códigofacilito

Modelado de microservicios

Bootcamp - Backend Avanzado

Ulises Navarrete Macías

¿Quién soy?

Temario:

- Definición de un servicio.
  - Qué es un servicio?.
  - Responsabilidades y límites del servicio.
  - Ejercicio.
- DDD (Domain-Driven Design).
  - Principios básicos de Domain-Driven Design.
  - Pros vs Contras.
  - Ejercicio.

Temario:

- Acoplamiento alto vs bajo.
  - Qué es acoplamiento?
  - Importancia en microservicios.
  - Pros y contras del acoplamiento alto.
  - Pros y contras del acoplamiento bajo.
  - Tips para lograr un acoplamiento bajo.
- Cómo descomponer un monolito.
  - Razones para migrar un monolito.
  - Estrategias para descomponer un monolito.
  - Ejercicio | Tarea

Modelado de microservicios

Qué es un servicio?

Un servicio representa una unidad lógica y funcional del sistema, diseñada para abordar una tarea específica o un conjunto de tareas relacionadas.

Responsabilidades y límites del servicio

- Claridad en el propósito del servicio.
- Reducción de la complejidad y sobrecarga de funcionalidades.
- Independencia y cohesión.
- Facilita la escalabilidad y la flexibilidad.
- Mejora la reutilización y la modularidad.

Ejercicio: Modelado de microservicios para un sistema de agencia de viajes

- Link del ejercicio

QA

DDD
Domain-Driven Design

DDD Se centra en comprender el dominio del problema y reflejar esa comprensión en el diseño del software.

Principios básicos de Domain-Driven Design

- Lenguaje ubicuo.
- Contextos delimitados.
- Entidades.
- Objetos de valor.
- Agregados.

Pros vs Contras

- Proceso y software más flexible.
- Claridad de un problema muy complejo.
- Comunicación efectiva entre expertos y desarrolladores.
- Código bien organizado y aislado.

- Aislar la lógica de negocio suele llevar mucho tiempo.
- Necesitamos un experto de dominio.
- Curva de aprendizaje alta.
- Solo es sugerido para aplicaciones complejas, no recomendado para CRUD’s.

Ejercicio: Modelado de microservicios para un sistema de comercio electrónico

- Link del ejercicio

Acoplamiento alto
vs
Acoplamiento bajo

Qué es acoplamiento?

El grado de interdependencia entre los diferentes componentes o módulos de un sistema de software. Indica la fuerza en la que están relacionados o dependen entre sí los distintos elementos del sistema.

Importancia en Microservicios

En el contexto de microservicios, el acoplamiento juega un papel crucial debido a la naturaleza distribuida y modular. Un bajo acoplamiento es fundamental para garantizar la independencia, escalabilidad, flexibilidad y mantenibilidad de los microservicios.

Acoplamiento alto

Pros vs Contras

- Facilita la comunicación directa entre componentes.
- Puede ser más fácil de implementar en sistemas pequeños y simples.

- Aumenta la dependencia entre componentes, lo que hace que el sistema sea más frágil.
- Dificulta la reutilización y la sustitución de componentes.
- Puede dificultar el mantenimiento y la evolución del sistema.

Acoplamiento bajo

Pros vs Contras

- Promueve la independencia y la modularidad de los componentes.
- Facilita la reutilización, sustitución y el despliegue independiente.
- Permite una mayor flexibilidad y adaptabilidad del sistema ante cambios.

- Puede requerir un mayor esfuerzo de diseño y desarrollo inicial.
- Puede introducir una mayor complejidad debido a la necesidad de definir interfaces y comunicación entre componentes.

Tip lograr un acoplamiento bajo

- Definir interfaces claras.
- Segregar la base de datos.
- Implementar patrones de mensajería.
- Aplicar principios de diseño SOLID.
- Evitar el acoplamiento directo.
- Descomponer funcionalidades complejas.

Cómo descomponer un monolito

Razones para migrar un monolito

- Escalabilidad independiente
- Mayor flexibilidad tecnológica
- Despliegue continuo y rápido
- Mejora de la resiliencia
- Facilita la colaboración y la escalabilidad del equipo
- Facilita la evolución del sistema
- Alineación con la arquitectura basada en la nube

Estrategias para descomponer un monolito de manera gradual

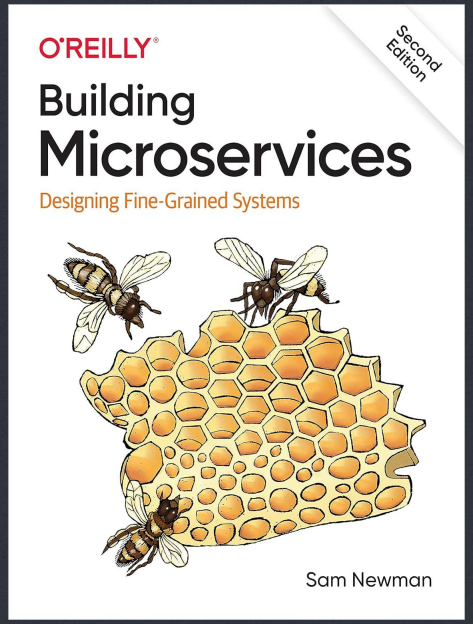
- Identificar dominios contextuales
- Crear capas de servicios
- Utilizar funciones facade o adaptadores
- Aplicar patrones de gateway o proxy
- Separar funcionalidades por módulos o componentes
- Refactorizar gradualmente

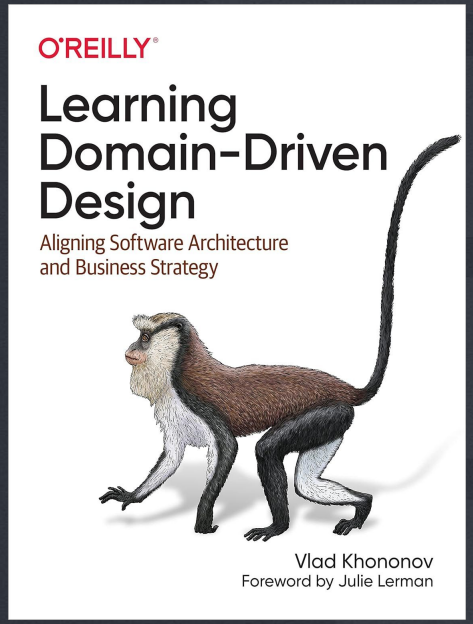
Tarea/Ejercicio: Descomposición de un monolito

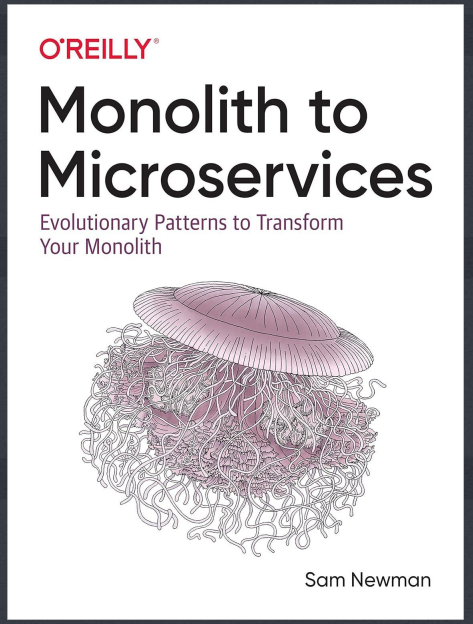
- Link del ejercicio

QA

Recomendaciones







códigofacilito

Modelado de microservicios

Bootcamp - Backend Avanzado

Ulises Navarrete Macías