Grupo 6

API Rest, Swagger, React, Microservicios

Nava Agustin, Marin Damian, Aquino Arami, Soengas Sofia, Foppoli Jennifer

Application Programming Interface

Funciona como un traductor que permite a dos componentes de software entenderse y trabajar juntos

Introducción

Por sus siglas en inglés, Interfaz de Programación de Aplicaciones.

Las API son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un **conjunto de definiciones y protocolos**.

En tal sentido, es un intermediario virtual: envía información de una interfaz a otra. Pueden funcionar de cuatro maneras diferentes, según el momento y el motivo de su creación.

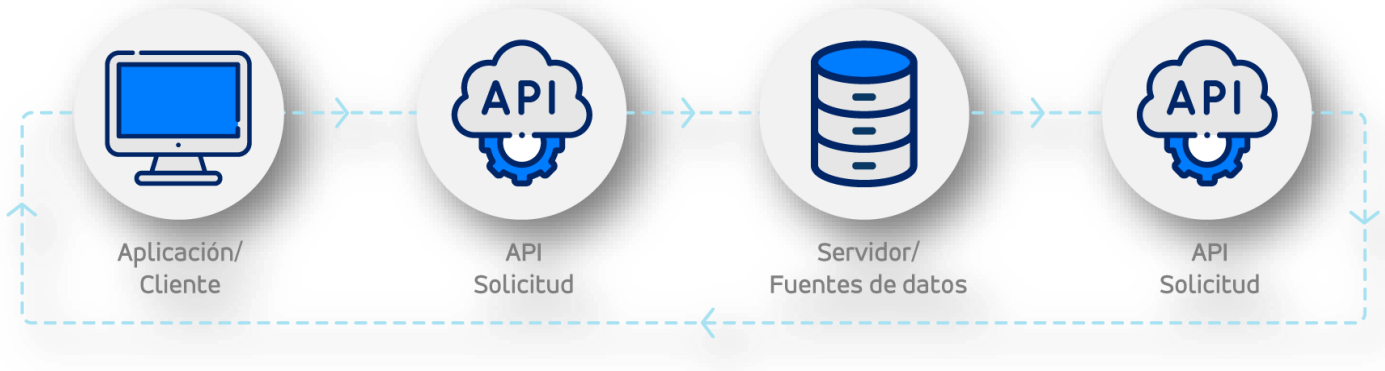
Funcionamiento

Arquitectura

La arquitectura de las API suele explicarse en términos de cliente y servidor. La aplicación que envía la solicitud se llama cliente, y la que envía la respuesta se llama servidor.

Interoperabilidad

Las API definen cómo deben ser los mensajes que se intercambian, qué información se necesita enviar y recibir, y cómo deben interpretarse estos datos. De esta manera, los diferentes componentes de software pueden colaborar de forma eficiente, utilizando las funcionalidades que cada uno ofrece, sin necesidad de conocer todos los detalles internos del otro.



Clasificación según su funcionamiento

SOAP

- Menos flexible
- Intercambio de mensajes mediante XML
- Utiliza el protocolo simple de acceso a objetos

RCP

- Llamadas a procedimientos remotos
- Los procesos pueden distribuirse
- No existe un estándar unificado

WebSocket

- Protocolo de red basado en TCP
- Intercambio de datos en ambas direcciones al mismo tiempo
- Comunicación más eficiente que HTTP

REST

- Más populares y flexibles
- Conjunto de principios arquitectónicos
- Intercambio de mensajes mediante JSON

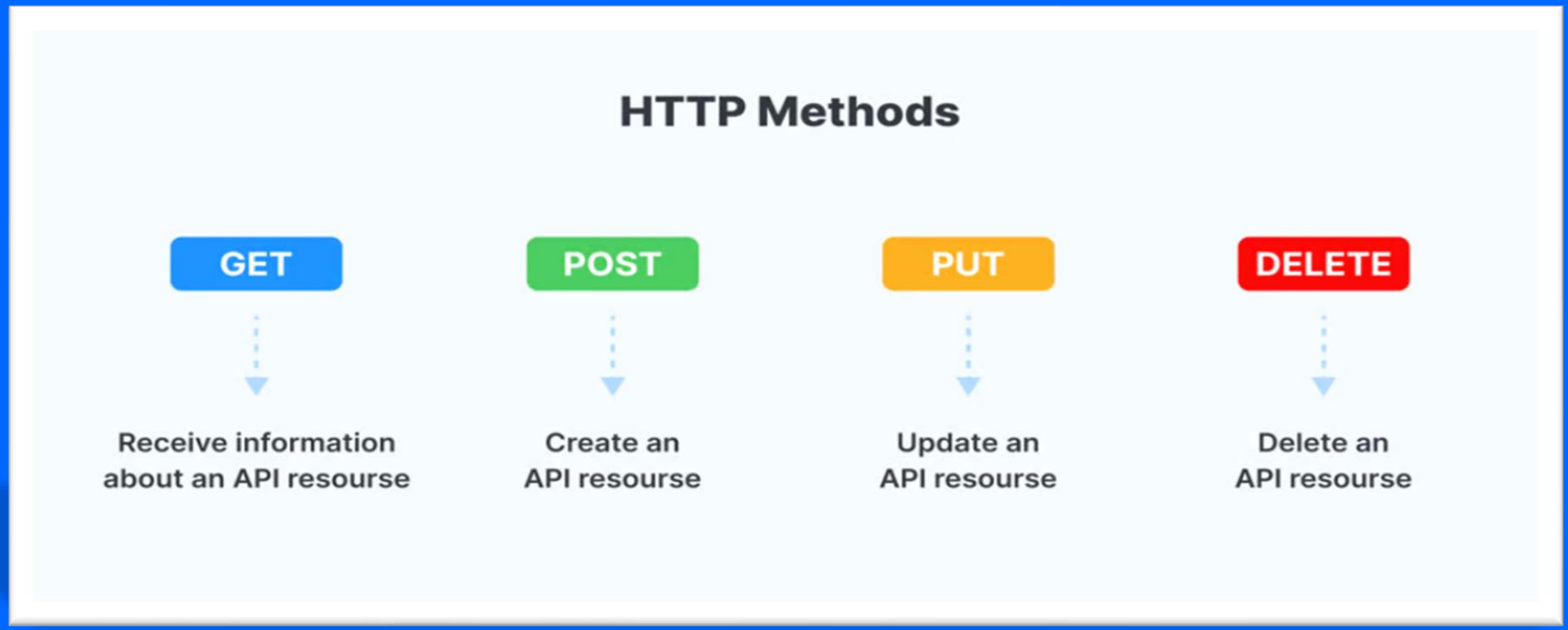
API REST

(Representational state transfer)

No es un protocolo ni un estándar, sino más bien un conjunto de límites de arquitectura.

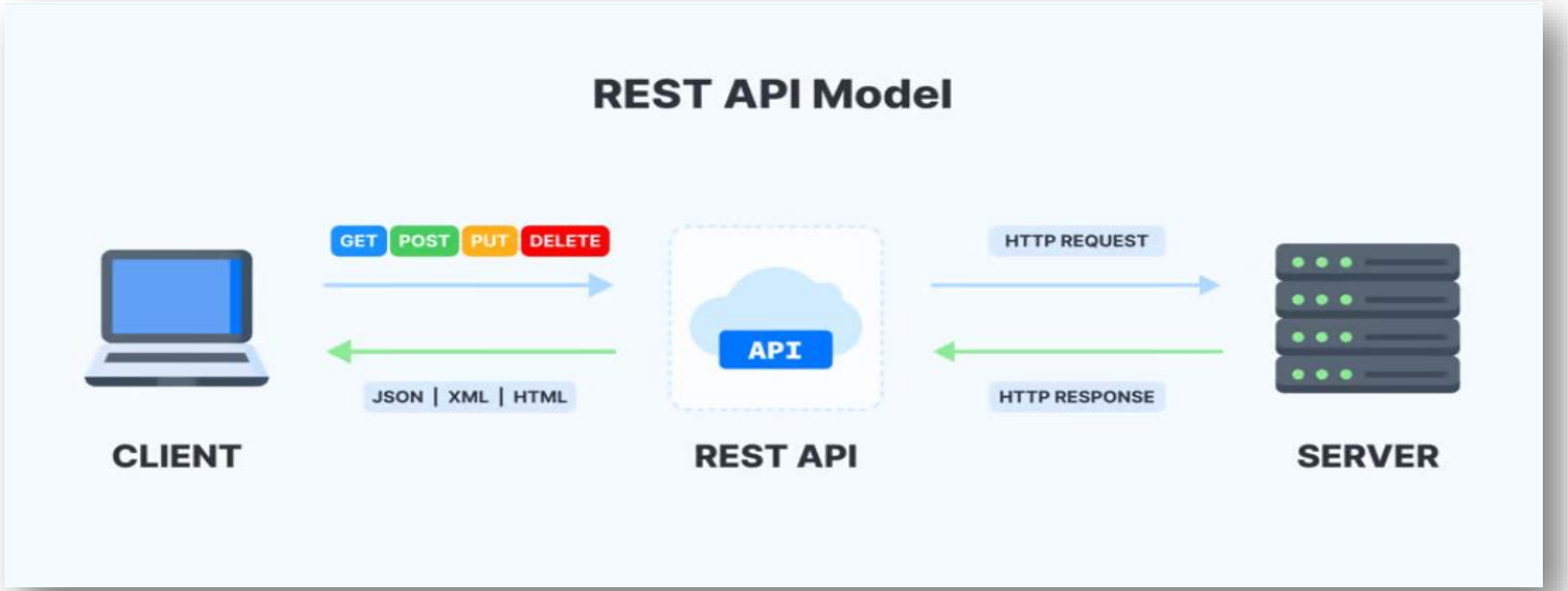
Fundamentos

Utiliza el protocolo de transferencia de hipertexto (hypertext transfer protocol o HTTP, por sus siglas en inglés) para obtener datos o ejecutar operaciones sobre dichos datos en diversos formatos, como pueden ser XML o JSON.



La principal característica de la API REST es que no tiene estado. La ausencia de estado significa que los servidores no guardan los datos del cliente entre las solicitudes.

Se basa en el modelo cliente-servidor donde el cliente es el que solicita obtener los recursos o realizar alguna operación sobre dichos datos, mientras que el servidor es aquel ente que entrega o procesa dichos datos a solicitud del cliente.



Recurso

cualquier contenido (video, texto, etc.) que el servidor transmite al cliente. Es un concepto crítico en REST API, una abstracción de información. Puede ser cualquier información: documento, imagen, servicio temporal.

Funcionamiento de una API REST

Como ya se mencionó antes, La API REST se comunica a través de solicitudes HTTP. REST proporciona la información sobre los recursos solicitados y utiliza métodos para describir qué hacer con un recurso.

La información puede ser entregada al cliente en varios formatos: El más popular y usado es JSON porque es legible por humanos y máquinas, y es independiente del lenguaje.

Para acceder a un recurso, un cliente debe realizar una solicitud. Después de recibirlo, el servidor generará una respuesta con datos codificados sobre un recurso.

Estructura de una solicitud

Método HTTP	Describe lo que se debe hacer con el recurso.
Endpoint	Contiene un URI: identificador uniforme de recursos, que indica cómo y dónde se puede encontrar el recurso.
Encabezados	Contienen los datos relacionados con el cliente y el servidor, como también la información sobre el formato de respuesta.
cuerpo	Se usa para enviar información adicional al servidor, como los datos que desea agregar.

Ventajas

1. Integración
2. Innovación
3. Ampliación
4. Facilidad de mantenimiento

Documentación

Aunque las API son explicativas, la documentación de las mismas sirve de guía para mejorar su uso. Las API bien documentadas que ofrecen una gama de funciones y casos de uso tienden a ser más populares en una arquitectura orientada a servicios.

La escritura de la documentación completa de la API forma parte del proceso de administración de la API. La documentación de la API puede generarse automáticamente mediante herramientas como Swagger, la cual veremos más adelante, o escribirse manualmente.

Buenas prácticas para una documentación manual

Entre algunas de las prácticas recomendadas se encuentran las siguientes:

- Escribir las explicaciones en un inglés sencillo y fácil de leer.
- Utilizar ejemplos de código para explicar la funcionalidad.
- Mantener la documentación para que sea precisa y esté actualizada.
- Orientar el estilo de escritura a los principiantes
- Cubrir todos los problemas que la API puede resolver por los usuarios.

Swagger

Es una serie de reglas, especificaciones y herramientas que nos ayudan a documentar nuestras APIs

Swagger bajo el estándar Open API

Cuando hablamos de Swagger nos referimos a una serie de reglas, especificaciones y herramientas que nos ayudan a documentar nuestras APIs. De esta manera, podemos realizar documentación que sea realmente útil para las personas que la necesitan. Swagger nos ayuda a crear documentación que todo el mundo entienda.

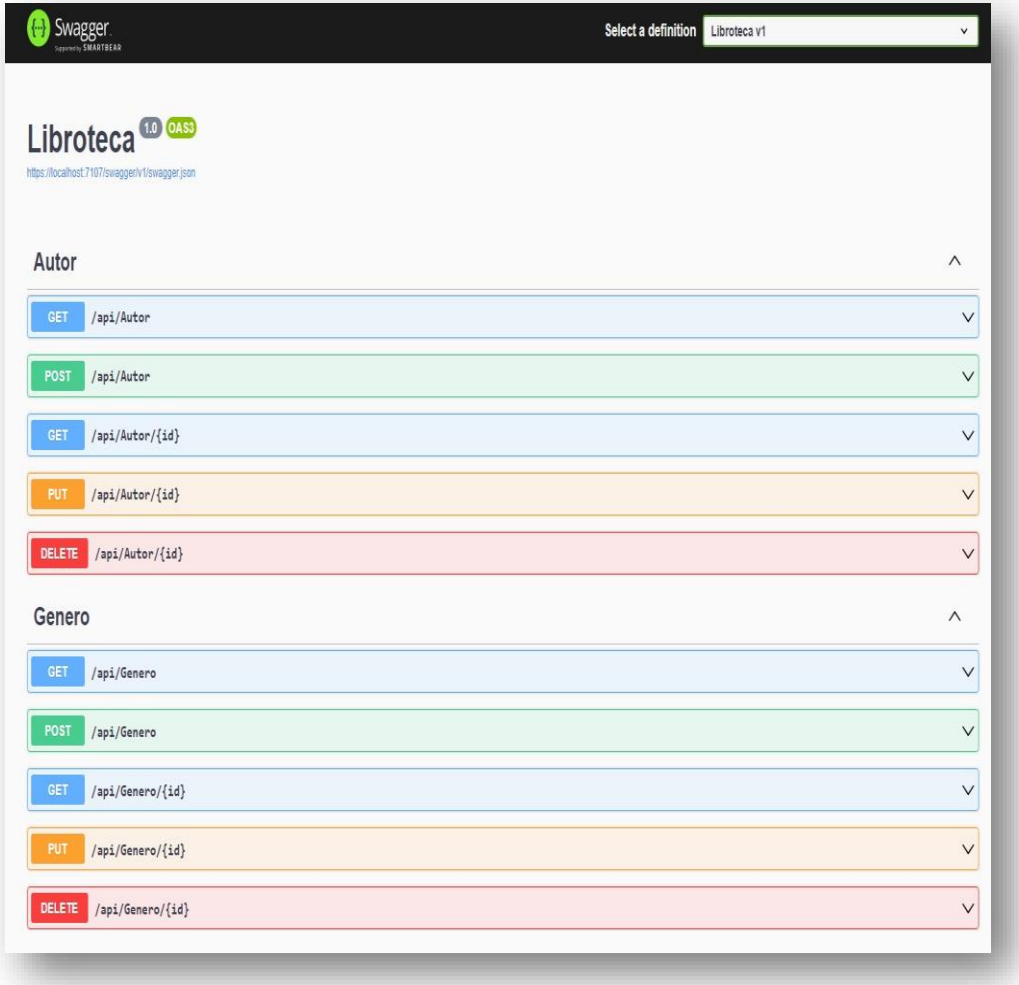
Todos Entienden swagger

Entonces, ¿qué es Swagger? Es una documentación online que se genera sobre una API. Por lo tanto, en esta herramienta podemos ver todos los endpoint que hemos desarrollado en nuestra API. También nos demuestra cómo son los elementos o datos que debemos pasar para hacer que funcione y nos permite probarlos directamente en su interfaz.

- La principal ventaja es que todo el mundo entiende Swagger
- Aun sin conocimientos se puede lograr entender su funcionamiento gracias al Swagger UI
- Incluso las máquinas pueden leerlo
- la documentación creada puede utilizarse directamente para automatizar procesos dependientes de APIs.

Swagger UI

La interfaz de usuario de Swagger



Swagger UI permite a cualquier persona, visualizar e interactuar con los recursos de la API. Se genera automáticamente a partir de escribir archivos que cumplan con la especificación OpenAPI, con esto, la documentación visual se crea y facilita la implementación de back-end y el consumo del lado del cliente.

Consiste en una infraestructura de visualización que puede analizar la especificación OpenAPI, hablaremos de esta especificación más adelante, y generar una consola de API para que los usuarios puedan aprender y ejecutar las APIs de forma rápida y sencilla. De esta manera nos encontramos con que todas las APIs creadas en nuestro proyecto están disponibles en una sola página.

Open API

Estandarizando el desarrollo de APIs

Inicios

Comenzó como parte del marco Swagger, siendo conocida como “Especificación swagger”. Swagger presentó las mejores prácticas de construcción de APIs y luego esas mejores prácticas se convirtieron en la especificación OpenAPI.

Actualidad

Proyecto separado de colaboración de código abierto de la Fundación Linux, Swagger y algunas otras herramientas que pueden generar código, documentación y casos de prueba con un archivo de interfaz, supervisado por la Iniciativa OpenAPI.

Flexible

No se tiene que volver a escribir el código de una API existente que no fue diseñada con estas especificaciones solo deben ajustarse para cumplir con este estándar global.

Ventaja

Permite a los desarrolladores de todo el planeta estandarizar el diseño de sus APIs.

OPENAPI
INITIATIVE

Microservicios

Son un enfoque arquitectónico y organizativo para el desarrollo de software

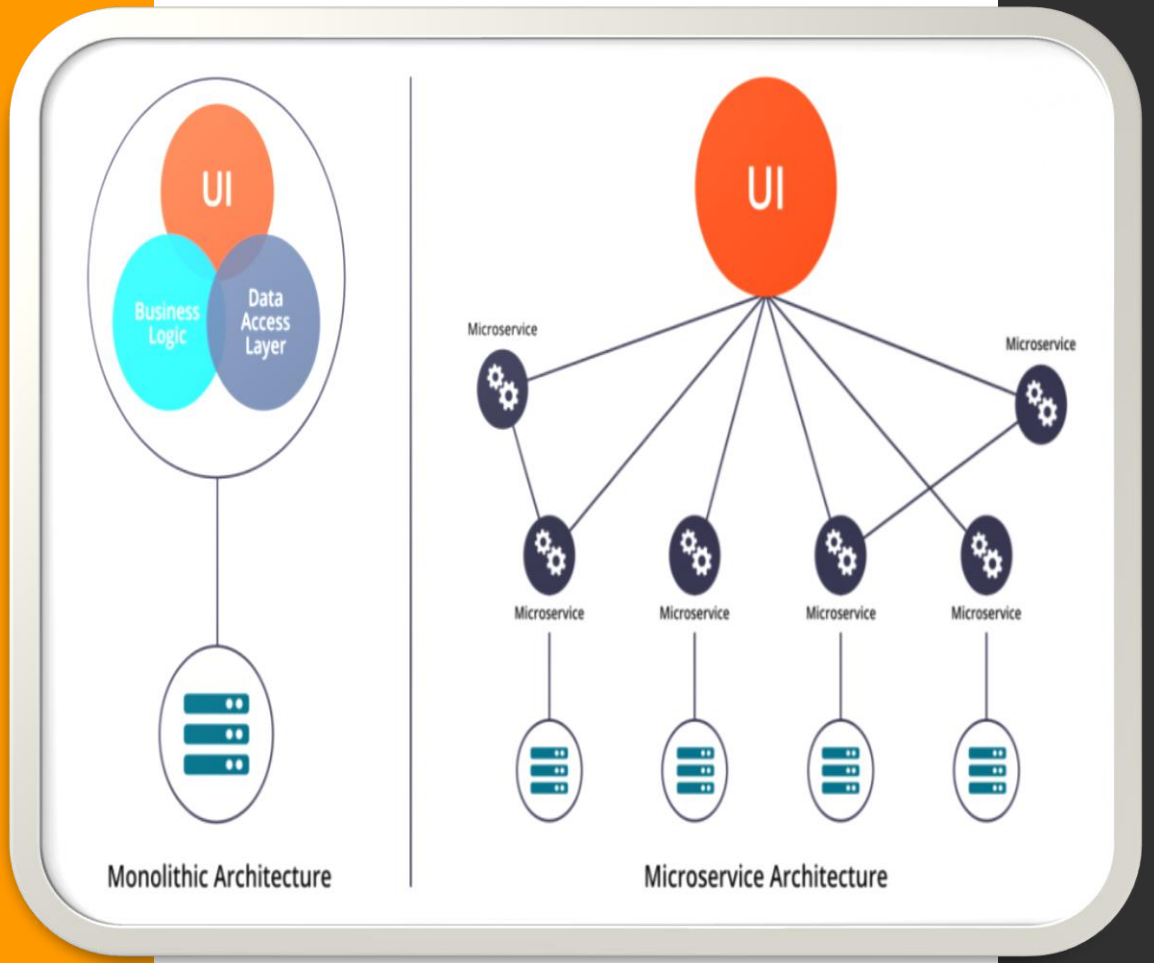
La arquitectura que impulsa el crecimiento

Despliegue ágil, escalabilidad y autonomía

Las arquitecturas de microservicios hacen que las aplicaciones sean más fáciles de escalar y más rápidas de desarrollar. Esto permite la innovación y acelera el tiempo de comercialización. La arquitectura de microservicios se considera un reemplazo moderno y flexible del modelo de desarrollo más tradicional de la arquitectura monolítica.

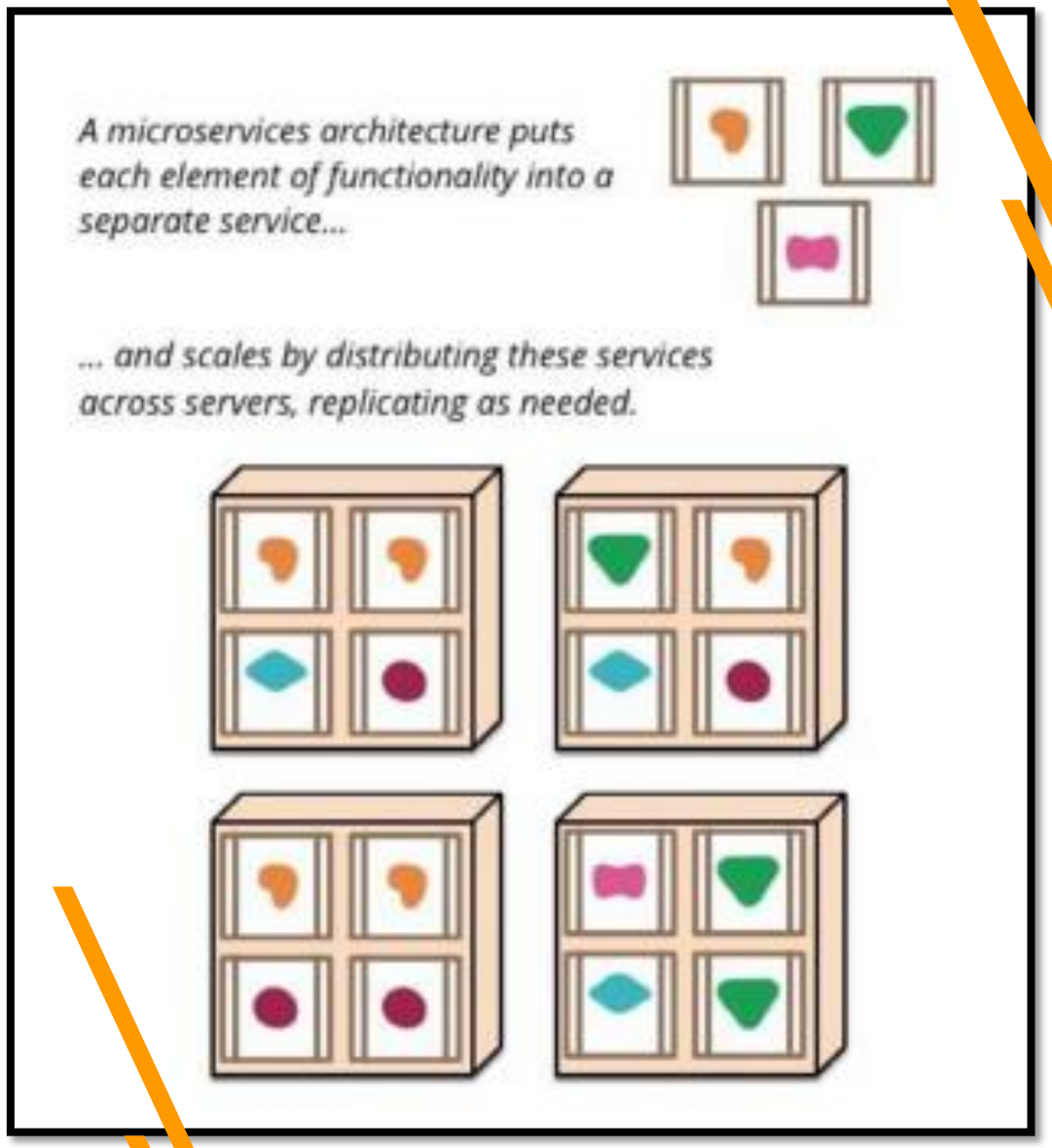
Monolítico

En una arquitectura monolítica, generalmente hay un servidor virtualizado o físico asignado a cada aplicación y esos servidores siempre están en ejecución. La disponibilidad y el escalamiento de la aplicación dependen completamente del hardware subyacente, que es una entidad fija.



Microservicios

Pueden operar varias instancias en un solo servidor o en varios servidores, a medida que los recursos escalan de forma dinámica para admitir las demandas de la carga de trabajo. Los microservicios individuales suelen estar en contenedores para mejorar la portabilidad y la escalabilidad.



Microservicios y sus Caracteristicas

01

Autónomos

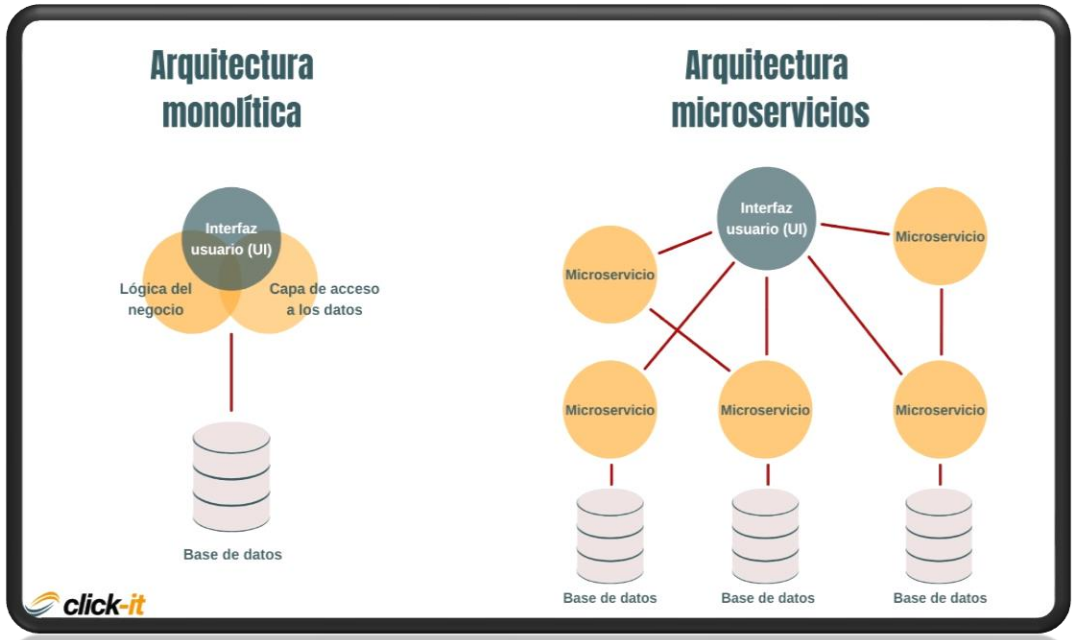
Cada servicio componente en una arquitectura de microservicios se puede desarrollar, implementar, operar y escalar sin afectar el funcionamiento de otros servicios. Los servicios no necesitan compartir ninguno de sus códigos o implementaciones con otros servicios.

02

Especializados

Cada servicio está diseñado para un conjunto de capacidades y se enfoca en resolver un problema específico.

Microservicios y sus Befenicios



01

Agilidad

Fomentan una organización de equipos pequeños e independientes que se apropian de los servicios

02

Escalado flexible

Permiten que cada servicio se escale de forma independiente para satisfacer la demanda de la aplicación que respalda

03

Implementación sencilla

Permiten la integración y la entrega continuas, lo que facilita probar nuevas ideas y revertirlas si algo no funciona

04

Libertad tecnológica

Los equipos tienen la libertad de elegir la mejor herramienta para resolver sus problemas específicos

05

Código reutilizable

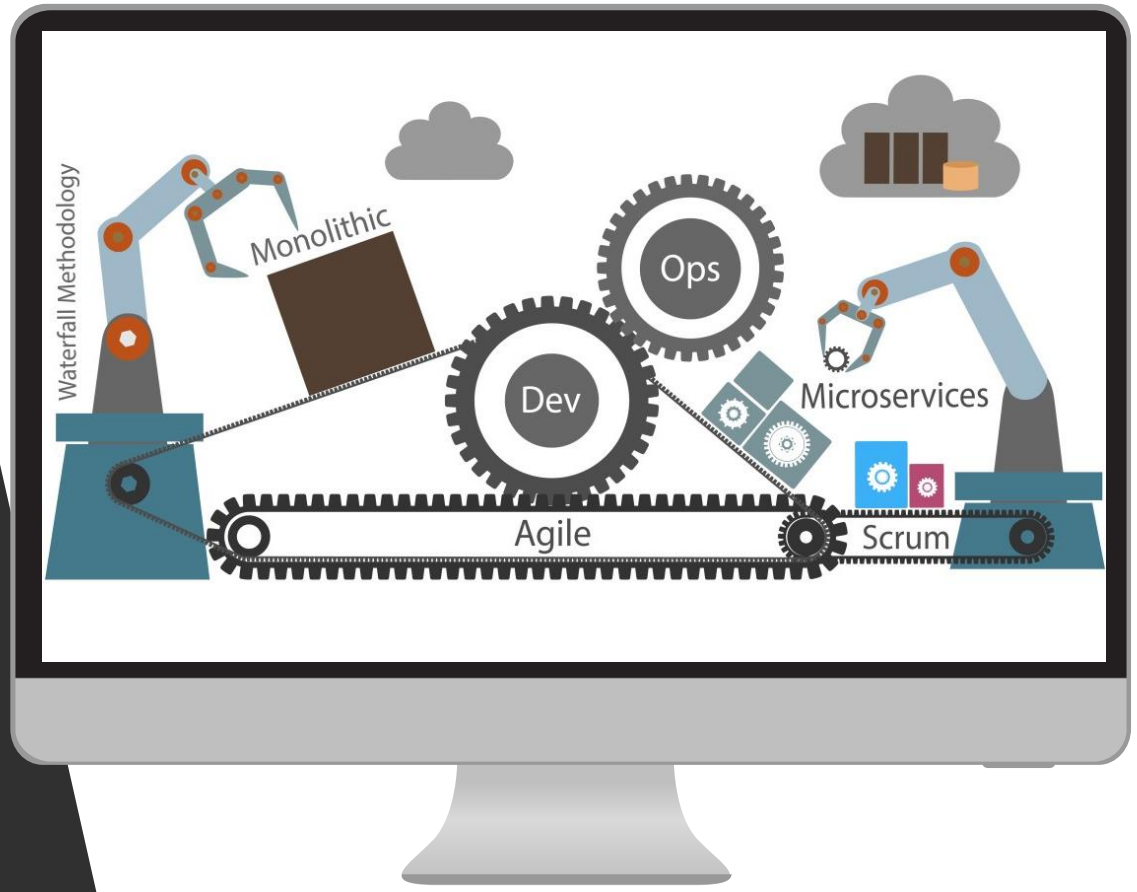
Un servicio escrito para una determinada función se puede usar como un componente básico para otra característica

06

Resistencia

La independencia del servicio aumenta la resistencia de una aplicación a los errores

Impacto en la transformación digital



Actualidad

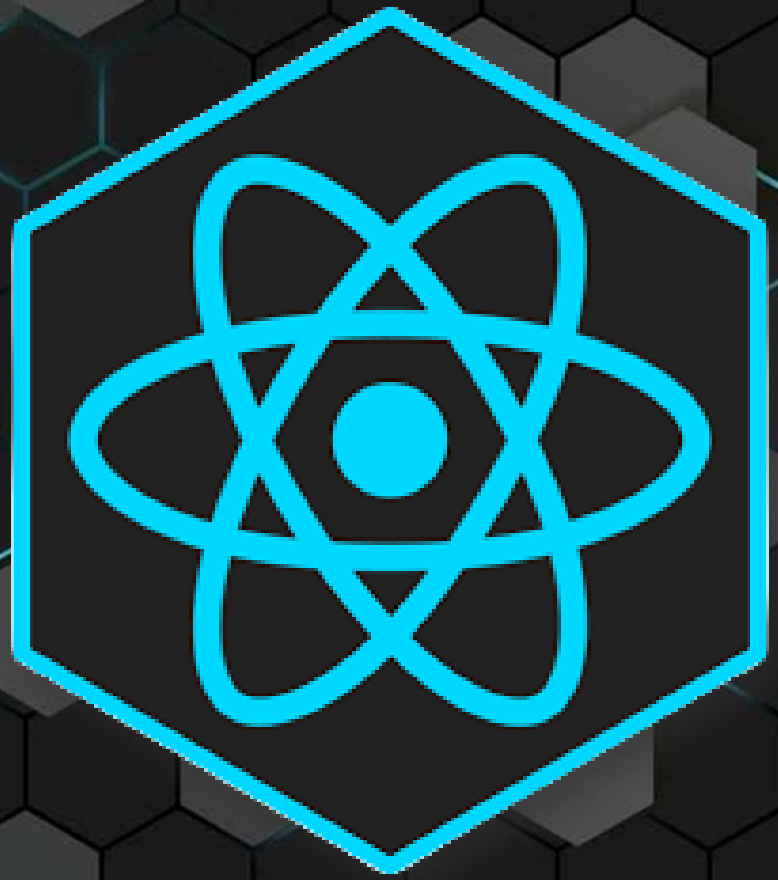
En los últimos años las empresas están migrando los datos y las cargas de trabajo a la nube por todas las ventajas que ofrece esta tecnología

Usados como motor al tratarse de arquitecturas muy escalables, flexibles y que permiten la innovación.

Motivo

Los atrae la promesa de un método para acelerar los cambios en el software, sin introducir peligro innecesario para el negocio.

En el diseño organizacional, el objetivo es descentralizar la autoridad de decisión.



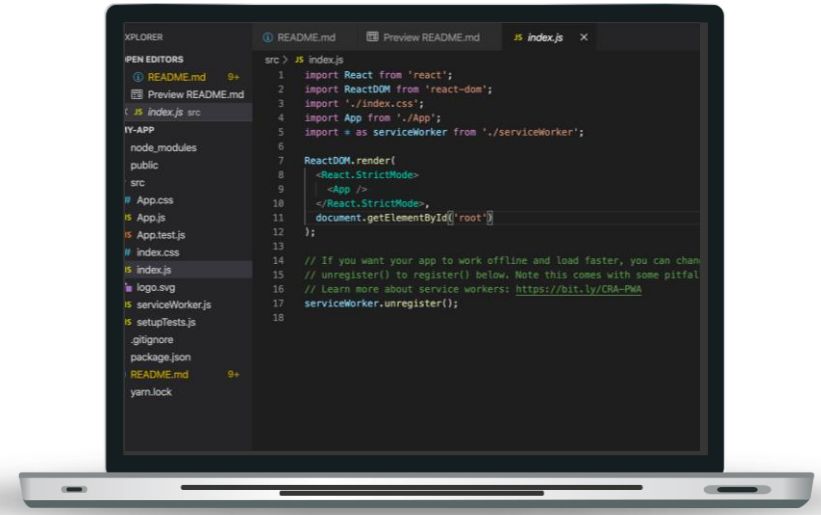
React

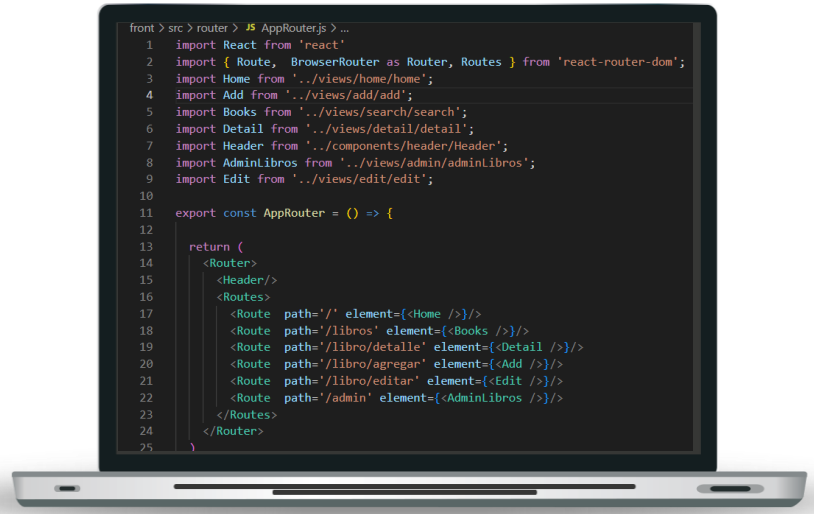
Es una librería open source de JavaScript para desarrollar interfaces de usuario

También es una
Arquitectura

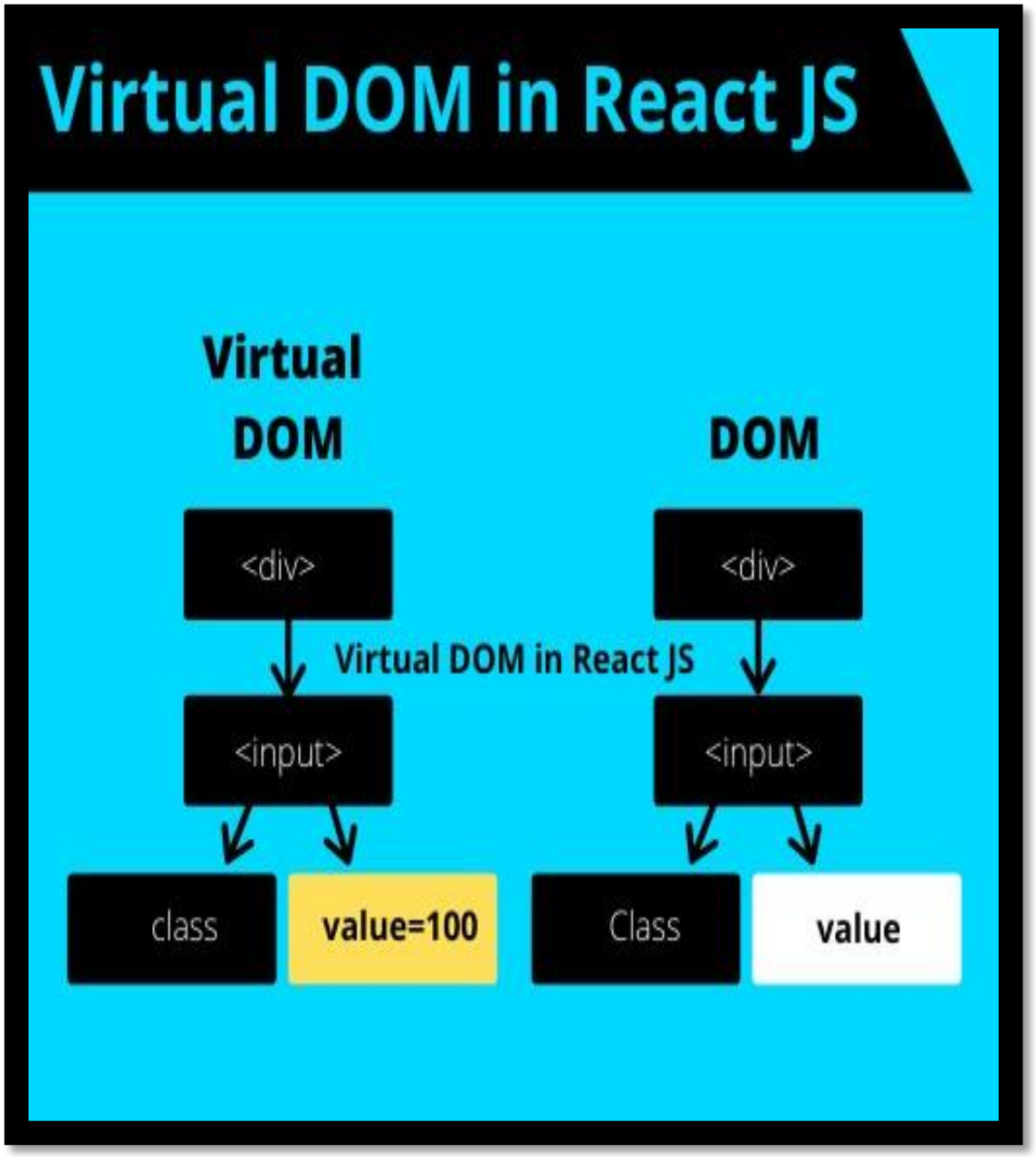
Desarrolla Aplicaciones Web Modernas y Dinámicas

React cuenta con un óptimo desempeño que se encarga de actualizar y renderizar los cambios realizados de forma automática. Esta cualidad permite a los programadores desarrollar sus códigos sin mayores contratiempos en el modelo en objetos para la representación de documentos (DOM). Entonces podemos ver que react ofrece grandes beneficios en performance, modularidad y promueve un flujo muy claro de datos y eventos, facilitando la planeación y desarrollo de apps complejas.





Fue lanzada en el año 2013 y desarrollada por Facebook, quienes también la mantienen actualmente junto a una comunidad de desarrolladores independientes y compañías.



El Secreto De react

El secreto de ReactJS para tener un performance muy alto, es que implementa algo llamado Virtual DOM y en vez de renderizar todo el DOM en cada cambio, que es lo que normalmente se hace, este hace los cambios en una copia en memoria y después usa un algoritmo para comparar las propiedades de la copia en memoria con las de la versión del DOM y así aplicar cambios exclusivamente en las partes que varían.

Header.jsx

front > src > components > header > Header.jsx > Header > useEffect() callback

```jsx
import React, { useEffect, useState } from 'react';
import { NavLink, useNavigate } from 'react-router-dom';

import '../header/header.css';

const Header = () => {
  const navigate = useNavigate();
  const [scrolled, setScrolled] = useState(false);

  useEffect(() => {
    const handleScroll = () => {
      if (window.scrollY > 0) {
        setScrolled(true);
      } else {
        setScrolled(false);
      }
    };

    window.addEventListener('scroll', handleScroll);

    return () => {
      window.removeEventListener('scroll', handleScroll);
    };
  }, []);

  return (
      <nav  className={`navbar navbar-expand-lg justify-content-center fixed-top px-5 py-4 ${scrolled ? 'scrolled shadow-bottom' : ''}`}>
        <div className="container-fluid row">
          <div className="col-6 text-start">
            <a className="navbar-brand logo fw-bold fs-2" href="/">Libroteca</a>
          </div>

          <button className="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
              aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span className="navbar-toggler-icon"><i className="bi bi-list fs-1"></i></span>
          </button>

          <div className="col-6 collapse navbar-collapse my-2 my-lg-0 justify-content-end" id="navbarNav">
            <ul className="navbar-nav text-center mt-md-3 mt-lg-0 flex-md-row flex-md-nowrap justify-content-center">
              <li className="nav-item me-lg-3">
                <NavLink to="/libros" className='nav nav-link fs-5 underline'>Libros</NavLink>
              </li>
              <li className="nav-item me-lg-3">
                <a className='nav-link fs-5 underline' href="https://localhost:7107/swagger/index.html" target="_blank">Documentación</a>
              </li>
              <li className="nav-item me-lg-3">
                <a className='nav-link fs-5 underline' href="#">Nosotros</a>
              </li>
            </ul>
            <button className='btn btn-primary rounded-pill fs-5 px-3 py-2' onClick={() => navigate('/admin')}>Administrar</button>
          </div>
        </div>
      </nav>
  );
}

export default Header;
```

Características

Antes de mencionar las principales características, hay que mencionar la más importante la cual es que ReactJS promueve el flujo de datos en un solo sentido.

Lenguaje JSX

La sintaxis que usa React es el JavaScript XML (JSX), el cual es una combinación del lenguaje HTML y el JavaScript, por lo que también se considera una extensión.

Componentes

Estos son partes de la interfaz del usuario que es independiente y que se puede reutilizar en otro lugar del sitio web.

Ciclos de vida

Dentro de React los componentes pasan por una serie de etapas durante su creación. Regularmente se divide en tres fases esenciales: montaje, utilización y desmontaje.

GRACIAS