

**Resultado de investigación**

**Arquitectura de Software**

**Integrantes**

Maria Jose Avellaneda

Ana María Ortegón Sepúlveda

Felipe García Castiblanco

**Profesor**

Andres Armando Sanchez Martin

**Pontificia Universidad Javeriana**

Bogotá

18 de marzo, 2024

1. **Definición, historia y evolución**
   1. **GraphQL**

GraphQL surgió como una respuesta a las demandas cambiantes de la industria, especialmente con el crecimiento del uso de dispositivos móviles. Fue desarrollado originalmente por Facebook en 2012 por Lee Byron, y su lanzamiento público fue en 2015 cuando inicio a convertirse en una tecnología ampliamente reconocida y adoptada.

La problemática principal fue la transición de la web a dispositivos móviles y aunque todo se quería lograr con HTML5 ya que prometía un futuro más interactivo, la realidad era que las plataformas web tradicionales basadas en servidores no ofrecían el rendimiento necesario en términos de consumo de red para satisfacer las expectativas de los usuarios.

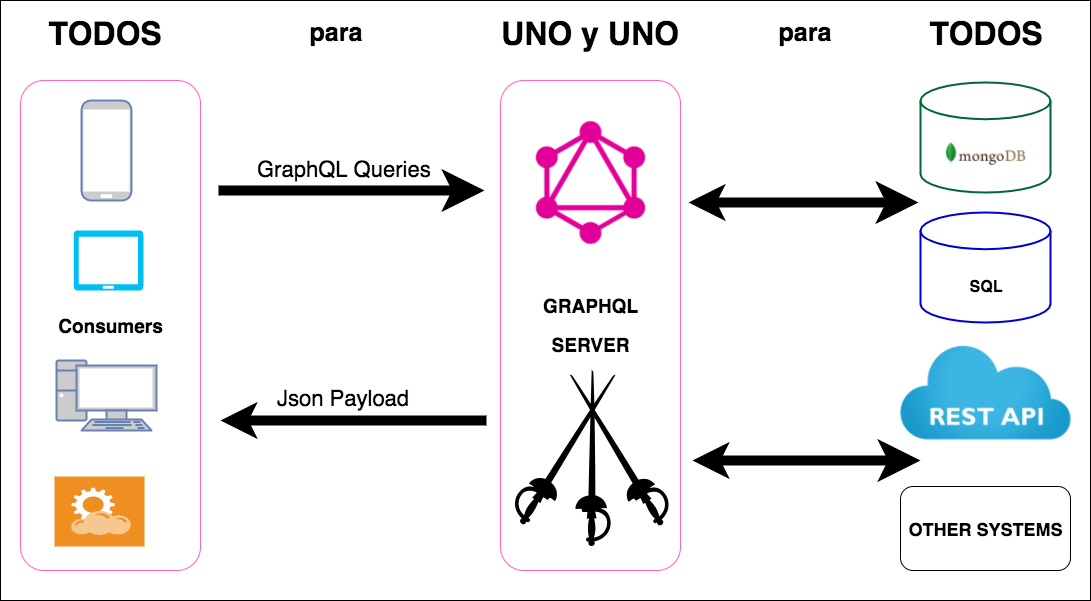
Esta situación llevó a un cambio en el paradigma de desarrollo, con un énfasis renovado en la construcción de aplicaciones nativas para dispositivos móviles. Sin embargo, esta nueva orientación presentaba su propio conjunto de desafíos, particularmente en el manejo eficiente de grandes volúmenes de datos y estructuras complejas.

Aquí es donde nace GraphQL, ofreciendo una manera eficiente y flexible de solicitar y entregar datos, era solución ideal para las necesidades de las aplicaciones orientadas a dispositivos móviles. Con la capacidad para permitir a los clientes solicitar exactamente los datos que necesitan, sin exceso de carga de información innecesaria, lo convirtió en una herramienta invaluable para optimizar el rendimiento en redes móviles y mejorar la experiencia del usuario.

Ahora, haciendo énfasis en una definición formal, GraphQL es un lenguaje de consulta diseñado para interactuar con APIs de datos desde el frontend hasta la capa de tiempo de ejecución del backend. Se destaca por su flexibilidad, eficiencia y naturaleza declarativa, lo que lo convierte en una herramienta fundamental en el desarrollo de aplicaciones modernas. Además de resolver problemas técnicos, GraphQL también aborda desafíos de comunicación al permitir que cada desarrollador defina claramente la estructura de los datos requeridos, cómo acceder a estos y qué datos se deben retornar. Una característica notable es su capacidad para ser consumido por múltiples clientes, no solo aquellos que utilizan GraphQL directamente, lo que amplía su alcance y versatilidad en diversos entornos de desarrollo.

Cuenta con una compatibilidad con más de 20 lenguajes de programación, siendo un lenguaje tipado que permite consultas precisas y evita la sobrecarga de datos no deseados. GraphQL no está vinculado a una base de datos específica, lo que ofrece libertad en la elección de tecnologías de almacenamiento de datos. Al permitir la especificación precisa de los datos necesarios, GraphQL proporciona respuestas rápidas y eficientes, reduciendo la demanda de ancho de banda y mejorando el rendimiento en caché. Esto se traduce en una reducción significativa en el tiempo de ejecución de las consultas, una gestión más eficiente de los recursos y una experiencia de usuario mejorada en términos de velocidad y fluidez en la interacción con la aplicación.

Con la siguiente imagen se puede apreciar una descripción grafica de GraphQL según [1]:



*Figura. Representación gráfica de operación de GraphQL*

* 1. **Express**

Express.js fue creado por TJ Holowaychuk y lanzado al público el 22 de mayo de 2010. En 2016, Express.js pasó al dominio de la fundación Node.js.A medida que Node.js ganaba popularidad, también lo hacía Express.js, gracias a su flexibilidad y potencia, por lo mismo los desarrolladores adoptaron rápidamente el framework.

Node.js, es un entorno de ejecución multiplataforma que establece un ambiente propicio para ejecutar JavaScript del lado del servidor. Express.js es un framework de backend de código abierto, ampliamente reconocido como uno de los más populares para el desarrollo web basado en Node.js.

Express.js proporciona una capa de abstracción que simplifica el proceso de desarrollo de backend, permitiendo crear aplicaciones web y APIs de manera eficiente. Destaca por su simplicidad y facilidad de uso, lo que lo convierte en una opción atractiva para desarrolladores de todos los niveles de experiencia. Una de las características distintivas de Express.js es su enfoque en la escalabilidad, mantenibilidad y alto rendimiento de las aplicaciones backend. Esto se logra mediante la gestión de errores, enrutamiento, sesiones y la facilidad en el manejo de solicitudes y respuestas HTTP.

Este framework ofrece una amplia gama de herramientas y plugins que facilitan diversos procesos de desarrollo, lo que contribuye a acelerar el ciclo de desarrollo y mejorar la productividad de cada equipo. Además, Express.js se destaca por su rendimiento y flexibilidad, permitiendo ejecutar cualquier código y adaptarse fácilmente a las necesidades específicas de cada proyecto. Además, una de las características más poderosas de Express.js es su sistema de middleware, que permite ejecutar una serie de funciones middleware en secuencia para manipular cambios en la solicitud y la respuesta de manera eficiente. Express.js se integra fácilmente con una variedad de bases de datos y tecnologías, lo que lo hace altamente compatible y versátil. Todo esto, combinado con su escritura en JavaScript, lo convierte en una opción atractiva y eficiente para el desarrollo de aplicaciones web y APIs escalables y robustas.

* 1. **VueJS**

Definición

Es un framework progresivo basado en JavaScript diseñado para la construcción de interfaces de usuario. Su librería principal se enfoca específicamente en la manipulación y presentación de datos visuales (capa de visualización).

Una de las características más atractivas de Vue.js es su tamaño compacto, contando con solo 18 KB de peso. Esta característica lo hace altamente eficiente para descargas rápidas y su implementación en dispositivos con restricciones de memoria.

Historia

Evan You, un ingeniero de software que anteriormente trabajaba en Google, es el creador de Vue.js, un framework que inicialmente desarrolló como un proyecto personal en 2013. La motivación principal detrás de Vue.js fue fusionar las mejores características de otros frameworks como Angular y React en una solución más liviana y fácil de usar.

Evan, luego de su experiencia utilizando AngularJS en varios proyectos en Google, reflexionó sobre la posibilidad de extraer las partes que más le gustaban de Angular para construir algo más ligero. Este pensamiento lo llevó a iniciar el desarrollo de Vue.js. El primer compromiso de código fuente en el proyecto se realizó en julio de 2013, y Vue fue lanzado oficialmente en febrero de 2014.

Evolución

Vue.js lanzó su primera versión estable en febrero de 2014, marcando el inicio de un crecimiento continuo en popularidad y adopción dentro de la comunidad de desarrollo web. A lo largo de su historia, Vue.js ha alcanzado diversos hitos y versiones significativas.

|  |  |  |
| --- | --- | --- |
| **Versión** | **Fecha de Liberación** | **Titulo** |
| 0.6 | 08 de diciembre de 2013 | [0.6.0: VueJS](https://github.com/vuejs/vue/releases/tag/0.6.0). |
| 0.7 | 24 de diciembre de 2013 | [v0.7.0](https://github.com/vuejs/vue/releases/tag/v0.7.0). |
| 0.8 | 27 de enero de 2014 | [v0.8.0](https://github.com/vuejs/vue/releases/tag/v0.8.0) |
| 0.9 | 25 de febrero de 2014 | [Animatrix](https://es.wikipedia.org/wiki/The_Animatrix) |
| 0.10 | 23 de marzo de 2014 | [Blade Runner](https://es.wikipedia.org/wiki/Blade_Runner) |
| 0.11 | 07 de noviembre de 2014 | [Cowboy Bebop](https://es.wikipedia.org/wiki/Cowboy_Bebop) |
| 0.12 | 12 de junio de 2015 | [Dragon Ball](https://es.wikipedia.org/wiki/Dragon_Ball) |
| 1.0 | 27 de octubre de 2015 | [Evangelion](https://es.wikipedia.org/wiki/Neon_Genesis_Evangelion) |
| 2.0 | 30 de septiembre de 2016 | [Ghost in the Shell](https://es.wikipedia.org/wiki/Ghost_in_the_Shell) |
| 2.1 | 22 de noviembre de 2016 | [Hunter × Hunter](https://es.wikipedia.org/wiki/Hunter_%C3%97_Hunter) |
| 2.2 | 26 de febrero de 2017 | [Initial D](https://es.wikipedia.org/wiki/Initial_D) |
| 2.3 | 27 de abril de 2017 | [JoJo's Bizarre Adventure](https://es.wikipedia.org/wiki/JoJo%27s_Bizarre_Adventure) |
| 2.4 | 13 de julio de 2017 | [Kill la Kill](https://es.wikipedia.org/wiki/Kill_la_Kill) |
| 2.5 | 13 de octubre de 2017 | Level E |
| 2.6 | 04 de febrero de 2019 | [Macross](https://es.wikipedia.org/wiki/Macross) |
| 2.7 | 1 de julio de 2022 | Naruto |
| 3.0 | 18 de septiembre de 2020 | [One Piece](https://es.wikipedia.org/wiki/One_Piece) |
| 3.1 | 7 de junio de 2021 | Pluto |
| 3.2 | 5 de agosto de 2021 | Quintessential Quintuplets |

* 1. **MySQL**

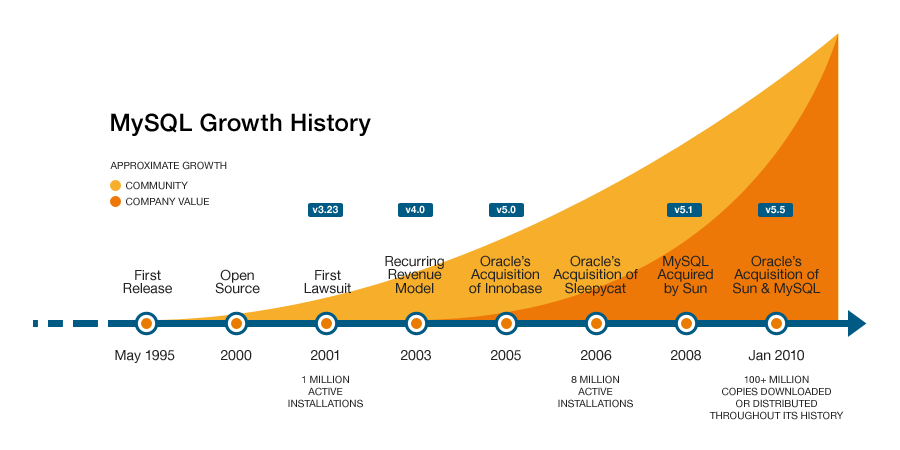
Definición  
MySQL es un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto que cuenta con el respaldo de Oracle y se basa en el lenguaje de consulta estructurado (SQL). Es ampliamente compatible con diversas plataformas, incluyendo Linux, UNIX y Windows.

Este sistema es un componente fundamental de la pila empresarial de código abierto conocida como LAMP. LAMP es una plataforma de desarrollo web que se basa en Linux como sistema operativo, Apache como servidor web, MySQL como sistema de gestión de bases de datos relacionales, y PHP como lenguaje de scripting orientado a objetos (aunque a veces se utiliza Perl o Python en lugar de PHP).

Historia

MySQL fue originalmente desarrollado por MySQL AB, una empresa fundada por David Axmark, Allan Larsson y Michael Widenius. En 2008, MySQL AB fue adquirida por Sun Microsystems por $1 mil millones, aunque esta decisión fue objeto de críticas por parte de los cofundadores de MySQL AB. Sun Microsystems consideraba a MySQL como una parte vital de la economía de Internet. Sin embargo, la adquisición por parte de Sun no fue particularmente fructífera, y en 2009, se llegó a un acuerdo según el cual Oracle Corporation adquiriría Sun Microsystems, incluyendo los derechos de autor y la marca registrada de MySQL. Este acuerdo fue aprobado por el gobierno de los EE. UU. en agosto de 2009. Posteriormente, Michael Widenius dejó Sun Microsystems y desarrolló una versión bifurcada (fork) de MySQL llamada MariaDB. Los forks son proyectos relacionados que pueden considerarse como versiones alternativas de MySQL estándar.

Evolución



Gráfico

Descripción generada automáticamente

Imagen que contiene Diagrama

Descripción generada automáticamente

1. **Relación entre los temas**
   1. **GraphQL**

GraphQL y Express.js:

* La relación entre GraphQL y Express.js se establece al integrar GraphQL como una capa de enrutamiento en Express.js. Esto implica definir rutas en Express.js que correspondan a puntos de entrada de GraphQL y utilizar middleware específico para procesar estas consultas. Así, Express.js actúa como un servidor que maneja las solicitudes GraphQL entrantes, ejecutando las consultas pertinentes y devolviendo los resultados adecuados al cliente.

GraphQL y MySQL:

* La relación entre GraphQL y MySQL se fundamenta en el uso de MySQL como fuente de datos para las consultas GraphQL. Esto se logra mediante un servidor GraphQL que traduce las consultas GraphQL recibidas en consultas SQL para interactuar con la base de datos MySQL. Por lo tanto, GraphQL se encarga de realizar consultas a MySQL, recuperando, creando, actualizando o eliminando datos según lo solicitado por el cliente a través de las consultas GraphQL.

GraphQL y Vue.js:

* La relación entre GraphQL y Vue.js radica en la comunicación entre el frontend y el backend para obtener y mostrar datos específicos en la interfaz de usuario. Vue.js envía consultas GraphQL al servidor para solicitar datos específicos necesarios para la interfaz de usuario. Estas consultas son procesadas en el servidor GraphQL, que a su vez interactúa con MySQL u otras fuentes de datos para obtener los datos solicitados. Finalmente, los datos devueltos en formato JSON son renderizados dinámicamente por Vue.js en la interfaz de usuario.
  1. **Express**

Express.js y GraphQL:

* + La relación entre Express.js y GraphQL se manifiesta al utilizar Express.js como servidor para una API GraphQL. Esto implica definir rutas y middleware en Express.js para gestionar las solicitudes y respuestas de GraphQL. Express.js se encarga de manejar las consultas GraphQL entrantes, ejecutando las consultas correspondientes y devolviendo los resultados al cliente en formato adecuado.

Express.js y MySQL:

* + La relación entre Express.js y MySQL se establece al utilizar Express.js como un middleware para interactuar con la base de datos MySQL. Express.js define rutas y controladores que gestionan las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en los datos almacenados en MySQL. De esta manera, Express.js actúa como un intermediario entre el frontend y la base de datos MySQL, manejando las solicitudes HTTP y ejecutando consultas SQL para acceder y modificar los datos según sea necesario.

Express.js y Vue.js:

* + La relación entre Express.js y Vue.js se concreta al utilizar Express.js como servidor para servir los archivos estáticos de una aplicación Vue.js y proporcionar las rutas necesarias para la comunicación entre el frontend y el backend. Vue.js envía solicitudes HTTP al backend Express.js para realizar diversas operaciones, como autenticación de usuarios, acceso a datos y cualquier otra funcionalidad que requiera interacción con el servidor. De esta manera, Express.js y Vue.js trabajan juntos para crear una aplicación web completa y funcional.
  1. **VueJS**

Vue.js y el web service (servidor) en general

* La relación entre Vue.js y el servidor web en general implica definir y mantener una comunicación bidireccional entre el frontend y el backend para garantizar que la aplicación funcione correctamente y tenga acceso a los recursos necesarios.
* Ejemplo de relación: una aplicación web desarrollada con Vue.js que permite a los usuarios registrarse e iniciar sesión. Cuando un usuario completa el formulario de registro, Vue.js enviará una solicitud HTTP POST al servidor para crear una nueva cuenta de usuario. El servidor recibirá esta solicitud, procesará los datos recibidos y almacenará la información del usuario en la base de datos. Luego, cuando el usuario intenta iniciar sesión, Vue.js enviará otra solicitud HTTP POST al servidor para autenticar las credenciales proporcionadas. El servidor verificará las credenciales en la base de datos y responderá con un token de acceso si la autenticación es exitosa.

Vue.js y MySQL

* La relación entre Vue.js y MySQL se establece a través de solicitudes HTTP enviadas desde Vue.js al backend (que puede ser manejado por Express.js). Estas solicitudes incluyen operaciones como la recuperación, creación, actualización y eliminación de datos, que son procesadas por el backend y se traducen en consultas SQL para interactuar con la base de datos MySQL.
* Ejemplo de relación: una aplicación de lista de tareas desarrollada con Vue.js. Cuando el usuario agrega una nueva tarea a la lista, Vue.js enviará una solicitud HTTP POST al servidor. El servidor, que está utilizando Express.js, recibirá esta solicitud y ejecutará una consulta SQL para insertar la nueva tarea en la base de datos MySQL. Cuando el usuario actualiza el estado de una tarea (por ejemplo, marcándola como completada), Vue.js enviará una solicitud HTTP PUT al servidor, que actualizará el estado correspondiente en la base de datos MySQL.

Vue.js y GraphQL

* La relación entre Vue.js y GraphQL se basa en definir consultas GraphQL en el frontend para solicitar datos específicos, que luego son procesadas por el servidor GraphQL para devolver los datos solicitados en un formato JSON que se adapta exactamente a los requisitos del frontend.
* Ejemplo de relación: una aplicación de redes sociales desarrollada con Vue.js. Cuando un usuario accede al perfil de otro usuario, Vue.js enviará una consulta GraphQL al servidor para solicitar información específica sobre ese usuario, como su nombre, foto de perfil y publicaciones recientes. El servidor GraphQL procesará esta consulta y devolverá los datos solicitados en un formato JSON que se adaptará exactamente a los requisitos del frontend de Vue.js.

Vue.js y Express.js

* La relación entre Vue.js y Express.js se basa en definir rutas en el servidor Express.js que manejen las solicitudes entrantes desde Vue.js y realicen las operaciones necesarias, como interactuar con la base de datos, procesar datos o devolver respuestas al frontend.
* Ejemplo de relación: una aplicación de comercio electrónico desarrollada con Vue.js. Cuando un usuario realiza una compra en el sitio web, Vue.js enviará una solicitud HTTP POST al servidor Express.js para procesar el pedido. El servidor Express.js recibirá esta solicitud, interactuará con la base de datos para actualizar el inventario de productos y realizará cualquier otra operación necesaria, como generar una confirmación de pedido que se enviará al usuario.
  1. **MySQL**

MySQL y Vue.js

* La relación entre MySQL y Vue.js se establece a través de solicitudes HTTP enviadas desde Vue.js al backend (posiblemente manejado por Express.js). Estas solicitudes incluyen operaciones como la recuperación, creación, actualización y eliminación de datos, que son procesadas por el backend y se traducen en consultas SQL para interactuar con la base de datos MySQL.

MySQL y GraphQL

* GraphQL actúa como una capa de abstracción sobre las bases de datos y otros servicios, proporcionando una forma flexible de solicitar y recibir datos según las necesidades específicas del cliente.
* MySQL, como sistema de gestión de bases de datos relacional, puede ser una fuente de datos para un servidor GraphQL. El servidor GraphQL se encarga de traducir las solicitudes GraphQL en consultas SQL para interactuar con la base de datos MySQL y devolver los datos solicitados en un formato estructurado.

MySQL y Express.js

* MySQL, como sistema de gestión de bases de datos relacional, puede ser utilizado en conjunto con Express.js para almacenar y recuperar datos en una aplicación web.
* En una aplicación Express.js, se pueden definir rutas y controladores que interactúen con la base de datos MySQL para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en los datos almacenados.

1. **Situaciones y/o problemas donde se pueden aplicar los temas.**
   1. **GraphQL**

Principalmente el uso de GraphQL destaca en:

* En aplicaciones móviles, donde la limitación del ancho de banda y las redes lentas son problemas comunes, GraphQL ofrece una solución al permitir la agrupación de múltiples solicitudes en una sola consulta. Esto reduce el número total de solicitudes realizadas, disminuyendo la carga en el servidor y mejorando el rendimiento en condiciones de conectividad limitada.
* En entornos complejos con interacción entre bases de datos o sistemas heredados, GraphQL ayuda a reducir la complejidad y mejorar la eficiencia. Al encapsular la lógica de cómo obtener datos de diferentes recursos, GraphQL permite una integración más fluida sin la necesidad de especificar exactamente dónde se encuentran esos recursos.
* En el contexto de aplicaciones basadas en microservicios, GraphQL proporciona un enrutamiento eficiente de las solicitudes, gestión de errores y ejecución paralela de las mismas. Esto se traduce en una mayor agilidad y rendimiento, ya que GraphQL se encarga de dirigir las solicitudes a todos los microservicios asociados y devolver los resultados de manera optimizada.
  1. **Express**

Principalmente el uso de Express es:

* + - En cuanto desarrollar aplicaciones web de cualquier tamaño, desde pequeñas aplicaciones hasta grandes sitios web, ya que proporciona un conjunto de funciones simples pero importantes para el manejo de solicitudes HTTP, manejo de rutas y la renderización de vistas.
    - Puede actuar como un proxy inverso para enrutar solicitudes a diferentes servidores backend según ciertas reglas o como un servidor de middleware para interceptar solicitudes y realizar acciones específicas antes de pasarlas a la aplicación principal.
    - Para aplicaciones de una sola página que utilizan frameworks de frontend como Angular, React o Vue.js, Express.js se puede utilizar para servir los archivos estáticos de la aplicación y manejar todas las solicitudes de API necesarias.
  1. **VueJS**

Desarrollo Rápido de Prototipos:

* + - Vue.js es ideal para crear prototipos rápidos y experimentar con ideas. Su sintaxis simple y su enfoque gradual permiten desarrollar rápidamente interfaces de usuario funcionales.

Aplicaciones de Página Única (SPAs):

* + - Vue.js es perfecto para construir SPAs. Puedes crear una aplicación completa sin necesidad de recargar la página, lo que mejora la experiencia del usuario.

Componentes Reutilizables:

* + - Vue.js se basa en componentes, lo que facilita la creación de piezas de interfaz reutilizables. Puedes construir componentes independientes y combinarlos para formar aplicaciones más grandes.

Integración con Otras Tecnologías:

* + - Vue.js se integra fácilmente con otras bibliotecas y frameworks. Puedes usarlo junto con herramientas como Vuex para la gestión del estado o Vue Router para la navegación.

Optimización del Rendimiento:

* + - Vue.js ofrece una excelente reactividad y un sistema de actualización eficiente. Puedes optimizar el rendimiento de tus aplicaciones sin complicaciones.

Aplicaciones Empresariales:

* + - Vue.js es una excelente opción para aplicaciones empresariales. Su curva de aprendizaje suave y su flexibilidad lo hacen atractivo para equipos de desarrollo.

Pequeños Proyectos y Páginas Web:

* + - Si necesitas construir una pequeña aplicación o una página web, Vue.js es liviano y fácil de configurar. No requiere una infraestructura compleja.

Complemento en Proyectos Existentes:

* + - Puedes agregar Vue.js a proyectos existentes sin problemas. Incluso puedes usarlo en partes específicas de una página sin afectar el resto del código.

Aplicaciones Híbridas y Móviles:

* + - Con herramientas como Vue Native o Quasar, puedes utilizar Vue.js para crear aplicaciones móviles y híbridas.
  1. **MySQL**

Aplicaciones web tradicionales:

* MySQL es ideal para aplicaciones web tradicionales que requieren almacenamiento y gestión de datos estructurados, como blogs, sitios web corporativos y portales de noticias.

Aplicaciones de comercio electrónico:

* En el comercio electrónico, MySQL se utiliza para almacenar información sobre productos, pedidos, clientes y transacciones, proporcionando una base sólida para la gestión de datos en tiendas en línea y plataformas de comercio electrónico.

Sistemas de gestión de contenidos (CMS):

* Los sistemas de gestión de contenidos como WordPress, Joomla y Drupal utilizan MySQL como base de datos subyacente para almacenar contenido, configuraciones de sitio y metadatos.

Aplicaciones de gestión de relaciones con el cliente (CRM):

* MySQL se utiliza en aplicaciones CRM para almacenar información sobre clientes, contactos, historiales de interacción y datos relacionados con ventas y marketing.

Aplicaciones de gestión de recursos empresariales (ERP):

* En los sistemas ERP, MySQL se utiliza para gestionar datos empresariales críticos, como inventario, recursos humanos, finanzas y logística.

Aplicaciones de análisis de datos:

* MySQL se utiliza en aplicaciones de análisis de datos para almacenar grandes volúmenes de datos transaccionales y de negocios, proporcionando una base para consultas complejas y análisis de datos.

Aplicaciones de redes sociales y colaboración:

* MySQL se utiliza en aplicaciones de redes sociales y colaboración para almacenar perfiles de usuarios, publicaciones, comentarios, mensajes y otros datos generados por los usuarios.

Aplicaciones móviles y juegos:

* MySQL se utiliza en el backend de aplicaciones móviles y juegos para gestionar usuarios, datos de juegos, configuraciones y otras funcionalidades relacionadas con la aplicación.

Aplicaciones de seguimiento y análisis de datos de IoT:

* En el Internet de las cosas (IoT), MySQL se utiliza en aplicaciones para recopilar, almacenar y analizar datos de sensores, dispositivos y sistemas conectados.

Aplicaciones de informes y análisis empresarial:

* MySQL se utiliza en aplicaciones de informes y análisis empresarial para almacenar datos transaccionales y de negocios, permitiendo la generación de informes, dashboards y análisis de datos en tiempo real.

utenticación y autorización de usuarios:

* En un web service, MySQL se utiliza para almacenar información de usuarios, como nombres de usuario, contraseñas cifradas y roles de acceso. Esto permite la autenticación de usuarios y la gestión de permisos de acceso a recursos protegidos.

Almacenamiento de datos de sesión:

* MySQL se utiliza para almacenar datos de sesión de usuarios en aplicaciones web, como tokens de sesión, información de inicio de sesión persistente y preferencias del usuario. Esto permite mantener el estado de la sesión del usuario a lo largo de las interacciones con el servicio web.

Gestión de contenido dinámico:

* En servicios web que proporcionan contenido dinámico, como blogs, foros y plataformas de medios sociales, MySQL se utiliza para almacenar publicaciones de usuarios, comentarios, mensajes y otros datos generados por los usuarios.

Almacenamiento de datos transaccionales:

* MySQL se utiliza para almacenar datos transaccionales en servicios web de comercio electrónico, como detalles de pedidos, información de productos, carritos de compra y datos de transacciones financieras.

Gestión de datos de formularios:

* En servicios web que procesan formularios web, MySQL se utiliza para almacenar datos enviados por usuarios, como información de contacto, respuestas a encuestas y registros de eventos.

Almacenamiento de datos de configuración y preferencias:

* MySQL se utiliza para almacenar datos de configuración y preferencias de usuario en servicios web, permitiendo a los usuarios personalizar su experiencia de usuario y ajustar la configuración de la aplicación según sus preferencias.

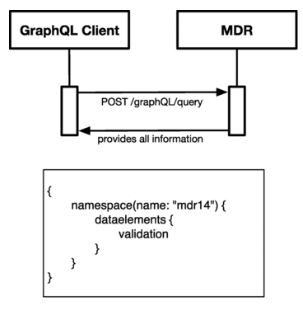
Seguimiento y análisis de datos:

* MySQL se utiliza para almacenar datos de seguimiento y análisis en servicios web, como registros de actividad del usuario, métricas de rendimiento, datos de análisis y registros de errores. Esto permite realizar análisis retrospectivos y tomar decisiones informadas sobre el rendimiento y la eficacia del servicio web.

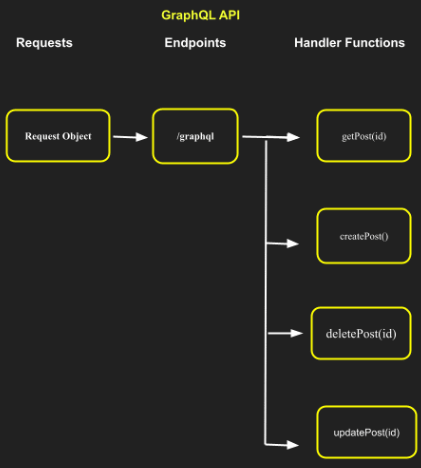
1. **Estilos y patrones**
   1. **GraphQL**

La comunicación en GraphQL, se basa en:

* + Un cliente puede enviar una única consulta que especifica exactamente los datos que necesita.
  + La consulta se estructura según el esquema GraphQL definido en el servidor, lo que permite al cliente recuperar solo los datos necesarios.
  + Esto significa que la cantidad de mensajes intercambiados entre el cliente y el servidor es mínima, ya que una sola consulta puede recuperar todos los datos necesarios para satisfacer la solicitud del cliente.

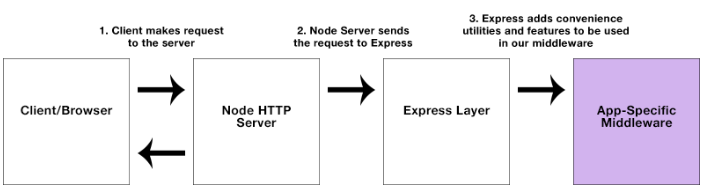


* + En el siguiente diagrama, se puede observar que solo hay una única ruta de acceso al recurso en el servidor GraphQL. Esto significa que, para interactuar con los datos y funcionalidades proporcionadas por el servidor GraphQL, los clientes solo necesitan hacer una solicitud a este punto final, en lugar de múltiples rutas dispersas. Esto simplifica la arquitectura y la gestión de las solicitudes, ya que toda la lógica de acceso a los datos se concentra en un solo punto.



* 1. **Express**

En términos más técnicos, Express utiliza los métodos HTTP (como GET, POST, PUT, DELETE, etc.) para definir las rutas y manejar las solicitudes entrantes desde el cliente. Luego, basado en la ruta y el método HTTP, se ejecuta el código correspondiente para procesar la solicitud y generar una respuesta.



Express.js simplifica el desarrollo de aplicaciones web y servicios de API basados en Node.js al proporcionar una capa de abstracción adicional sobre las características fundamentales de Node.js. Esta capa adicional permite una comunicación más eficiente entre el cliente y el servidor, al facilitar tareas como el envío de imágenes y la gestión de solicitudes HTTP. Además, Express promueve una arquitectura de código más modular y mantenible, en contraste con la naturaleza monolítica de los manejadores de rutas en Node.js. Esto se logra a través de la capacidad de Express para dividir el código en módulos más pequeños y fáciles de gestionar, lo que resulta en un código más limpio y estructurado.

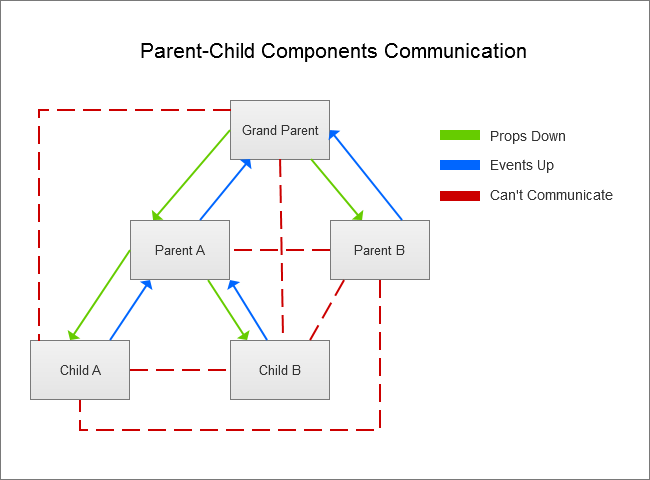
* 1. **VueJS**

Hay dos tipos de comunicación principales entre componentes en Vue.js

* + - Comunicación directa padre-hijo, basada en las estrictas relaciones padre-a-hijo e hijo-a-padre
    - Comunicación multicomponente, en la que un componente puede "hablar" con cualquier otro sin importar su relación.

Comunicación directa padre-hijo:

El modelo estándar para la comunicación de componentes que Vue.js admite de forma inmediata es el modelo padre-hijo, implementado con propiedades y eventos personalizados. El siguiente diagrama muestra cómo se ve este modelo en acción.



Comunicación multicomponente:

El modelo de comunicación entre padre-hijo rápidamente se vuelve engorroso y poco práctico a medida que nuestra aplicación se vuelve más compleja. El problema con el sistema de eventos props-eventos es que funciona directamente y está estrechamente acoplado al árbol de componentes. Los eventos de Vue no se activan, a diferencia de los eventos nativos, por lo que tenemos que enviarlos repetidamente hasta llegar a nuestro destino. Como resultado, nuestro código se aburre con demasiados despachadores y oyentes de eventos. Por lo tanto, deberíamos considerar el uso de un modelo de comunicación multicomponente en aplicaciones más complejas.

Diagrama

Descripción generada automáticamente

Estructuración de vuejs

Carpeta raíz

Texto

Descripción generada automáticamente

|  |  |
| --- | --- |
| index.html | Fichero [HTML](https://lenguajehtml.com/) principal de la página o aplicación. |
| .eslintrc.cjs | Fichero de configuración del linter de Javascript ESLint. |
| .gitignore | Fichero que indica los archivos que Git debe ignorar al hacer el versionado. |
| .prettierrc.json | Fichero de configuración del formateador de código Prettier. |
| README.md | Fichero Markdown donde se documenta información sobre el proyecto. |
| .env | Fichero de variables de entorno del proyecto. |
| package.json | Fichero de configuración del proyecto, usando NPM. [Más info en NPM](https://lenguajejs.com/npm/). |
| package-lock.json | Fichero histórico de versionado de apoyo para el package.json. [Más info en NPM](https://lenguajejs.com/npm/). |
| playwright.config.js | Fichero de configuración de Playwright, framework de tests end-to-end. |
| tsconfig.json | Fichero de configuración de TypeScript, un metalenguaje de Javascript con tipos. |
| vitest.config.js | Fichero de configuración de Vitest, framework de tests unitarios para usar con Vite. |
| vite.config.js | Fichero de configuración de Vite, automatizador de aplicaciones web Javascript. |

Carpeta public

Texto

Descripción generada automáticamente con confianza media

Carpeta src

Diagrama

Descripción generada automáticamente con confianza baja

* 1. **MySQL**

Arquitectura enfocada en php con MySQL:

La arquitectura cliente/servidor es la más comúnmente empleada en proyectos de gran escala. En este enfoque, la base de datos se divide en dos partes: el servidor, encargado de gestionar y almacenar los datos, y el cliente, responsable de realizar solicitudes para acceder a dichos datos. En el caso específico de PHP, el servidor corresponde al hardware donde se aloja la base de datos MySQL, mientras que el cliente consiste en la aplicación PHP que solicita información mediante consultas SQL.

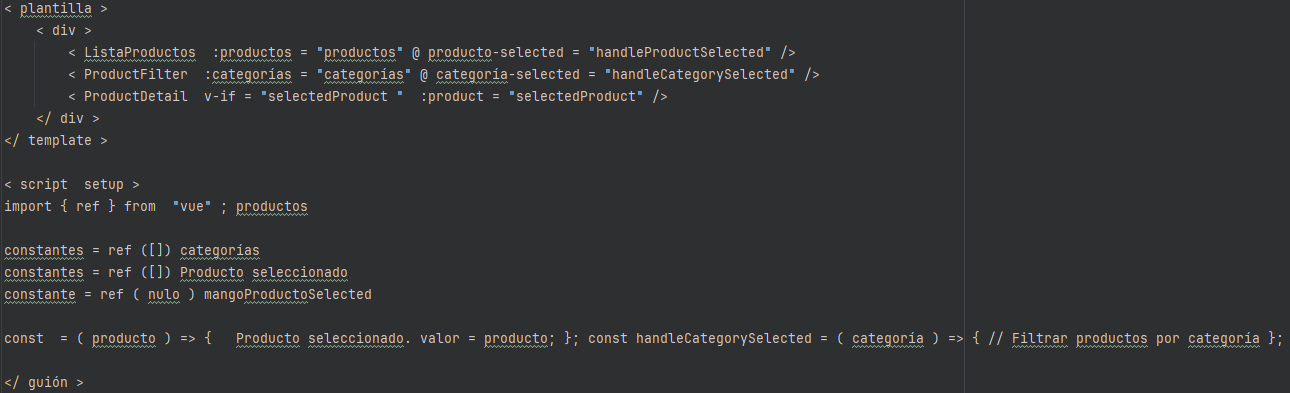
Por otro lado, la arquitectura de tres capas es otra opción ampliamente utilizada en proyectos de gran envergadura. En este modelo, la base de datos se divide en tres partes: la capa de presentación, que representa la interfaz con la que interactúa el usuario final; la capa de aplicación, encargada de procesar la lógica de negocio y servir como intermediaria entre la capa de presentación y la capa de datos; y la capa de datos, donde se almacenan los datos de la aplicación. En PHP, siguiendo esta arquitectura, la interfaz de usuario se construye con HTML y CSS, mientras que el cliente PHP maneja los datos que la aplicación envía al servidor MySQL.

1. **Ventajas y Desventajas**
   1. **Ventajas GraphQL**
      * Alta flexibilidad en la consulta de datos, permitiendo solicitar exactamente los datos que se necesitan, evitando la sobrecarga de información irrelevante y permitiendo consultas complejas en una sola solicitud.
      * Intercambio de datos eficiente, permitiendo a los clientes especificar la estructura de los datos requeridos, optimiza así la eficiencia del intercambio de datos, reduciendo la cantidad de información innecesaria transmitida sobre la red.
      * Facilita la evolución de la API sin complicaciones, haciendo más fácil introducir cambios en la API sin cambiar las aplicaciones existentes.
      * Reducción de problemas de comunicación cliente/servidor, al ser un lenguaje tipado y tener una estructura clara para las consultas y mutaciones, ayuda a minimizar los malentendidos y errores de comunicación entre el cliente y el servidor.
      * Comunicación con múltiples servicios, ya que facilita la integración de datos de múltiples fuentes y servicios.
      * Mejor manejo de versiones, eliminando la necesidad de versionar la API para cada cambio.
   2. **Desventajas GraphQL**
      * Curva de aprendizaje alta porque puede requerir tiempo y esfuerzo para que los desarrolladores se familiaricen completamente con este, lo que puede ralentizar el proceso de desarrollo al principio.
      * Mayor complejidad para los desarrolladores de servidores esto se debe a que gran parte del trabajo de procesamiento de consultas de datos se delega al servidor. Esto puede requerir que los desarrolladores de servidores implementen resolvers complejos para manejar las consultas y recuperar los datos solicitados de manera eficiente.
      * La implementación de GraphQL puede requerir estrategias de gestión de API diferentes a las utilizadas para las API de REST. Esto puede incluir consideraciones adicionales sobre precios, límites de frecuencia y políticas de seguridad, ya que GraphQL permite a los clientes solicitar datos de forma más flexible y específica.
      * Debido a la flexibilidad de las consultas en GraphQL, el almacenamiento en caché puede ser más complejo en comparación con las API de REST. Las consultas pueden variar significativamente entre clientes y pueden incluir solo los datos necesarios en cada solicitud, lo que dificulta la implementación de estrategias de caché eficientes.
   3. **Ventajas Express**
      * Curva de aprendizaje suave, especialmente para personas con conocimientos previos en JavaScrip.
      * Altamente flexible y se puede adaptar fácilmente a los requisitos específicos de cada proyecto, lo que permite un desarrollo ágil y eficiente.
      * Desarrollo gratuito y de código abierto, lo que significa que es accesible para cualquier persona y puede ser personalizada según las necesidades del proyecto.
      * Proporciona a los desarrolladores la libertad de tomar decisiones arquitectónicas según las necesidades del proyecto, lo que permite un mayor control sobre la estructura de la aplicación.
      * Simplifica el manejo de solicitudes HTTP, lo que permite a los desarrolladores crear fácilmente endpoints y gestionar la comunicación entre el cliente y el servidor.
      * Facilidad para agregar funcionalidades, separar y organizar código de manera modular.
   4. **Desventajas Express**

* Mayor responsabilidad del desarrollador en la configuración de herramientas y módulos adicionales.
* En algunos casos, Express puede carecer de ciertas funcionalidades específicas, lo que puede requerir el uso de módulos externos o la implementación de soluciones personalizadas.
* No es la mejor opción para grandes proyectos que requieran una estructura robusta o un alto nivel de abstracción
  1. **Ventajas Vue**
* Pequeño tamaño: Vue.js destaca por su reducido tamaño de descarga, aproximadamente 18 KB, lo cual es notablemente menor en comparación con otros frameworks más grandes. Esta característica no solo beneficia al rendimiento de la aplicación, sino que también influye positivamente en el SEO y la experiencia del usuario.
* Arquitectura basada en componentes: Vue.js sigue una estructura modular basada en componentes, lo que permite dividir el código en unidades más pequeñas y manejables. Todo en Vue.js se trata como un componente, lo que facilita la legibilidad y la comprensión del código al escribir en HTML, CSS y JavaScript.
* Amplia gama de herramientas: Vue.js cuenta con una sólida colección de herramientas de desarrollo frontend que están integradas de manera nativa, lo que reduce la necesidad de configuraciones complejas. Esto incluye soporte para herramientas como Babel y Webpack, así como características como pruebas unitarias, enrutamiento flexible y renderización del lado del servidor (SSR), entre otros.
* Facilidad de uso: Vue.js es reconocido por su facilidad de uso, lo que lo hace accesible incluso para principiantes en el desarrollo web. Su enfoque moderno simplifica el proceso de desarrollo y permite a los usuarios familiarizarse rápidamente con la tecnología.
  1. **Desventajas Vue**
* Complejidad en la reactividad: Aunque Vue.js ofrece una herramienta útil de binding bidireccional, su sistema de reactividad puede presentar complejidades, especialmente cuando se trata de actualizar datos de manera eficiente. Esto puede generar dificultades al manejar la reactividad de los datos y requerir soluciones adicionales para evitar errores.
* Barrera lingüística: Inicialmente, Vue.js fue mayormente adoptado por la comunidad china, lo que resultó en una concentración de recursos y discusiones en chino en foros y otros medios. Aunque Vue.js ha avanzado en la internacionalización, aún puede haber limitaciones en el soporte de idiomas, lo que puede dificultar la adopción para desarrolladores no nativos.
* Riesgos de exceso de flexibilidad: La flexibilidad de Vue.js puede llevar a la creación de código poco estructurado o "espagueti", especialmente en equipos donde cada miembro puede tener diferentes enfoques y preferencias. Esto puede complicar la mantenibilidad y escalabilidad del proyecto si no se establecen prácticas de desarrollo consistentes.
  1. **Ventajas MySQL**
* Facilidad de uso y aprendizaje: MySQL destaca por su facilidad de uso y aprendizaje, lo que lo convierte en una opción popular para desarrolladores de todos los niveles de experiencia. Su sintaxis simple y la amplia documentación disponible hacen que sea relativamente sencillo comenzar a trabajar con MySQL y desarrollar aplicaciones basadas en bases de datos.
* Compatibilidad multiplataforma: MySQL es compatible con una amplia variedad de plataformas, incluyendo Windows, Linux y macOS. Esta característica permite a los desarrolladores utilizar MySQL en diferentes entornos de desarrollo según sus necesidades y preferencias, sin preocuparse por problemas de compatibilidad.
* Escalabilidad y rendimiento: MySQL es reconocido por su capacidad para escalar y manejar grandes volúmenes de datos sin sacrificar el rendimiento. Gracias a su arquitectura optimizada y su capacidad para aprovechar múltiples hilos de ejecución, puede procesar consultas y transacciones de manera eficiente, lo que lo hace adecuado para aplicaciones de alto tráfico.
* Amplia comunidad y soporte: MySQL cuenta con una gran comunidad de usuarios y desarrolladores en todo el mundo. Esto significa que siempre hay recursos disponibles, como foros, grupos de discusión y documentación en línea, que pueden ayudar a los desarrolladores a resolver problemas y encontrar soluciones a sus desafíos de manera rápida y eficiente.
* Costo y licencia: MySQL es una base de datos de código abierto, lo que significa que es de uso gratuito y está disponible bajo la Licencia Pública General de GNU (GPL). Esta característica reduce significativamente los costos de desarrollo y licenciamiento para empresas y desarrolladores individuales, haciendo que MySQL sea una opción atractiva desde el punto de vista económico.
  1. **Desventajas MySQL**
* Limitaciones de almacenamiento y tamaño de base de datos: Aunque MySQL ofrece un almacenamiento considerable y límites de tamaño de base de datos altos, algunas empresas con grandes volúmenes de datos pueden encontrar restricciones en cuanto a escalabilidad y capacidad de almacenamiento. A pesar de ser suficientes para la mayoría de las aplicaciones, estas limitaciones pueden representar un desafío para empresas que manejan grandes cantidades de datos.
* Funcionalidades avanzadas limitadas: A pesar de su popularidad, MySQL tiene algunas limitaciones en términos de funcionalidades avanzadas. Algunas características más complejas presentes en otros sistemas de gestión de bases de datos pueden no estar disponibles o tener un soporte limitado en MySQL. Esto puede ser un obstáculo para aplicaciones que requieren funcionalidades sofisticadas.
* Replicación y alta disponibilidad: Aunque MySQL ofrece capacidades de replicación y alta disponibilidad, configurar y administrar estos sistemas puede ser complejo y requerir un conocimiento profundo de la tecnología. Esta complejidad puede ser un desafío para los desarrolladores que no están familiarizados con estos conceptos, y puede resultar en una configuración subóptima que afecta la disponibilidad del sistema.
* Optimización de consultas: La optimización de consultas en MySQL puede ser desafiante, especialmente para desarrolladores menos experimentados. Aunque MySQL ofrece herramientas y técnicas para mejorar el rendimiento de las consultas, es necesario comprender bien la estructura de la base de datos y tener conocimientos sólidos sobre el motor de almacenamiento utilizado para optimizar eficazmente las consultas.
* Transacciones y bloqueo de tablas: MySQL utiliza bloqueo de tablas para gestionar las transacciones, lo que puede provocar cuellos de botella y tiempos de respuesta lentos en entornos con una gran cantidad de transacciones concurrentes. Esta limitación puede afectar el rendimiento y la escalabilidad de las aplicaciones que dependen en gran medida de transacciones simultáneas.

1. **Principios SOLID asociados**
   1. **GraphQL**
      * **Single Responsability:** En el contexto de GraphQL, esto se observa en que cada servicio o componente debe tener una responsabilidad clara y definida dentro del sistema. Al refactorizar hacia GraphQL, los equipos pueden definir claramente las responsabilidades de cada servicio y evitar la acumulación de lógica empresarial no relacionada. Además, al dividir la funcionalidad en esquemas GraphQL separados, se promueve la cohesión y la modularidad, lo que facilita el mantenimiento y la evolución del sistema.
      * **Open/Closed:** Se expresa en la capacidad de extender la estructura inicial para agregar nuevas funcionalidades sin modificar la misma. Esta se puede extender de manera modular y agregar nuevas capacidades sin alterar el funcionamiento existente. Lo que facilita la evolución del sistema sin introducir cambios disruptivos.
      * **Liskov Substitution:** Se refiere a la capacidad de intercambiar tipos de datos sin afectar las consultas o mutaciones existentes. Gracias a la tipificación estática de GraphQL, los clientes pueden confiar en la estructura y el tipo de datos devueltos por el mismo, lo que facilita la interoperabilidad y el intercambio de tipos de datos sin introducir errores en el sistema.
      * **Interface Segregation:** Se relaciona con la capacidad de definir esquemas específicos para diferentes clientes o partes del sistema. Al utilizar esquemas separados y definir tipos específicos para cada cliente o caso de uso, se evita la dependencia de interfaces genéricas y se promueve una comunicación más específica y eficiente entre clientes y servidores.
      * **Dependency inversion:** Es la capacidad de definir un esquema de alto nivel que actúa como una abstracción para los detalles de implementación subyacentes. Al separar la definición de esquema de su implementación subyacente, se promueve una arquitectura más flexible y desacoplada, donde los módulos de alto nivel dependen de abstracciones de bajo nivel en lugar de detalles de implementación concretos.
   2. **Express**
      * **Single Responsibility:** Cada controlador o manejador de ruta debería tener una única responsabilidad. Esto se logra separando las responsabilidades en controladores distintos, lo que implica que cada uno debe cambiar sólo por una razón específica, promoviendo la modularidad y la mantenibilidad del código.
      * **Open-Closed:**: Las entidades de software deben estar abiertas para la extensión, pero cerradas para la modificación. Esto se traduce en diseñar la aplicación de manera que pueda extenderse fácilmente mediante el uso de middleware y componentes reutilizables, sin necesidad de modificar el código existente.
      * **Liskov Substitution:** Los subtipos deben ser substituibles por sus tipos base sin afectar la corrección del programa. Lo que significa que las subclases deben poder reemplazar a las clases base sin introducir errores o comportamientos inesperados.
      * **Interface Segregation:** Los clientes no deben ser forzados a depender de interfaces que no utilizan. Se deben diseñar middleware específicos y enfocados en tareas concretas en lugar de tener middlewares monolíticos que realizan múltiples tareas.
      * **Dependency Inversion:** Se logra utilizando la inyección de dependencias y los contenedores de inversión de control para gestionar las dependencias y desacoplar los componentes de la aplicación.
   3. **Vue**
      * **Single Responsibility:**

Se pueden identificar tres componentes distintos con responsabilidades específicas: ProductList, ProductFilter y ProductDetail.



El componente ProductList se encarga de mostrar una lista de productos y de emitir un evento product-selected cuando se elige un producto.

Por otro lado, el componente ProductFilter tiene la responsabilidad de mostrar una lista de categorías y de emitir un evento category-selected al seleccionar una categoría.

Finalmente, el componente ProductDetail se encarga de mostrar información detallada sobre un producto seleccionado.

Al dividir las responsabilidades en componentes separados, el código se vuelve más comprensible, fácil de mantener y de probar. Cada componente puede ser probado de manera independiente, y los cambios realizados en uno no afectarán a los otros.

* + - **Open-Closed:**

En el ejemplo con Vue.js, tenemos un componente de lista que muestra elementos y queremos añadir la capacidad de ordenarlos. Para seguir el principio OCP, creamos un componible llamado useSorting.js para manejar la lógica de ordenación. Este componible se utiliza en el componente de la aplicación que usa la lista, añadiendo la función de ordenación sin necesidad de modificar el código original del componente de lista.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Se demuestra cómo aplicar el principio Abierto/Cerrado. El componente List muestra elementos, y el componible useSorting agrega la funcionalidad de ordenación sin modificar directamente el componente List. Finalmente, el componente de la aplicación App utiliza tanto el componente List como el componible useSorting para lograr la funcionalidad deseada de ordenación de elementos.

* + - **Liskov Substitution:**

En el contexto de los componentes Vue, esto se traduce en la capacidad de extender un componente base con funcionalidades adicionales en una subclase sin afectar su comportamiento general. Por ejemplo, en el componente base Form.vue, que representa un formulario con un botón de envío, podemos crear una subclase que agregue campos de texto adicionales, como un nombre de usuario y una contraseña, sin alterar la funcionalidad principal de enviar el formulario.

Este enfoque sigue el principio de Liskov, ya que la subclase puede usarse en lugar del componente base sin afectar la capacidad de enviar el formulario o registrar mensajes en la consola cuando se envía.

Al aplicar el principio de sustitución de Liskov de esta manera en el desarrollo de interfaces de usuario con Vue, se logra una mayor flexibilidad y escalabilidad, lo que facilita el mantenimiento y la extensión de la aplicación con el tiempo. Además, este principio se encuentra implícito en la arquitectura de Vue, lo que permite aprovechar sus beneficios sin necesidad de entender los detalles técnicos subyacentes

Texto

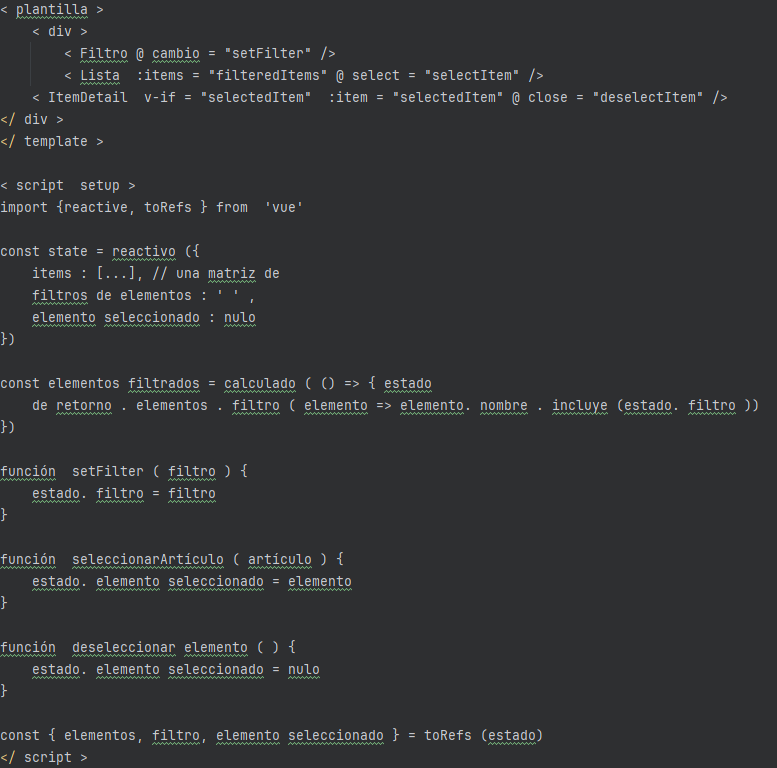
Descripción generada automáticamente

Texto

Descripción generada automáticamente

* + - **Interface Segregation:**

En Vue, este principio se refleja en el uso de componentes. Dividir estructuras complejas en componentes más pequeños con responsabilidades claras y simples permite una mejor modularidad y reutilización del código.



El ejemplo proporcionado demuestra cómo se puede aplicar el ISP en un componente Vue. En lugar de tener un componente grande que maneje el filtrado, la lista y los detalles de un elemento, se dividen estas responsabilidades en componentes más pequeños y específicos: Filtro, Lista y Detalle de Elemento. Esto facilita el mantenimiento y la modificación del código, ya que cada componente tiene una única responsabilidad y puede ser modificado independientemente de los otros.

* + - **Dependency Inversion:**

En Vue, este principio se refleja en el uso de componentes reutilizables, accesorios para la transmisión de datos entre componentes y servicios para el acceso y manipulación de datos, todos los cuales actúan como contratos que definen cómo interactúan los diferentes elementos de la aplicación.

El compromiso con la arquitectura basada en componentes de Vue y el uso de accesorios y servicios ejemplifican la aplicación del principio DIP. Al establecer contratos claros entre los distintos componentes de la aplicación, se fomenta una estructura más modular y flexible, donde los detalles específicos de implementación no comprometen la capacidad de cambio y mantenimiento del sistema en su conjunto.

* 1. **MySQL**
     + **Single Responsibility:**

Este principio establece que una clase (o en el caso de MySQL, una tabla) debería tener una única razón para cambiar.

En MySQL, se puede aplicar dividiendo las tablas en unidades cohesivas, cada una responsable de un aspecto específico de los datos. Por ejemplo, en lugar de tener una tabla de usuarios que almacena información sobre los usuarios y sus pedidos, sería mejor tener una tabla para los usuarios y otra para los pedidos, cada una con su propia responsabilidad

* + - **Open-Closed:**

Este principio establece que las entidades de software (o en el caso de MySQL, las tablas) deberían estar abiertas para su extensión pero cerradas para su modificación.

En MySQL, se puede aplicar diseñando las tablas de manera que puedan ampliarse para agregar nuevas funcionalidades sin modificar la estructura existente. Por ejemplo, si necesitas agregar un nuevo tipo de producto a una tienda en línea, deberías poder hacerlo sin tener que modificar la estructura de la tabla de productos existente.

* + - **Liskov Substitution:**

Este principio establece que los objetos de un programa deberían ser reemplazables por instancias de sus subtipos sin alterar la corrección del programa.

En el contexto de MySQL, esto se puede aplicar garantizando que las relaciones entre las tablas sean coherentes y que los datos puedan manipularse de manera consistente independientemente de la tabla específica en la que se encuentren.

* + - **Interface Segregation:**

Este principio establece que los módulos de alto nivel no deberían depender de los módulos de bajo nivel. Ambos deberían depender de abstracciones.

En el contexto de MySQL, esto puede implicar utilizar vistas, procedimientos almacenados o funciones para abstraer la lógica de acceso a datos, lo que permite que las aplicaciones interactúen con la base de datos a través de interfaces bien definidas y desacopladas de la implementación subyacente.

1. **Atributos de Calidad**
   1. **GraphQL**

**Adecuación funcional:**

* + - * Permite a los clientes solicitar exactamente los datos que necesitan, lo que garantiza que la información proporcionada sea precisa y relevante.
      * Facilita la integración de datos de múltiples fuentes y servicios, lo que mejora la interoperabilidad entre sistemas y aplicaciones.

**Fiabilidad:**

* + - * Basandose en que es un lenguaje tipado y cuenta con la capacidad para definir estructuras claras de datos, proporciona un enfoque maduro para el intercambio de datos entre clientes y servidores.
      * Maneja eficientemente grandes volúmenes de datos y solicitudes, lo que contribuye a su capacidad para tolerar fallos y mantener la disponibilidad del sistema.

**Usabilidad:**

* + - * Aunque puede tener una curva de aprendizaje alta, una vez dominado, GraphQL ofrece una forma clara y declarativa de solicitar y entregar datos, lo que mejora su comprensibilidad para los desarrolladores y usuarios finales.

**Eficiencia en el desempeño:**

* + - * Ofrece respuestas rápidas y eficientes al permitir la especificación precisa de los datos necesarios, lo que reduce el tiempo de respuesta del sistema.
      * Reduciendo la sobrecarga de información innecesaria y optimizando la transmisión de datos, se mejora el uso de recursos, como ancho de banda y capacidad de procesamiento.
  1. **Express**

**Compatibilidad**:

* + - * Cuando se habla del manejo de la carga y el rendimiento de las aplicaciones web. Utiliza técnicas como el clustering y caching para mejorar la capacidad de respuesta y la eficiencia en el procesamiento de solicitudes.

**Portabilidad**:

* + - * Ya que es altamente escalable, gracias a su arquitectura modular y su capacidad para gestionar grandes volúmenes de solicitudes HTTP de manera eficiente, Lo cual ofrece una capacidad mayor para crecer y adaptarse a las demandas cambiantes de las aplicaciones web.

**Fiabilidad**:

* + - * Express se integra fácilmente con otras tecnologías y sistemas, lo que contribuye a su capacidad para mantener la disponibilidad del sistema en entornos de producción.

**Eficiencia de desempeño**:

* + - * Al proporcionar una capa de abstracción eficiente para el manejo de solicitudes y respuestas, ayuda a optimizar la utilización de recursos del sistema, como la CPU y la memoria.
  1. **VueJS**

**Modularidad:**

* + - * Vue.js promueve la modularidad a través de su sistema de componentes, lo que permite dividir la aplicación en piezas más pequeñas y cohesivas. Esta modularidad facilita la reutilización del código, mejora la legibilidad y mantenibilidad, y permite un desarrollo más escalable y colaborativo.

**Flexibilidad:**

* + - * La flexibilidad de Vue.js permite a los desarrolladores adoptar diferentes enfoques arquitectónicos según las necesidades del proyecto. Puede ser utilizado para construir aplicaciones de una sola página (SPA), aplicaciones de múltiples páginas (MPA), aplicaciones híbridas, o incluso integrarse en aplicaciones existentes de forma gradual, lo que brinda una gran libertad arquitectónica.

**Desacoplamiento:**

* + - * Vue.js fomenta el desacoplamiento al separar claramente la capa de presentación (interfaz de usuario) de la lógica de negocio y los datos. Esta separación facilita la mantenibilidad, ya que los cambios en la interfaz de usuario no afectan la lógica subyacente y viceversa. Además, el uso de componentes independientes contribuye al desacoplamiento entre distintas partes de la aplicación

**Escalabilidad:**

* + - * La arquitectura de Vue.js es altamente escalable, lo que permite que las aplicaciones crezcan y evolucionen con facilidad. La modularidad, el desacoplamiento y la flexibilidad permiten agregar nuevas funcionalidades, modificar el código existente y escalar verticalmente o horizontalmente según sea necesario sin comprometer la estabilidad del sistema.

**Legibilidad:**

* + - * Vue.js fomenta la escritura de código limpio y legible gracias a su sintaxis intuitiva y su enfoque declarativo. Esto facilita la comprensión del código por parte de otros desarrolladores y mejora la colaboración en equipos de desarrollo.
  1. **MySQL**

**Integridad y Unicidad:**

Unicidad: Elimina duplicados y asegura que los datos sean únicos. Esto se logra mediante la definición de claves primarias y restricciones de unicidad.

Integridad: Evita la corrupción de datos. [Asegurarse de que los datos crucen correctamente entre diferentes fuentes](https://ignaciogavilan.com/los-ocho-atributos-de-la-calidad-de/).

**Diseño de la Base de Datos:**

Modelo Relacional: Dedica tiempo al diseño de tu base de datos. Utiliza un modelo relacional para definir tablas, campos y relaciones. Esto facilita el mantenimiento y garantiza un rendimiento adecuado.

Índices: Crea índices en los campos relevantes para optimizar las consultas. Utiliza índices en campos utilizados en cláusulas WHERE o JOIN. Los índices sobre campos con valores únicos son especialmente efectivos.

**Optimización de Consultas:**

Consulta Eficiente: Diseña consultas que utilicen índices y filtros adecuados. Evita operaciones costosas como JOINs innecesarios.

Uso de Índices: Crea índices en campos utilizados en búsquedas. Los índices cortos y sobre campos con valores únicos son preferibles

**Scripts y Programación:**

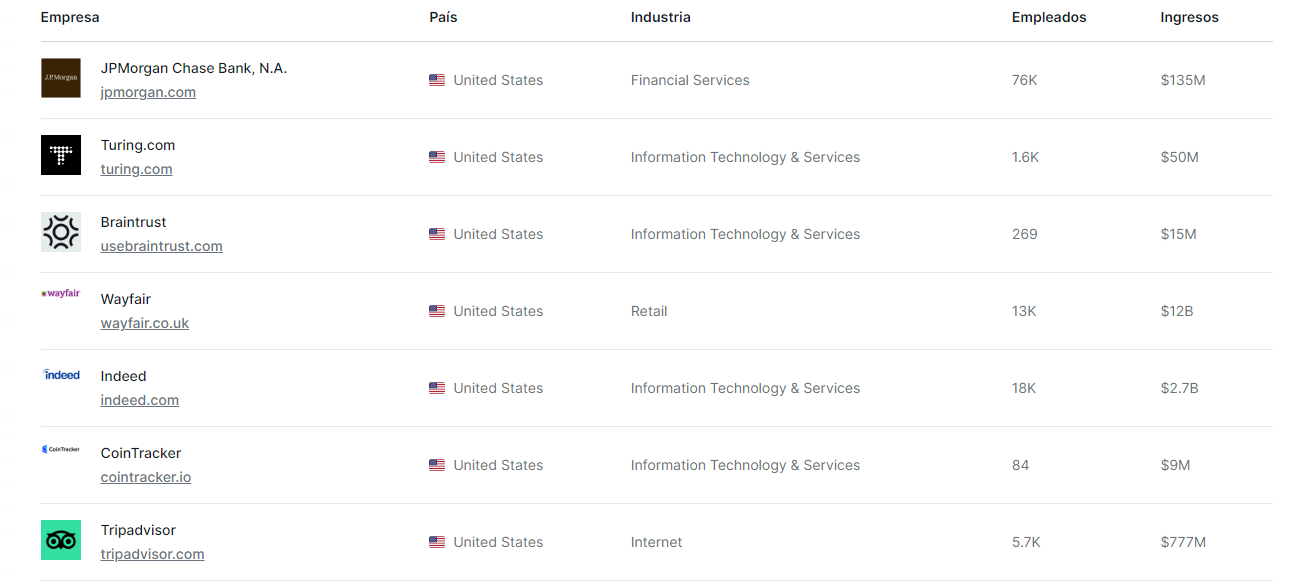
Optimización de Scripts: Mejora los scripts (por ejemplo, PHP, ASP) que muestran los resultados de las consultas. Asegúrate de que el código sea eficiente y siga buenas prácticas.

Seguridad: Considera la seguridad al diseñar tus scripts. Evita vulnerabilidades como inyección SQL

1. **Casos de estudio relevantes** 
   1. **GraphQL**

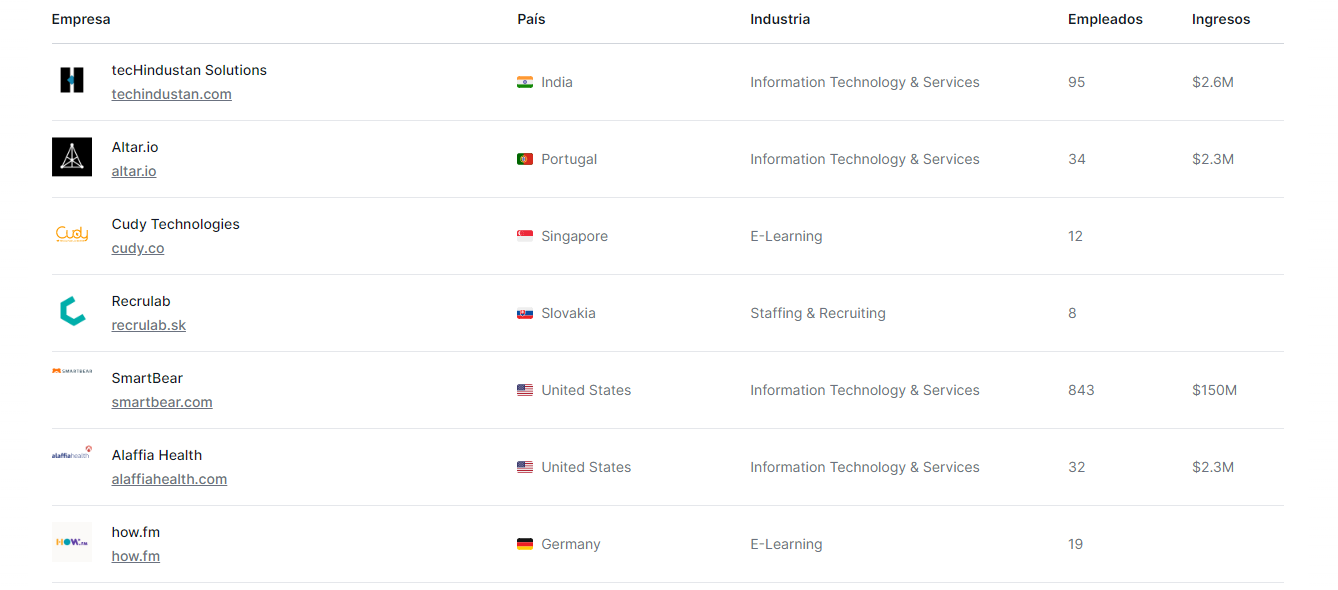
* GitHub: Mejora la eficiencia al permitir solicitudes precisas y flexibles, reduciendo la necesidad de múltiples llamadas API y simplificando la integración y automatización de flujos de trabajo.
* Facebook: Para aplicaciones móviles, GraphQL ayuda a optimizar el ancho de banda y a gestionar las conexiones lentas al agrupar múltiples solicitudes en una, reduciendo así la carga del servidor.
* Airbnb: La adopción de GraphQL ha permitido una migración gradual de servicios, eliminando código repetitivo y mejorando la velocidad de carga de las páginas, lo que mejora la experiencia del usuario.
* Netflix: GraphQL se ha utilizado para crear un backend completamente formado a partir de microservicios, lo que ha permitido un desarrollo más rápido y eficiente de aplicaciones CRUD.
* PayPal: Utiliza GraphQL para mejorar la agilidad del desarrollo y simplificar la experiencia del desarrollador, proporcionando un único punto de acceso independiente del lenguaje para clientes externos.
* Coursera: Ha adoptado GraphQL después de experimentar con otras soluciones, utilizando un servicio ensamblador para facilitar la comunicación entre microservicios y simplificar la búsqueda de datos en toda la plataforma.

A continuación, se pueden observar más compañias que hacen uso de GraphQL:



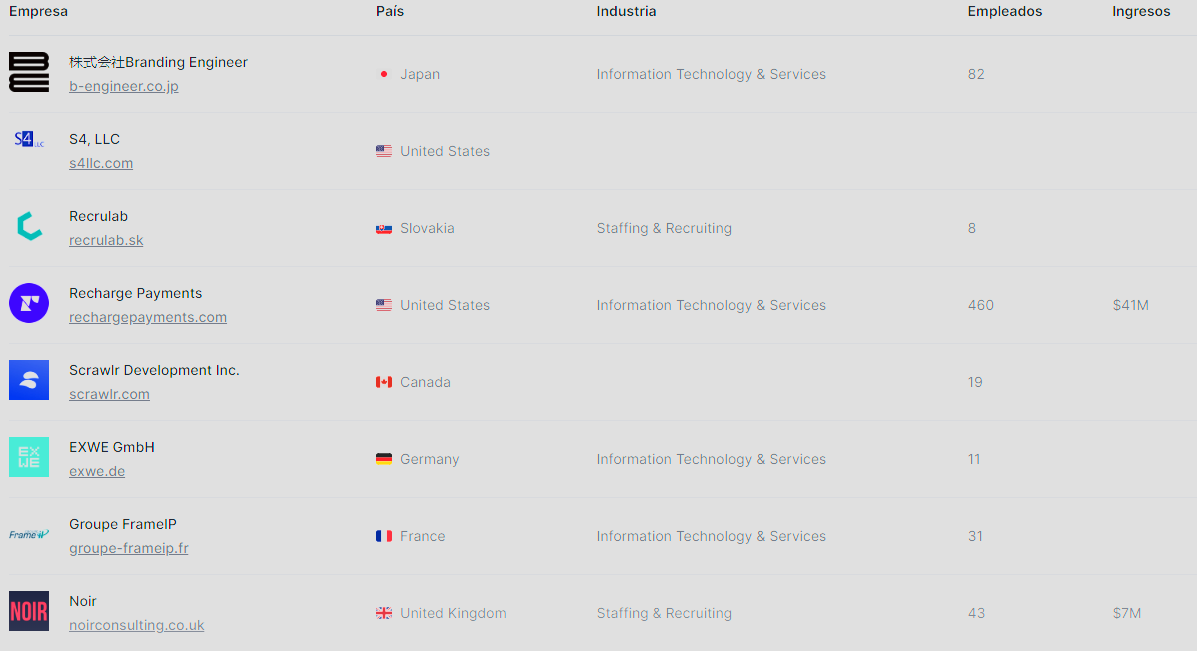
* 1. **Express**
     + FoxSports: Ya que se destaca por su eficiencia en el manejo de picos de tráfico durante eventos deportivos cruciales. La capacidad de Express.js para mantener un rendimiento óptimo asegura una experiencia de usuario fluida y sin interrupciones mientras los espectadores acceden al contenido deportivo en línea.
     + X: Para gestionar las solicitudes del servidor y manejar la lógica de enrutamiento necesaria para mostrar tweets, perfiles de usuario y otras funcionalidades de la plataforma en tiempo real.
     + IBM: Dada la capacidad para facilitar un desarrollo ágil y adaptable en proyectos empresariales, la flexibilidad y modularidad de Express.js permiten a IBM ajustarse rápidamente a los requisitos cambiantes del proyecto.
     + Uber: Debido a la capacidad para manejar eficientemente grandes volúmenes de solicitudes de usuarios en tiempo real, asegurando una experiencia fluida y rápida para sus usuarios al solicitar y gestionar viajes.
     + QuizUp: Para una plataforma de juegos en línea que requiere una comunicación rápida y fluida entre jugadores y servidor, Express.js ayuda debido a su eficiencia, lo que permite un manejo ágil de las interacciones en tiempo real y una respuesta rápida a las solicitudes de los usuarios.

A continuación, se pueden observar más compañías que hacen uso de Express:



* 1. **Vue**

Lista de empresas usando Vue.js





**Paises que Usan vuejs**

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**Empresas mas populares que hacen uso de VueJs**

Adobe: Adobe, conocida por sus productos de software creativo como Photoshop y Illustrator, utiliza Vue.js en su plataforma Adobe Portfolio. Adobe Portfolio permite a los usuarios crear y personalizar sus propios sitios web de portafolio para mostrar su trabajo creativo. Vue.js se utiliza para crear interfaces de usuario interactivas y receptivas que facilitan la creación y personalización de sitios web.

GitLab: GitLab, una plataforma de desarrollo de software que proporciona herramientas para el control de versiones, seguimiento de problemas, integración continua y más, utiliza Vue.js en su aplicación web. Vue.js se utiliza para crear interfaces de usuario dinámicas y receptivas que permiten a los usuarios gestionar sus repositorios de código, colaborar en proyectos y realizar un seguimiento del progreso del desarrollo de software de manera eficiente.

Nintendo: Nintendo, una de las principales empresas de videojuegos del mundo, utiliza Vue.js en su aplicación web My Nintendo. My Nintendo es una plataforma que permite a los usuarios acceder a recompensas, ofertas especiales y contenido exclusivo relacionado con los productos de Nintendo. Vue.js se utiliza para crear una experiencia de usuario interactiva y fluida que permite a los usuarios navegar por el contenido de manera intuitiva y realizar acciones como canjear puntos de recompensa y participar en promociones.

Grammarly: Grammarly, una herramienta de corrección de gramática y ortografía ampliamente utilizada, utiliza Vue.js en su aplicación web y extensiones de navegador. Vue.js se utiliza para crear interfaces de usuario dinámicas y receptivas que permiten a los usuarios corregir errores gramaticales y ortográficos en tiempo real mientras escriben en línea. La integración de Vue.js ayuda a proporcionar una experiencia de usuario fluida y sin problemas en la aplicación y las extensiones de Grammarly.

* 1. **MySQL**
     + - Análisis y Diseño de Bases de Datos:

En un estudio académico, se exploró el análisis y diseño de bases de datos utilizando MySQL. Se utilizaron herramientas como MySQL Workbench para crear bases de datos de manera práctica. [El objetivo era que los estudiantes adquirieran habilidades aplicables en el ámbito laboral](http://www.ri.uagro.mx/bitstream/handle/uagro/3533/MCON_13701_21.pdf?sequence=1).

* + - * Casos de Uso de MySQL Database:

Oracle proporciona ejemplos de casos de uso de MySQL, como la gestión de tablas de rutas de VCN y listas de seguridad. [Estos ejemplos muestran cómo MySQL se aplica en entornos de red y seguridad](https://docs.oracle.com/es-ww/iaas/database-tools/doc/mysql-database-use-cases.html).

* + - * Desarrollo de Software y Bases de Datos:

En un caso práctico, se diseñó una base de datos llamada ALUMNOS utilizando MySQL. Se aplicaron conceptos de análisis, diseño y creación de bases de datos. [Este enfoque práctico ayuda a los estudiantes a adquirir experiencia para futuros proyectos](https://www.studocu.com/es-mx/document/instituto-politecnico-nacional/construccion-de-base-de-datos/caso-practico-de-my-sql-alumnos/6931360).

* + - * Uber: Uber, la popular plataforma de transporte compartido, utiliza MySQL para gestionar una amplia gama de datos, incluidos perfiles de usuarios, viajes, pagos y más. MySQL ha sido crucial para permitir a Uber escalar y manejar grandes volúmenes de datos mientras mantiene un rendimiento óptimo en su plataforma.
      * Airbnb: Airbnb, el servicio de alojamiento y alquiler de propiedades a corto plazo, ha utilizado MySQL en su infraestructura desde sus primeros días. MySQL se utiliza para gestionar datos de listados de propiedades, reservas, perfiles de usuarios y más. La escalabilidad y la confiabilidad de MySQL han sido fundamentales para permitir a Airbnb crecer hasta convertirse en una empresa global líder en su industria.
      * LinkedIn: LinkedIn, la red social profesional más grande del mundo, también ha utilizado MySQL en su infraestructura desde sus inicios. MySQL se utiliza para gestionar datos de perfiles de usuarios, conexiones, mensajes y más. La escalabilidad y la capacidad de MySQL para manejar grandes volúmenes de datos han sido fundamentales para permitir a LinkedIn crecer y escalar su plataforma a millones de usuarios en todo el mundo.
      * Shopify: Shopify, una plataforma de comercio electrónico líder, utiliza MySQL para gestionar una amplia gama de datos relacionados con productos, pedidos, clientes y más. MySQL ha sido crucial para permitir a Shopify ofrecer una plataforma escalable y confiable para miles de comerciantes en línea en todo el mundo.
      * GitHub: GitHub, la plataforma de desarrollo de software más grande del mundo, utiliza MySQL para gestionar datos críticos, como repositorios de código, solicitudes de extracción, problemas y más. La escalabilidad y la confiabilidad de MySQL han sido fundamentales para permitir a GitHub manejar millones de repositorios y millones de desarrolladores en su plataforma.

**Referencias bibliograficas**

* [1] Rozas, J. M. G. (2023, 8 noviembre). GraphQL: ¡todos para uno y uno para todos! 1/2. *Paradigma Digital*. <https://www.paradigmadigital.com/dev/graphql-todos-uno-uno-todos/>
* Acosta, J. (2023, 12 abril). GraphQL: Qué es y qué ventajas ofrece. *OpenWebinars.net*. <https://openwebinars.net/blog/graphql-que-es-y-que-ventajas-ofrece/>
* Apptec - Fabricamos el software que necesitas. (2020, 11 mayo). *Apptec: GraphQL: Historia, origen y funcionamiento*. Apptec. <https://apptec.cl/blog/graphql-historia-origen-y-funcionamiento>
* Charboneau, T. (2024, 2 enero). *6 Examples of GraphQL in Production at Large Companies | Nordic APIs |*. Nordic APIs. <https://nordicapis.com/6-examples-of-graphql-in-production-at-large-companies/>
* Eseme, S. (2023, 29 mayo). *GraphQL vs REST: Todo lo que Necesitas Saber*. Kinsta®. <https://kinsta.com/es/blog/graphql-vs-rest/#ventajas-de-graphql>
* *GraphQL y API de REST: diferencia entre las arquitecturas de diseño de API. AWS*. (s. f.). Amazon Web Services, Inc. <https://aws.amazon.com/es/compare/the-difference-between-graphql-and-rest/#:~:text=GraphQL%20es%20ideal%20para%20or%C3%ADgenes%20de%20datos%20grandes%2C%20complejos%20e%20interrelacionados.&text=REST%20tiene%20varios%20puntos%20de,URL%20para%20definir%20los%20recursos>.
* Hamilton, D. (2024, 12 enero). *¿Qué es la prueba GraphQL? - Parasoft*. Parasoft. <https://es.parasoft.com/blog/what-is-graphql-testing/#:~:text=GraphQL%20es%20un%20lenguaje%20de,ineficiente%2C%20especialmente%20en%20redes%20m%C3%B3viles>.
* Lee, A. (2022, 29 junio). Rediscovering SOLID Principles with GraphQL - Prodigy Engineering - Medium. *Medium*. <https://medium.com/prodigy-engineering/rediscovering-solid-principles-with-graphql-c24070221263>
* *¿Qué es GraphQL?* (s. f.). <https://www.redhat.com/es/topics/api/what-is-graphql>
* *An Intro to ExpressJS | Flexiple*. (s. f.). Flexiple. <https://flexiple.com/express-js/deep-dive>
* Arvindpdmn, D. (2022, 15 febrero). *Express.js*. Devopedia. <https://devopedia.org/express-js>
* *Companies using Express*. (s. f.). <https://expressjs.com/en/resources/companies-using-express.html>
* Gil, M. B. (2023, 13 diciembre). *Análisis de las Ventajas y Desventajas de Express JS: Descubre lo bueno y lo malo*. Ventajas y Desventajas Top. <https://ventajasydesventajastop.com/express-js-ventajas-y-desventajas/>
* Jesús. (2023, 12 noviembre). *Introducción a Express.js: Explorando sus Funciones y Beneficios*. Tutoriales Dongee. <https://www.dongee.com/tutoriales/que-es-y-para-que-sirve-express-js/>
* Kinsta. (2022, 19 diciembre). *¿Qué es Express.js? Todo lo que Debes Saber*. Kinsta®. <https://kinsta.com/es/base-de-conocimiento/que-es-express/>
* Leroux, B. (2023, 25 agosto). *Express.js: What it is used for and when, where to use it for your enterprise app development*. Startechup Inc. <https://www.startechup.com/es/blog/express-js-what-it-is-used-for-and-when-where-to-use-it-for-your-enterprise-app-development/>
* Parker, R. N. (2023, 30 septiembre). The Notable History of Express.js - Ryan Neil Parker - Medium. *Medium*. <https://medium.com/@ryanneilparker/the-notable-history-of-express-js-6869ba4fcd25>
* colaboradores de Wikipedia. (2024, 2 febrero). *Vue.js*. Wikipedia, la Enciclopedia Libre. <https://es.wikipedia.org/wiki/Vue.js>
* Palomeque-Zambrano, E. E., & Campoverde-Molina, M. (2023). Propuesta de una arquitectura de software para el desarrollo de sitios web accesibles. MQRInvestigar, 7(3), 1458-1474. <https://doi.org/10.56048/mqr20225.7.3.2023.1458-1474>
* Eseme, S. (2023, 23 agosto). *10 cosas que debes saber sobre Vue.js Frontend Framework*. Kinsta®. <https://kinsta.com/es/blog/vue-js/>
* Gerchev, I. (2018, 19 diciembre). *Patrones de diseño para la comunicación entre componentes Vue.js*. Code Envato Tuts+. <https://code.tutsplus.com/es/design-patterns-for-communication-between-vuejs-component--cms-32354t>
* *Estructura de carpetas de VueJS - Javascript en español*. (s. f.). Lenguaje JS. <https://lenguajejs.com/vuejs/introduccion/estructura-carpetas/>
* Chacaltana, G., & Chacaltana, G. (2021, 11 enero). Una breve cronología de MySQL - $> SoloCodigoWeb. *%%STRING%%#62; SoloCodigoWeb - Bitácora Digital*. <https://solocodigoweb.com/blog/2014/03/04/una-breve-cronologia-de-mysql/>
* BlogAdmin, & BlogAdmin. (2023, 2 julio). MySQL ventajas y desventajas: un análisis completo. *Informática y Tecnología Digital*. <https://informatecdigital.com/bases-de-datos/mysql-ventajas-y-desventajas-un-analisis-completo/>
* colaboradores de Wikipedia. (2024b, marzo 1). *MySQL*. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/MySQL
* <https://www.studocu.com/pe/document/universidad-nacional-de-san-martin-peru/administracion-de-base-de-datos/evolucion-historica-de-los-gestores-de-base-de-datos-mysql/17576507>
* Tokio. (2024, 26 febrero). *¿Qué es MySQL? ¿Cómo surgió? ¿Cuáles son sus características?* Tokio School. <https://www.tokioschool.com/noticias/que-es-mysql/>
* *TheirStack App*. (s. f.). <https://app.theirstack.com/search/tech/new?query=JTdCJTIybmFtZSUyMiUzQSUyMkNvbXBhbmllcyUyMHVzaW5nJTIwVnVlLmpzJTIyJTJDJTIyYXV0b19zZWFyY2glMjIlM0F0cnVlJTJDJTIycXVlcnklMjIlM0ElN0IlMjJjb21wYW55X3RlY2hub2xvZ3lfc2x1Z19vciUyMiUzQSU1QiUyMnZ1ZS1qcyUyMiU1RCUyQyUyMm9yZGVyX2J5JTIyJTNBJTVCJTdCJTIyZmllbGQlMjIlM0ElMjJ0aGVpcnN0YWNrX3Njb3JlJTIyJTJDJTIyZGVzYyUyMiUzQXRydWUlN0QlNUQlN0QlN0Q=>
* *Arquitecturas de bases de datos MySQL en PHP para proyectos grandes*. (s. f.). desarrolladorphp.com. <https://desarrolladorphp.com/blog/arquitecturas-de-bases-de-datos-mysql-en-php-para-proyectos-grandes>