PATRONES DE ARQUITECTURA DE SOFTWARE

ARQUITECTURA ORIENTADA A MICROSERVICIOS

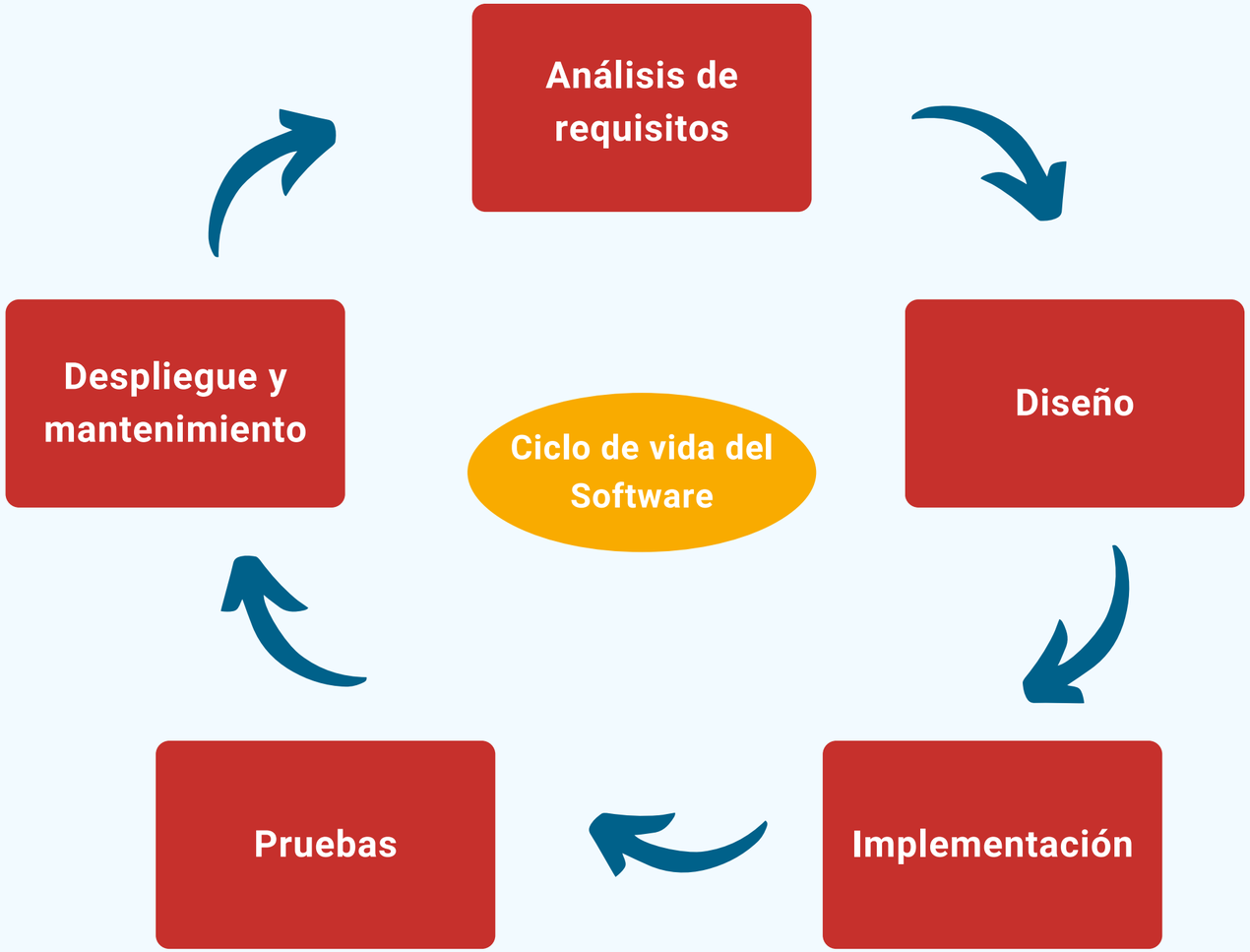
Daniel Blanco Calviño

¿QUÉ SON LOS MICROSERVICIOS?

La arquitectura de microservicios proporciona una serie de prácticas, organización de trabajo y de equipos para **construir software complejo de forma más eficiente, rápida y a gran escala.**

¡No tiene que ver con la tecnología! La arquitectura de microservicios es agnóstica de la tecnología que se use para implementar la solución.

CICLO DE VIDA DEL SOFTWARE



SOLUCIÓN MONOLÍTICA

- **Un único componente para todo el sistema.**
  - Repositorio de código único.
  - Base de Datos única.
  - Despliegue único de todo el sistema.
  - La tecnología se mantiene desde el inicio al fin.

- **Según va creciendo el sistema, aumentan los tiempos en mantenimiento.**
  - Más complicado integrar nuevas funcionalidades al código ya existente.
  - Aparición de más bugs.
  - Mantenimiento más costoso y complicado en general.

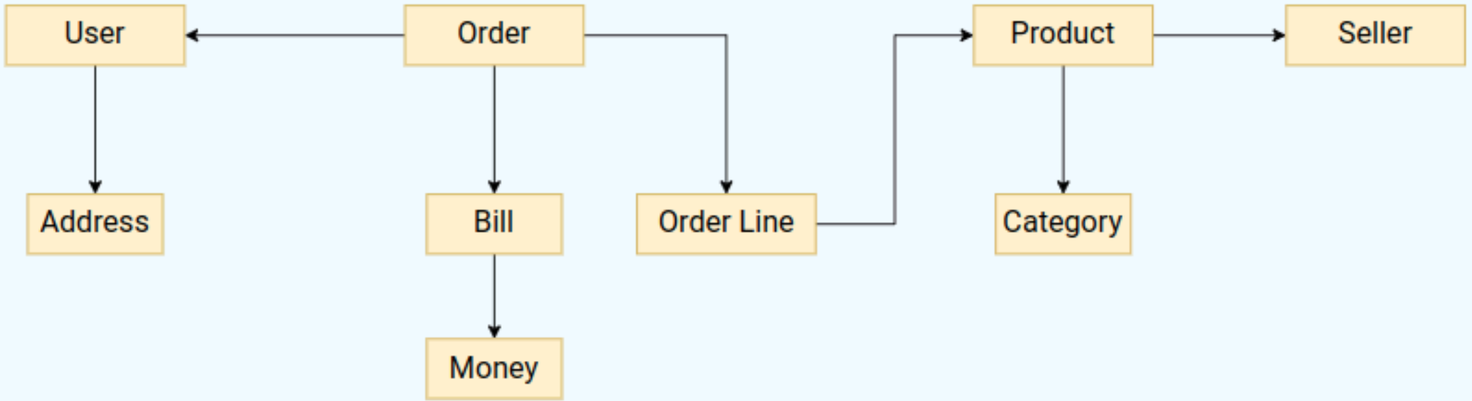
MICROSERVICIOS

- Dividimos el sistema en **contextos cerrados.**
  - Creamos un **microservicio para cada contexto.**

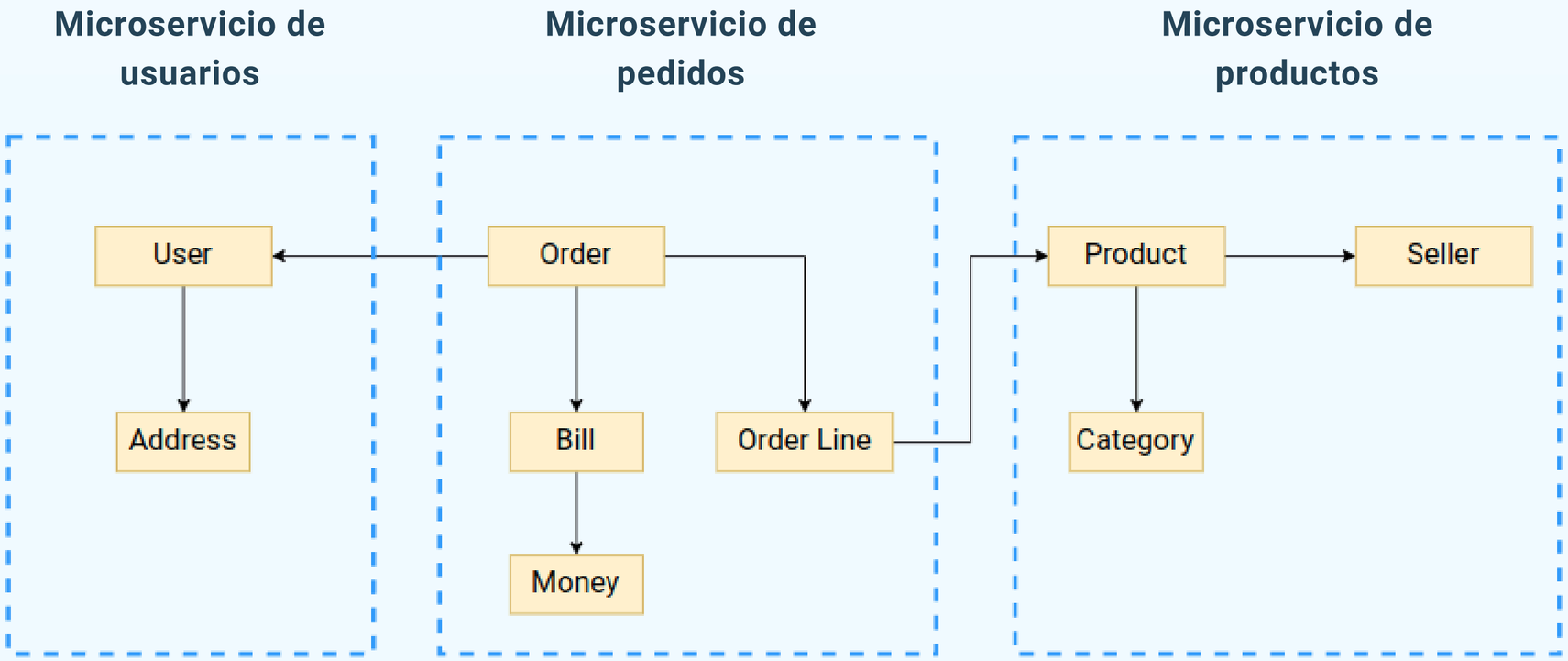
- *Micro* en microservicios se refiere al alcance de las funcionalidades.
  - No hay estándar para lo grandes que deben ser. **Deben hacer una cosa bien.**

- Cada microservicio vivirá de forma independiente a los demás.
  - Equipo propio.
  - Repositorio de código propio.
  - Base de Datos propia.
  - Cada uno elige las **tecnologías** más apropiadas para su caso.
  - Deben ser **desplegables de forma independiente.**

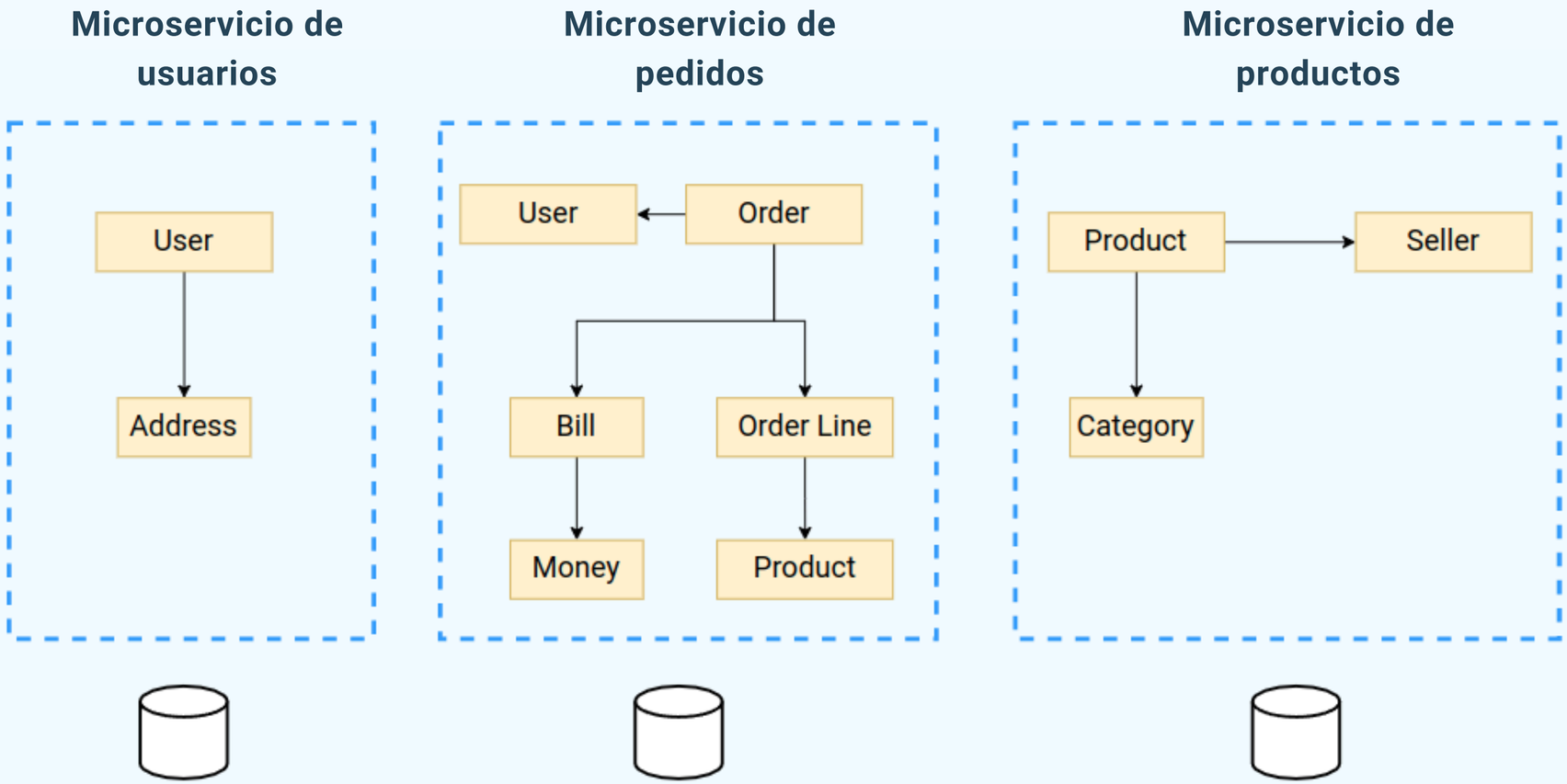
EJEMPLO MICROSERVICIOS



EJEMPLO MICROSERVICIOS

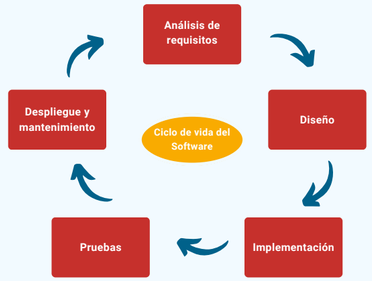


EJEMPLO MICROSERVICIOS

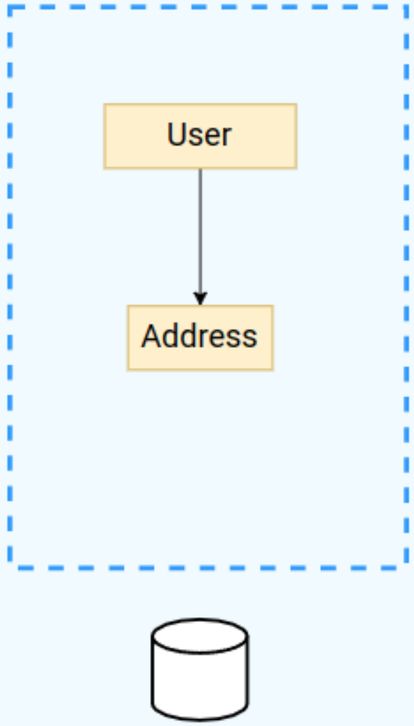


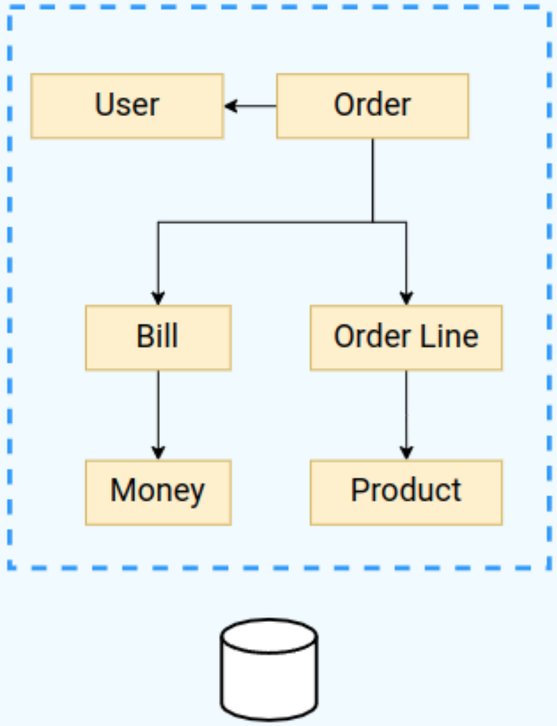
Microservicio de usuarios

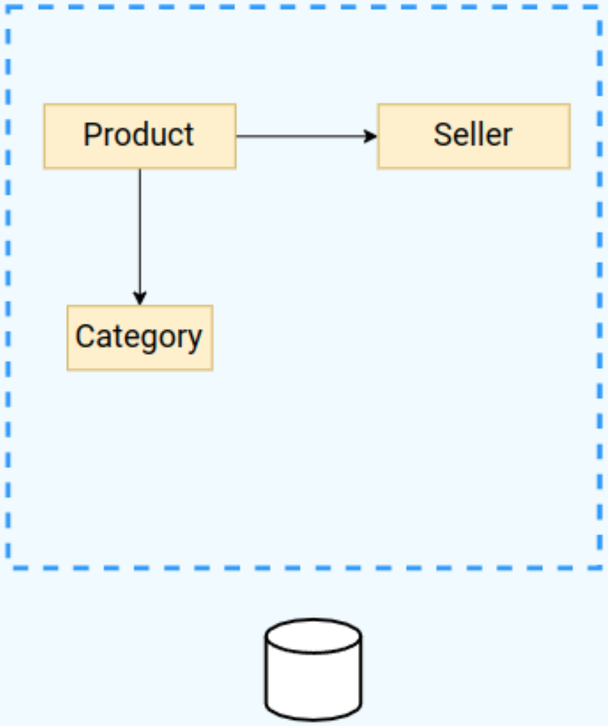
Microservicio de pedidos



Microservicio de productos







COMUNICACIÓN ENTRE MICROSERVICIOS

- **API Rest.**
  - Simple, pero servicios acoplados.
  - Ambos tienen que estar disponibles para que la comunicación funcione.
  - Suele ser síncrono por defecto.

- **Sistema de mensajes**
  - Kafka, RabbitMQ, ActiveMQ etc.
  - **Desacople** total **entre los** dos **servicios.**
  - Procesamiento asíncrono.
  - Si un sistema no está disponible, **el mensaje permanecerá en la cola hasta que vuelva a estar disponible**, momento en el cual será consumido.

ASPECTOS POSITIVOS

- **Despliegue independiente.** Si falla un microservicio, el resto podría seguir funcionando.

- Posibilidad de realizar **escalado y optimización independiente.**
  - Menor coste, empezamos con menos recursos y escalamos cuando sea necesario.

- División del sistema en **subsistemas más manejables.**
  - El código será **más fácil de mantener.**

- **Independencia real entre equipos**. Cada uno es dueño de su microservicio.

- Elección de la **tecnología apropiada para cada microservicio.**

ASPECTOS NEGATIVOS 

- Más complejo en general que una solución monolítica.
  - Necesidad de **identificar** correctamente los **subdominios**.
  - **Test en las fronteras** entre microservicios.
  - **Despliegue** del sistema completo.
  - **Seguridad.**

- **Cooperación entre distintos equipos** para los puntos en común entre microservicios.

- La capa de presentación u otros clientes deben **llamar a diferentes microservicios.**
  - Se mitiga con **API Gateway.**

CUÁNDO USAR

- Sistemas grandes y complejos con **subdominios claramente identificables.**
- **Disponibilidad de personal** para asignar al equipo de cada microservicio.
- **Sistemas de alta disponibilidad** en los que necesitemos escalar fácilmente cada pieza.