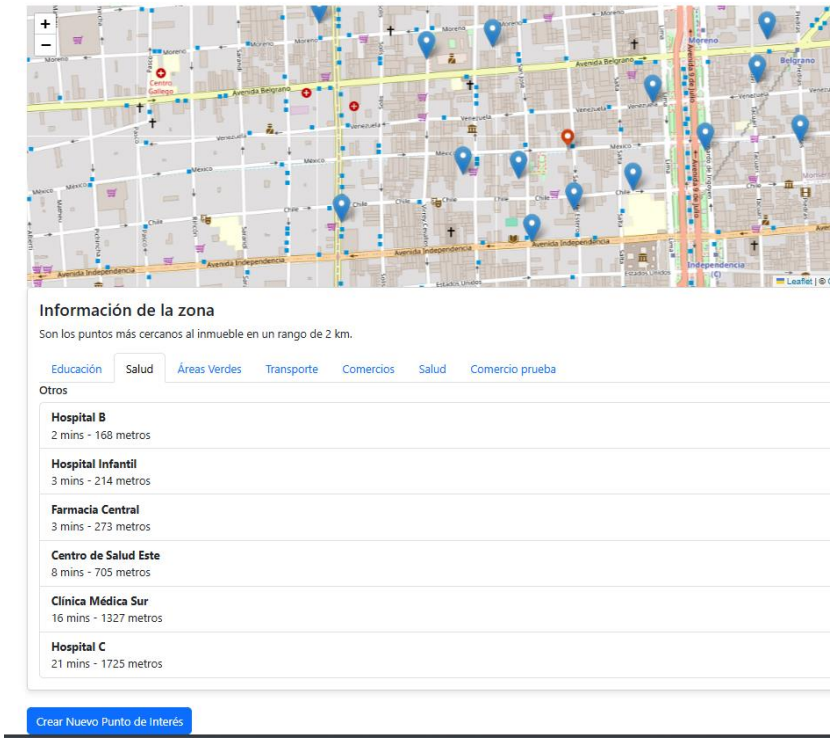

```
{
  "name": "Hospital C ",
  "category": "Salud",
  "address": "Piedras",
  "location": {
    "type": "Point",
    "coordinates": [
      -56.3840,
      -31.6090
    ]
  }
}
```


Validaciones:


```
{
    "error": "El punto de interés ya existe"
}
```

```
[
    "El campo name null",
    "El campo category null",
    "El campo address null",
    "El campo location null"
]
```


Solución Lograda.



Conclusión del Proyecto

El proyecto de implementación de una funcionalidad de geolocalización para mostrar puntos de interés (POIs) cercanos a propiedades en Portalinmobiliario.com ha resultado en una solución robusta y escalable que mejora significativamente la experiencia del usuario. A lo largo de esta documentación, hemos cubierto la arquitectura, los beneficios de las tecnologías

Para , mayor eficiencia escalabilidad y disponibilidad se podrían abordar soluciones como **Kubernetes** es una plataforma de orquestación de contenedores que automatiza la implementación, gestión y escalabilidad de aplicaciones contenedorizadas.

Sistemas Distribuidos

Un sistema distribuido es un modelo en el que componentes de software ubicados en múltiples computadoras interactúan y coordinan sus acciones mediante la red para lograr un objetivo común. Los beneficios incluyen:

- **Escalabilidad Horizontal**: Permite agregar más nodos al sistema para manejar una mayor carga de trabajo sin necesidad de grandes cambios en la arquitectura existente.
- **Desempeño Mejorado**: Distribuir la carga de trabajo entre múltiples nodos puede mejorar significativamente el desempeño, permitiendo procesar más solicitudes de manera eficiente.

Cierre

En resumen, este proyecto ha sido diseñado para ser altamente eficiente y escalable, utilizando tecnologías modernas como MongoDB, Redis, y Spring WebFlux, y considerando la implementación de sistemas distribuidos y Kubernetes para gestionar la alta demanda de tráfico. La solución propuesta no solo mejora la experiencia de usuario en tiempo real, sino que también asegura la capacidad de escalar y adaptarse a futuras necesidades y cambios.