DESARROLLO DE SOFTWARE

Arquitectura de Software / Herramientas CASE

INTRODUCCION

En esta clase, exploraremos dos enfoques populares para la arquitectura de software: **monolítico** y **microservicios**, comparándolos con el modelo tradicional **cliente-servidor**. Analizaremos las ventajas y desventajas de cada uno, así como los escenarios en los que son más adecuados.

ARQUITECTURA MONOLÍTICA

Definición: Una aplicación monolítica se construye como una única unidad grande, donde todo el código se encuentra en un mismo lugar.

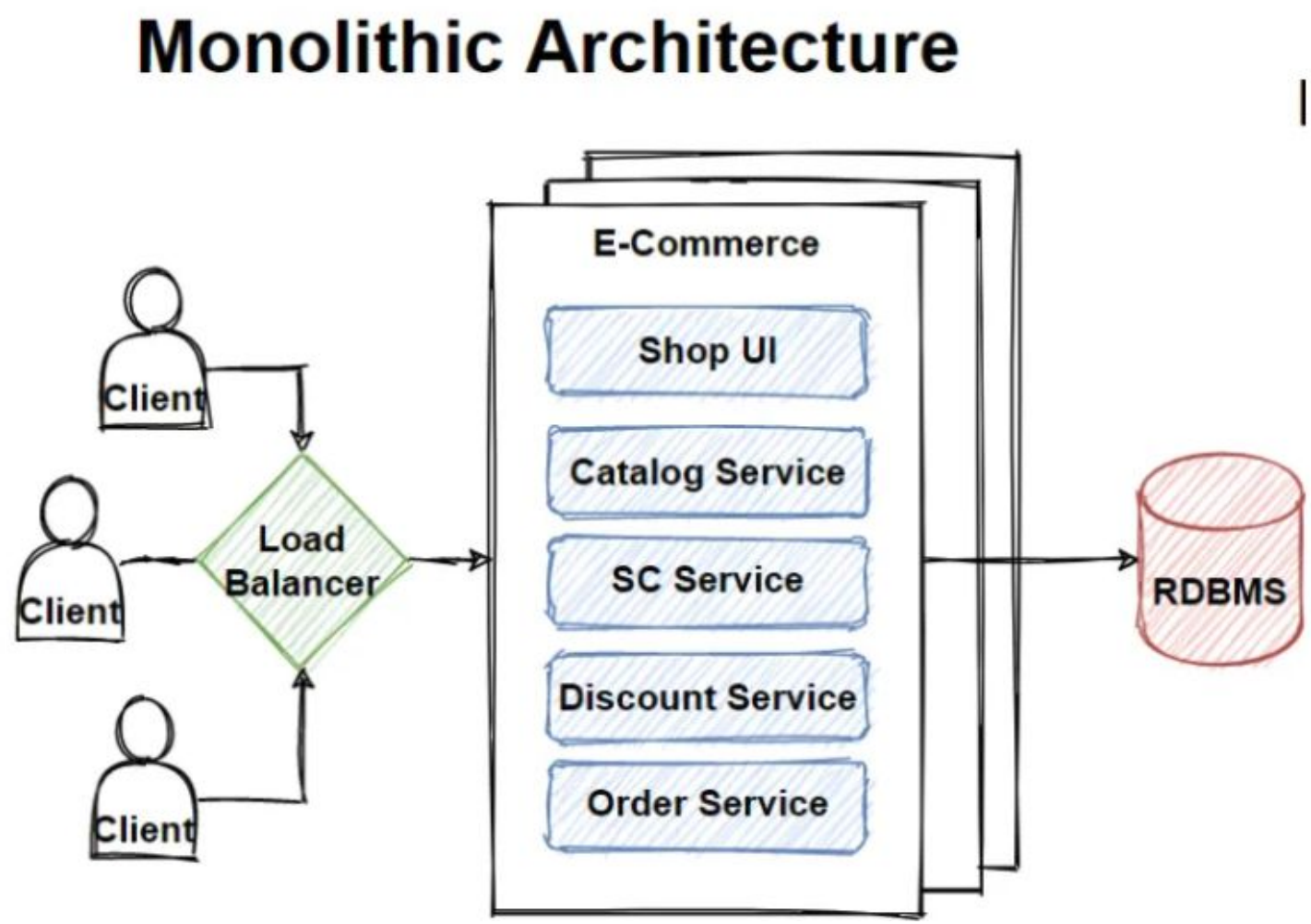
Características:

- Simplicidad: Fácil de implementar y comprender.
- Despliegue: Se implementa como una sola unidad.
- Escalabilidad: Difícil de escalar horizontalmente.
- Mantenimiento: Complejo de modificar y mantener a medida que crece la aplicación.

Ejemplos:

- Aplicaciones web pequeñas.
- Sistemas heredados.

ARQUITECTURA MONOLÍTICA



ARQUITECTURA DE MICROSERVICIOS

Definición: Una aplicación se divide en pequeños servicios independientes, cada uno con su propia responsabilidad y comunicación a través de APIs.

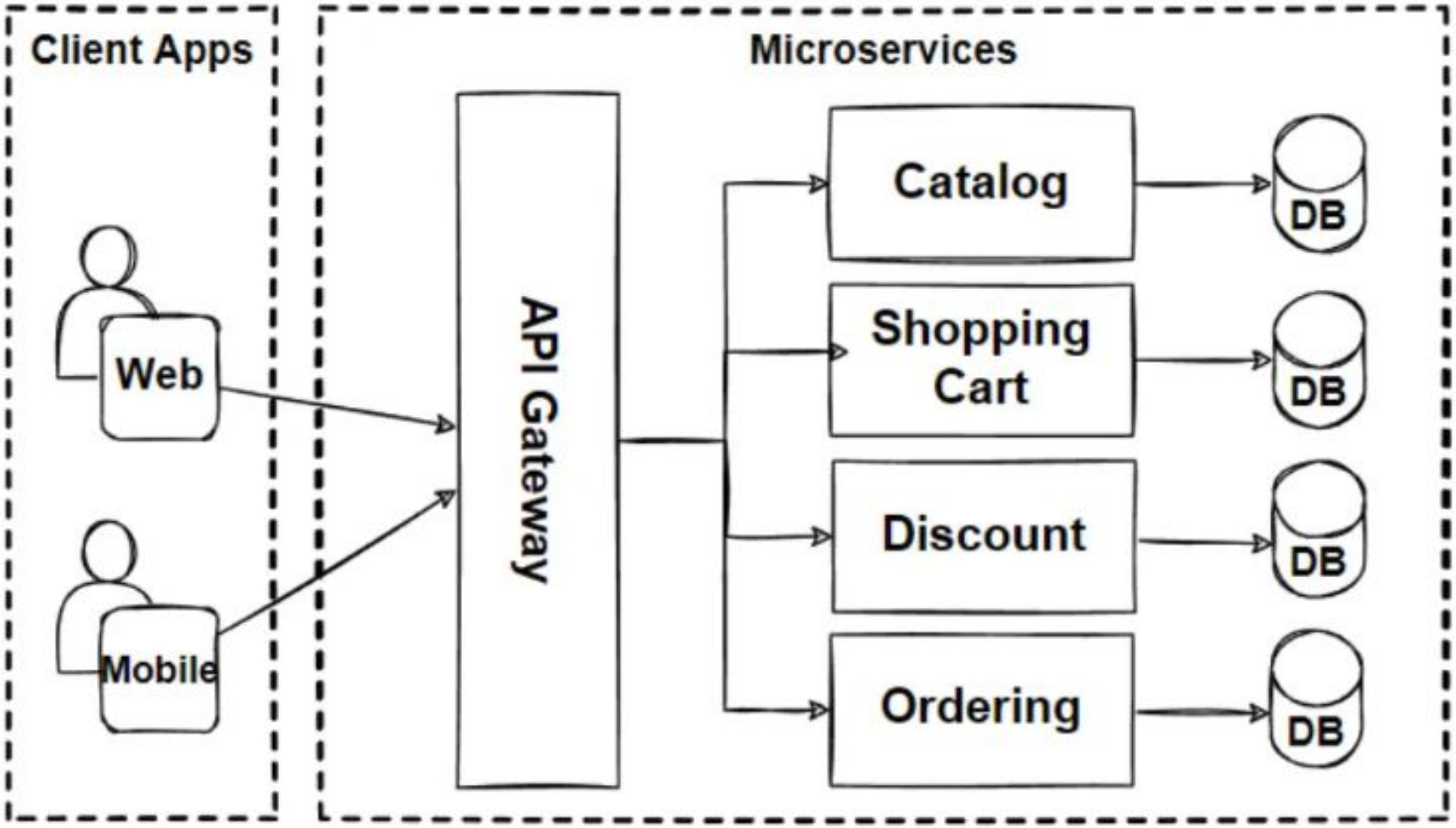
Características:

- Escalabilidad: Altamente escalable horizontalmente.
- Flexibilidad: Permite utilizar diferentes lenguajes y tecnologías.
- Mantenimiento: Facilita el mantenimiento y la actualización de componentes individuales.
- Complejidad: Mayor complejidad de implementación y gestión.

Ejemplos:

- Netflix.
- Amazon.

ARQUITECTURA DE MICROSERVICIOS



MODELO CLIENTE SERVIDOR

Definición: Un modelo de arquitectura donde los clientes solicitan recursos a los servidores, quienes los proporcionan y procesan.

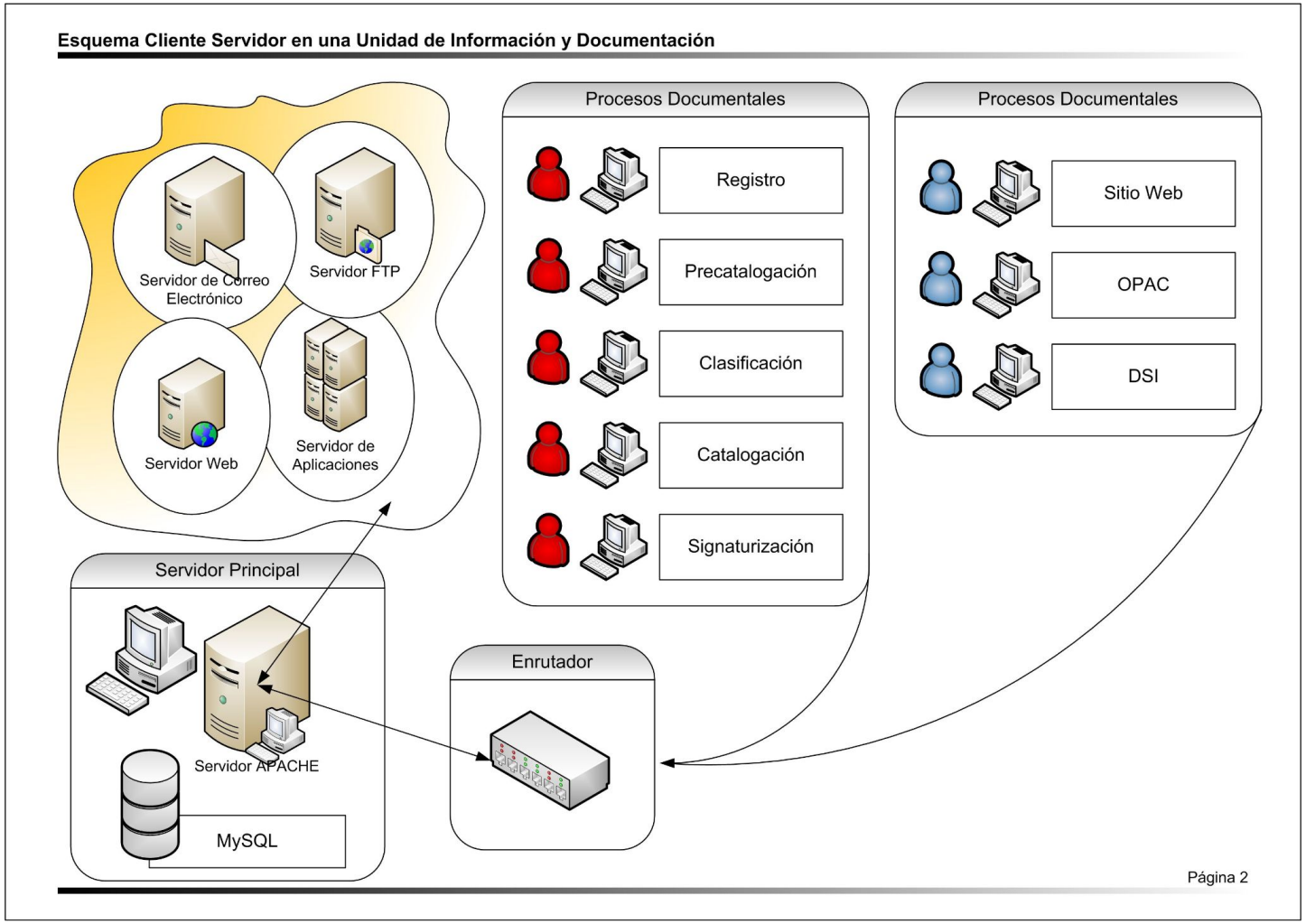
Características:

- Simplicidad: Concepto fácil de entender e implementar.
- Escalabilidad: Altamente escalable horizontalmente.
- Seguridad: Permite un buen control de acceso y seguridad.
- Desempeño: Puede tener un impacto en el rendimiento por la comunicación entre cliente y servidor.

Ejemplos:

- Aplicaciones web con gran tráfico.
- Correo electrónico.

MODELO CLIENTE SERVIDOR

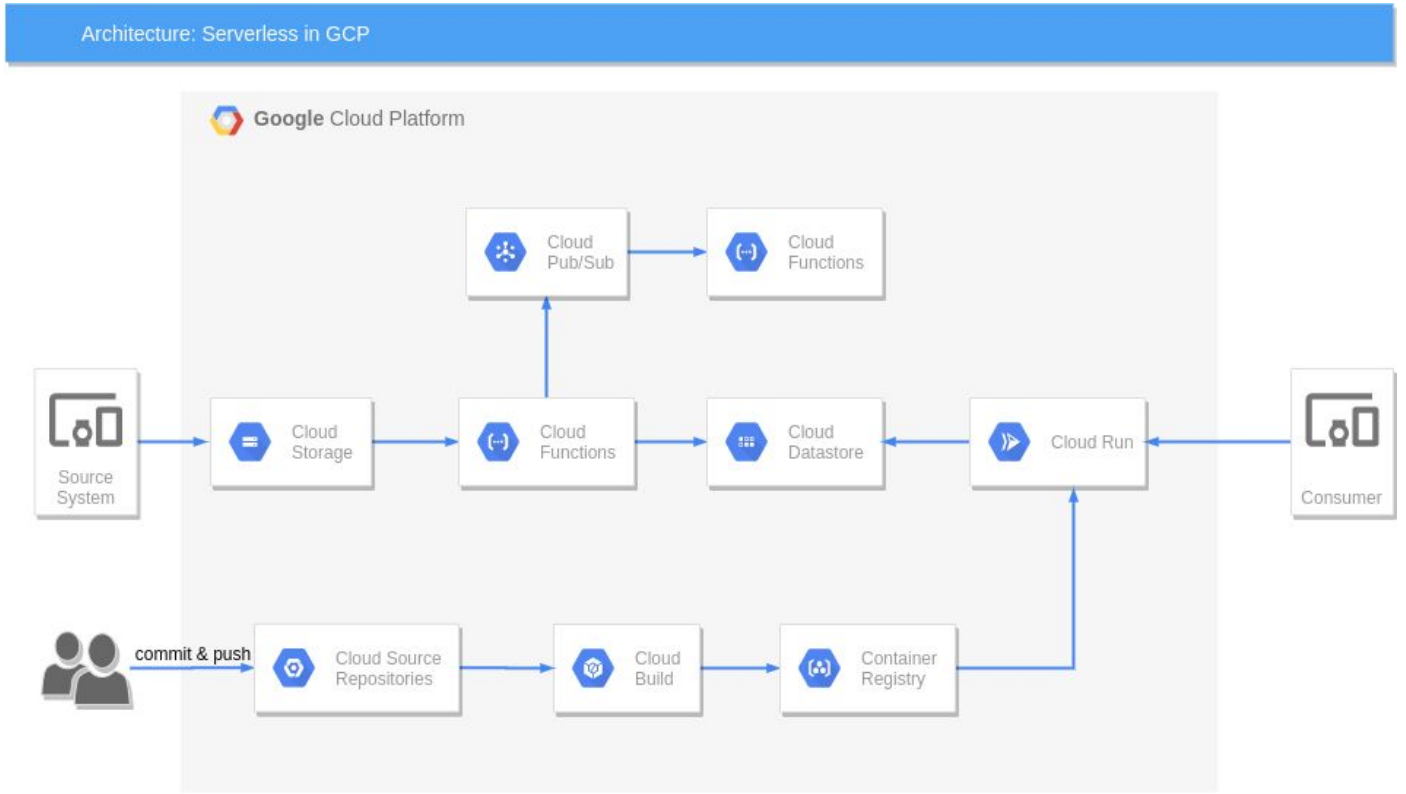


COMPARATIVA

Característica	Monolítico	Microservicios	Cliente-Servidor
Implementación	Simple	Compleja	Simple
Despliegue	Una sola unidad	Unidades individuales	Cliente y servidor
Escalabilidad	Limitada	Alta	Alta
Mantenimiento	Complejo	Fácil	Moderado
Flexibilidad	Baja	Alta	Moderada
Tecnología	Homogénea	Heterogénea	Homogénea o heterogénea

ARQUITECTURA SERVERLESS

- **CLOUD FUNCTION**
- **AWS LAMBDA**
- **AZURE FUNCTION**
- **LEER**



HERRAMIENTAS CASE - Computer Aided Software Engineering

Las Herramientas CASE son un conjunto de aplicaciones informáticas que automatizan y apoyan las diferentes etapas del ciclo de vida del desarrollo de software (SDLC). Estas herramientas brindan soporte a los desarrolladores, analistas y gerentes de proyectos, facilitando la creación, modelado, documentación y mantenimiento de sistemas de software.

HERRAMIENTAS CASE - Computer Aided Software Engineering

- **Herramientas de modelado**: Permiten crear diagramas y modelos que representan el sistema de software, como diagramas de flujo de datos, diagramas UML y modelos de entidad-relación.
- **Herramientas de generación de código**: Automatizan la generación de código fuente a partir de modelos o diagramas, reduciendo el tiempo de desarrollo y minimizando errores.
- **Herramientas de análisis**: Ayudan a analizar el diseño del software, identificando posibles problemas y mejorando la calidad del código.
- **Herramientas de gestión de proyectos**: Facilitan la planificación, organización, seguimiento y control del proyecto de desarrollo de software.
- **Herramientas de reingeniería**: Permiten comprender, modificar y actualizar sistemas de software existentes.

HERRAMIENTAS CASE - EJEMPLO

Modelado: UMLet, StarUML, Enterprise Architect.

Generación de código: JDeveloper, PowerBuilder, Copilot.

Análisis: Rational Rose, CodePro Analyzer, SonarQube.

Gestión de proyectos: Jira, Trello, Asana, Monday.

Controlador de versiones: Github, Bitbucket

Deployments: Jenkins, Github Actions

GRACIAS!