

Presentación 1

Tomas martinez, Daniel Millan , Camilo Rodriguez

Nuestro conjunto de conceptos, patrones, estilos, principios, tecnologías y frameworks:

- microservicios
- front:svelt
- int:REST/XML
- back:GO
- BD:mariaDB

Definición, Historia y Evolución

Microservicios

Definición:

Los microservicios son un patrón arquitectónico en el que una aplicación se divide en componentes pequeños e independientes, llamados microservicios. Cada microservicio cumple una función específica y se comunica con otros a través de API. Algunas características clave de los microservicios son:

- Independencia: Cada microservicio se puede desarrollar, implementar y operar de forma independiente sin afectar a otros servicios.
- Mantenibilidad y pruebas: Facilitan la experimentación con nuevas funciones y permiten revertirlas si no funcionan. También simplifican el aislamiento y la corrección de errores en servicios individuales.
- Equipos pequeños: Los equipos multifuncionales crean y mantienen microservicios, lo que acelera el desarrollo.

- Organización en torno a capacidades empresariales: Los servicios se agrupan según funcionalidades específicas.
- Infraestructura automatizada: Se utilizan prácticas de automatización de infraestructuras como CI/CD.

Historia y Evolución:

Aunque el término “microservicios” se popularizó en la última década, los principios subyacentes han existido durante mucho tiempo.

En la década de 1970, se promovía la modularidad y la separación de responsabilidades en el diseño de sistemas.

En la década de 2000, surgieron las arquitecturas orientadas a servicios (SOA), que compartían similitudes con los microservicios.

Sin embargo, SOA a menudo se asociaba con servicios pesados y acoplados.

Los microservicios modernos se popularizaron con el auge de la computación en la nube y la necesidad de escalabilidad y mantenibilidad en aplicaciones web y móviles.

Svelte

Definición: Svelte es un framework de desarrollo web que compila a JavaScript puro en tiempo de compilación. Permite crear aplicaciones web reactivas y eficientes.

Historia y Evolución:

Svelte fue creado por Rich Harris y se lanzó en 2016.

A diferencia de otros frameworks como React o Vue, Svelte no se ejecuta en tiempo de ejecución. En su lugar, compila el código en JavaScript optimizado.

Su enfoque en la eficiencia y la simplicidad lo ha convertido en una opción popular para desarrolladores web.

REST/XML

Definición: REST (Representational State Transfer) es un estilo arquitectónico para sistemas distribuidos basado en HTTP. Se centra en la transmisión de datos a través de API y en representaciones de recursos.

XML (Extensible Markup Language): Es un formato de marcado utilizado para estructurar datos.

Historia y Evolución:

REST se popularizó en la década de 2000 como una alternativa a los servicios web SOAP.

XML ha sido ampliamente utilizado para representar datos estructurados en aplicaciones web y servicios.

Go (Golang)

Definición: Go es un lenguaje de programación de código abierto creado por Google. Es conocido por su rendimiento, eficiencia y facilidad de uso.

Historia y Evolución:

Go fue anunciado en 2009 y se lanzó oficialmente en 2012.

Diseñado para ser simple, rápido y seguro, Go se ha convertido en una opción popular para el desarrollo backend y la creación de microservicios.

MariaDB

Definición: MariaDB es una base de datos relacional de código abierto. Es una bifurcación de MySQL y ofrece características adicionales y mejoras.

Historia y Evolución:

MariaDB se originó en 2009 como una respuesta a las preocupaciones sobre la adquisición de MySQL por parte de Oracle.

MariaDB mantiene la compatibilidad con MySQL y ha seguido evolucionando con nuevas características y mejoras.

Relación entre los Componentes

1. Microservicios y Svelte:

- Los microservicios se encargan de la lógica de negocio y la gestión de datos.
- Svelte se utiliza en el frontend para crear interfaces de usuario reactivas que consumen datos de los microservicios.
- La comunicación entre el frontend (Svelte) y los microservicios se realiza a través de API REST.

2. Microservicios y REST/XML:

- Los microservicios exponen sus funcionalidades a través de API REST.
- REST define cómo se comunican los servicios mediante solicitudes HTTP (GET, POST, PUT, DELETE).
- XML puede ser utilizado como formato de representación de datos en las respuestas de los microservicios.

3. Microservicios y Go (Golang):

- Go es una excelente opción para implementar los microservicios en el backend.
- Los microservicios escritos en Go pueden manejar múltiples solicitudes concurrentes de manera eficiente.
- Go proporciona bibliotecas estándar para crear servidores HTTP y gestionar rutas y controladores.

4. Microservicios y MariaDB:

- Los microservicios necesitan almacenar y recuperar datos.
- MariaDB es una base de datos relacional que puede utilizarse para persistir datos utilizados por los microservicios.
- Los microservicios se comunican con la base de datos a través de consultas SQL.

Situaciones y Problemas

1. Escalabilidad y Mantenibilidad:

- Los microservicios son ideales para aplicaciones grandes y complejas que requieren escalabilidad y mantenibilidad.
- Svelte permite crear interfaces de usuario interactivas y eficientes.
- Go es adecuado para aplicaciones backend que requieren alta concurrencia.

2. Integración de Sistemas Externos:

- REST/XML se aplica en la exposición de APIs y servicios web.
- Los microservicios pueden consumir APIs externas utilizando REST/XML para integrar funcionalidades adicionales.

3. Almacenamiento de Datos:

- MariaDB es una buena opción para almacenar datos estructurados en aplicaciones empresariales.
- Los microservicios pueden utilizar MariaDB para persistir datos relevantes.

Ventajas y Desventajas

Microservicios

Ventajas:

- Acelera la escalabilidad¹.
- Mejora el aislamiento de los fallos.
- Mejora la productividad del equipo.
- Acorta el tiempo de implementación.
- Costes más eficientes.

Desventajas:

- Complejidad en la comunicación y gestión.
- Mayor consumo de memoria.
- Mayor complejidad en la gestión de la configuración.
- Mayor complejidad en el monitoreo y la depuración.
- Mayor complejidad en el despliegue inicial.

Svelte

Ventajas:

- Sencillo desde la perspectiva del programador.
- Compilación rápida de componentes.
- Sintaxis simple.

- CSS por componentes o en modo global.
- Componentes reactivos.

Desventajas:

- No tiene tantas características como otros frameworks.
- La interfaz de usuario no es tan intuitiva como otras bases de datos.

Rest/XML

Ventajas:

- Facilidad de uso.
- Flexibilidad.

Desventajas:

- Puede ser difícil comprender y configurar para los principiantes.

Go

Ventajas:

- Es un lenguaje concurrente que soporta canales de comunicación CSP.
- Cuenta con un recolector de basura que permite elevar al máximo la eficiencia y el rendimiento, y reducir al mínimo la latencia.
- Tiene una sintaxis clara y concisa.
- Posee una comunidad activa de desarrolladores que contribuyen con bibliotecas, herramientas y recursos.

Desventajas:

- No se encontraron desventajas específicas en las fuentes consultadas.

MariaDB

Ventajas:

- MariaDB es más seguro que MySQL.
- Ofrece mejor rendimiento y escalabilidad.
- Al igual que MySQL, MariaDB es de software libre.

Desventajas:

- MariaDB no tiene tantas características como MySQL.
- La interfaz de usuario no es tan intuitiva como otras bases de datos.

Principios SOLID

MariaDB

SRP: Cada tabla en MariaDB debe tener una única responsabilidad.

OCP: MariaDB permite extender las tablas (agregar nuevas columnas) sin necesidad de modificar las existentes.

LSP: Las tablas en MariaDB deben ser intercambiables sin afectar el funcionamiento de la aplicación.

ISP: MariaDB promueve la creación de tablas con interfaces simples y específicas para los clientes.

DIP: En MariaDB, las tablas deben depender de abstracciones, no de implementaciones concretas.

Go

SRP: Cada paquete o función en Go debe tener una única responsabilidad.

OCP: Go permite extender el comportamiento de las estructuras a través de la implementación de interfaces, sin necesidad de modificar las estructuras.

LSP: Las estructuras en Go que implementan una interfaz deben ser intercambiables sin afectar el funcionamiento del programa.

ISP: Go promueve la creación de interfaces pequeñas y específicas.

DIP: En Go, las estructuras y funciones deben depender de interfaces, no de implementaciones concretas.

Rest/XML

SRP: Cada endpoint en una API Rest debe tener una única responsabilidad.

OCP: Las APIs Rest deben permitir la extensión (agregar nuevos endpoints) sin necesidad de modificar los existentes.

LSP: Los endpoints en una API Rest deben ser intercambiables sin afectar el funcionamiento de la aplicación.

ISP: Las APIs Rest deben proporcionar interfaces simples y específicas para los clientes.

DIP: En una API Rest, los endpoints deben depender de abstracciones, no de implementaciones concretas.

Svelte

SRP: Cada componente en Svelte debe tener una única responsabilidad.

OCP: Svelte permite extender componentes sin necesidad de modificarlos.

LSP: Los componentes en Svelte deben ser intercambiables sin afectar el funcionamiento de la aplicación.

ISP: Svelte promueve la creación de componentes con interfaces simples y específicas.

DIP: En Svelte, los componentes deben depender de abstracciones (como props y eventos), no de implementaciones concretas.

MicroServicios

Principio de Responsabilidad Única (SRP): Cada microservicio debe tener una única responsabilidad. Por ejemplo, un microservicio puede ser responsable de manejar todo lo relacionado con los usuarios, mientras que otro puede manejar todo lo relacionado con los productos.

Principio Abierto-Cerrado (OCP): Los microservicios deben estar abiertos para la extensión, pero cerrados para la modificación. Esto significa que deberíamos poder agregar nuevas funcionalidades a un microservicio sin cambiar su código.

Principio de Sustitución de Liskov (LSP): En el contexto de los microservicios, este principio se puede interpretar como que cada microservicio debe ser reemplazable por otro sin afectar el funcionamiento del sistema.

Principio de Segregación de Interfaces (ISP): Los microservicios deben proporcionar interfaces simples y específicas para los clientes.

Principio de Inversión de Dependencias (DIP): Los microservicios deben depender de abstracciones, no de implementaciones concretas.

Atributos de Calidad

Microservicios

Eficiencia: Los microservicios facilitan el mantenimiento, evolución y el mejoramiento continuo de los activos digitales.

Escalabilidad: Dividir una aplicación en microservicios permite que cada componente se pueda escalar de manera individual.

Compatibilidad con diferentes lenguajes y tecnologías.

Svelte

Rendimiento: Svelte es capaz de realizar consultas más complejas en comparación a otros frameworks, gracias a su compilador que le permite almacenar en caché los datos.

Facilidad de modificación: Svelte permite extender componentes sin necesidad de modificarlos.

Portabilidad: Svelte es un framework liviano, lo que facilita su implementación en diferentes plataformas.

Rest/XML

Eficiencia: Las APIs Rest son eficientes en términos de rendimiento y uso de recursos.

Interoperabilidad: Las APIs Rest pueden interactuar con cualquier sistema que use protocolos HTTP.

Modificabilidad: Las APIs Rest permiten la extensión (agregar nuevos endpoints) sin necesidad de modificar los existentes.

Go

Eficiencia: Go es conocido por su eficiencia en términos de rendimiento y uso de recursos.

Facilidad de modificación: Go permite extender el comportamiento de las estructuras a través de la implementación de interfaces, sin necesidad de modificar las estructuras.

Portabilidad: Go es un lenguaje compilado que puede ejecutarse en diferentes plataformas.

MariaDB

Eficiencia: MariaDB es capaz de realizar consultas más complejas en comparación a otros DBMS, gracias a su motor de almacenamiento que le permite almacenar en caché los datos.

Escalabilidad: MariaDB permite que cada componente se pueda escalar de manera individual.

Compatibilidad con diferentes lenguajes y tecnologías.

Casos de Uso:

Microservicios:

Netflix: Esta plataforma utiliza microservicios para el funcionamiento de sus productos (AWS).

Software de recursos humanos: Se puede desarrollar un software de recursos humanos utilizando una arquitectura de microservicios.

Servicios de autenticación: Los microservicios pueden ser utilizados para proteger aplicaciones de los ataques de hackers, controlar el acceso a los datos y asegurar que todos los usuarios autorizados tengan acceso a la información deseada.

Svelte:

Aplicaciones de una sola página (SPA): Svelte es una gran opción para la construcción de aplicaciones de una sola página que requieren actualizaciones en tiempo real e interfaces de usuario dinámicas.

Aplicaciones web progresivas (PWA): Svelte se puede utilizar para crear aplicaciones web progresivas que proporcionan una experiencia similar a una aplicación nativa en la web.



Why Svelte Is the Next Big Thing

150%

growth in usage
(from 8% to 20%
in the last 2 years)

94%

awareness ratio
(growth from 75%
in the last 2 years)

90%

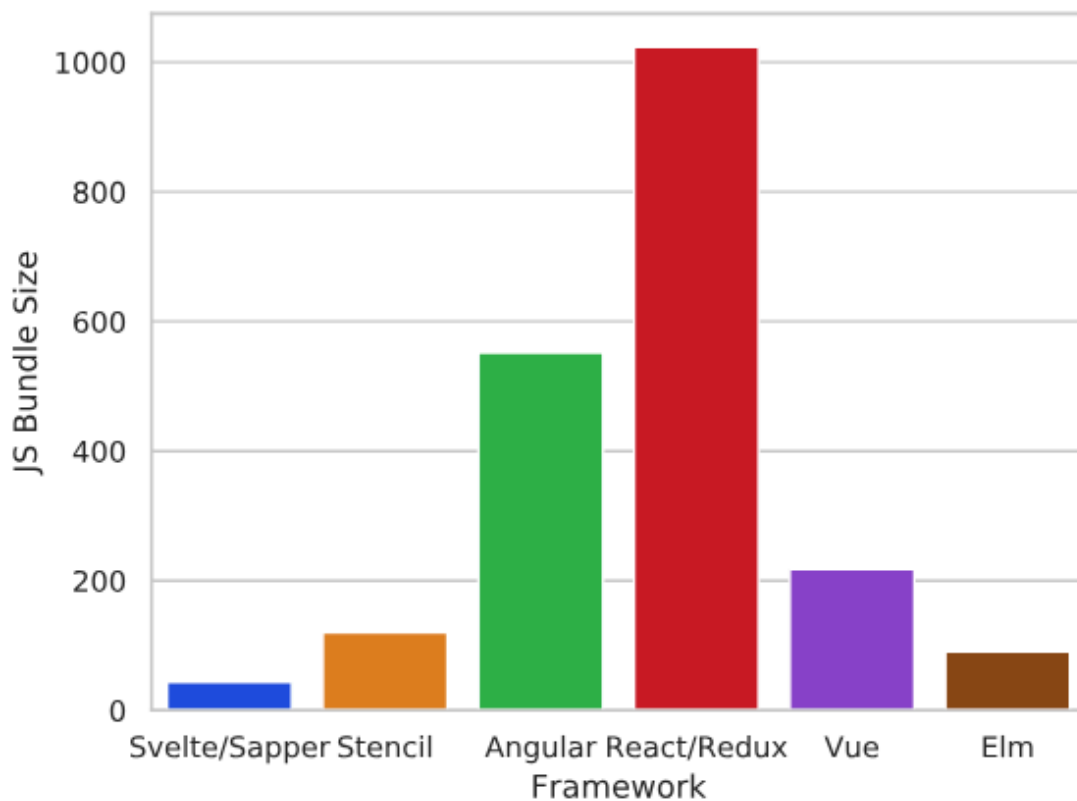
satisfaction ratio
(higher than React
or Vue)

First

position in the Stack Overflow's
ranking of Most Loved
Frameworks

naturally

Sources: 2021 Developer Survey (Stack Overflow), The State of JS 2021

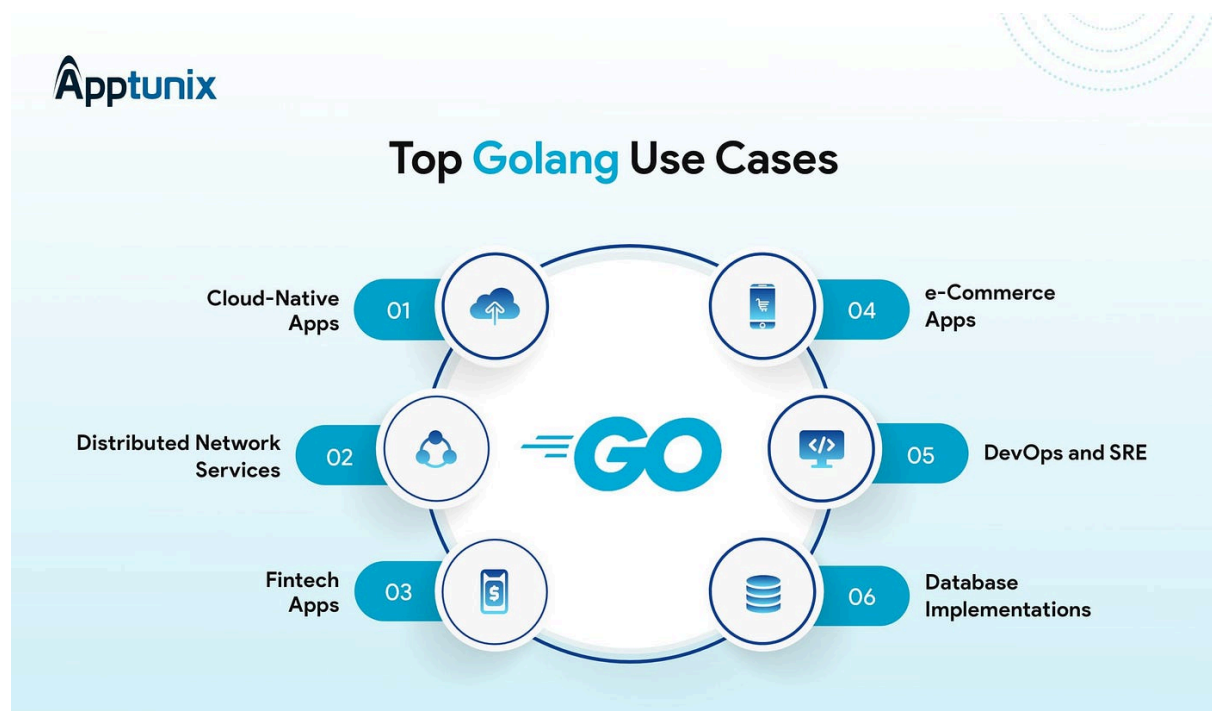




Go:

Google: Go fue creado en Google en 2007, y desde entonces, los equipos de ingeniería de Google han adoptado Go para construir productos y servicios a gran escala.

PayPal: PayPal utiliza Go para modernizar y escalar sus servicios.

American Express: Go proporciona a American Express la velocidad y escalabilidad que necesita para sus redes de pagos y recompensas.



Feature	Golang 	Node.js 
Learning Curve	Comparatively difficult	Easier
Raw Performance	Excellent	Poor compared to GO
Scalability & Concurrency	Better	Good
Error Handling	Average	Average
Development Tools	Fair enough	Plenty
Community	Growing	Vast community

MariaDB:

Transacciones: MariaDB Enterprise Server es la base de datos de código abierto líder para el procesamiento de transacciones a cualquier escala.

Analítica: MariaDB Enterprise Server puede ser desplegado como un almacén de datos o base de datos analítica utilizando almacenamiento columnar y procesamiento masivamente paralelo (MPP) para realizar consultas interactivas y ad hoc en cientos de miles de millones de filas sin crear índices.

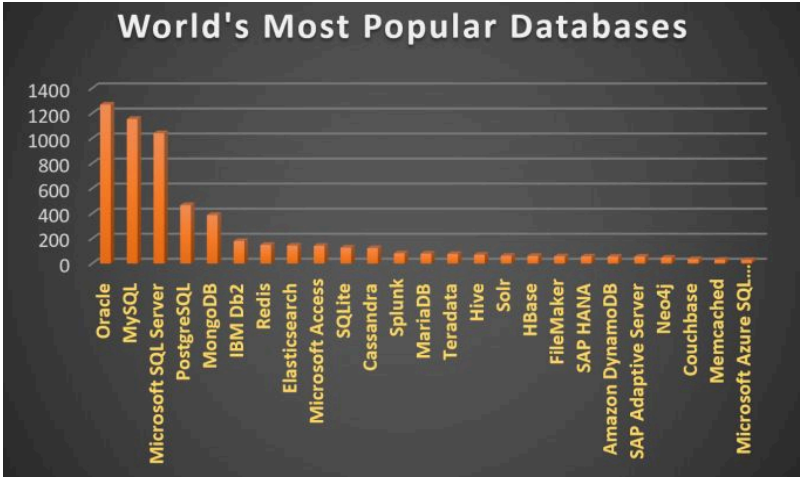
Transacciones inteligentes: MariaDB Enterprise Server soporta transacciones inteligentes (es decir, procesamiento transaccional/analítico híbrido o HTAP) combinando almacenamiento de filas optimizado para transacciones rápidas con almacenamiento columnar optimizado para análisis rápidos.



Empresas que Usan MariaDB:

DBS Bank: Una de las principales instituciones financieras de Singapur.

Infosim GmbH & Co. KG: Una empresa alemana de tecnología de la información y servicios.

EBP Informatique: Una empresa francesa de tecnología de la información y servicios.



Maria DB 	MySQL 
+ Performance	+ Support
+ Connections	+ Oracle Corp.
+ Open-Source	+ History