Backend

roadmap.sh

Internet

Internet es una red mundial de ordenadores conectados entre sí que se comunican mediante un conjunto normalizado de protocolos.

Visite los siguientes recursos para saber más:

- ¿Cómo funciona Internet?
- Explicación de Internet
- ¿Cómo funciona Internet?
- Introducción a Internet
- ¿Cómo funciona Internet?
- Cómo funciona Internet en 5 minutos
- Red informática | Certificado de asistencia informática de Google

¿Cómo funciona Internet?

Internet es una red mundial de ordenadores conectados entre sí que se comunican mediante un conjunto normalizado de protocolos.

Visite los siguientes recursos para saber más:

- ¿Cómo funciona Internet?
- Explicación de Internet
- ¿Cómo funciona Internet?
- Introducción a Internet
- ¿Cómo funciona Internet?
- Cómo funciona Internet en 5 minutos
- ¿Cómo funciona Internet? (Curso completo)

¿Qué es HTTP?

HTTP es el protocolo de comunicación de capa de aplicación basado en `TCP/IP` que estandariza la forma en que el cliente y el servidor se comunican entre sí. Define cómo se solicita y transmite el contenido a través de Internet.

Visite los siguientes recursos para obtener más información:

- Todo lo que necesita saber sobre HTTP
- ¿Qué es HTTP?
- Curso completo de redes HTTP
- Visión general de HTTP
- HTTP/3 de la A a la Z: Conceptos básicos
- De HTTP/1 a HTTP/2 y HTTP/3
- Curso intensivo y exploración de HTTP
- Explicación de SSL, TLS y HTTPS

¿Navegadores y su funcionamiento?

Un navegador web es una aplicación de software que permite al usuario acceder y visualizar páginas web u otros contenidos en línea a través de su interfaz gráfica de usuario.

Visite los siguientes recursos para obtener más información:

- Cómo funcionan los navegadores
- Función del motor de renderizado en los navegadores
- Rellenando la Página: Cómo funcionan los navegadores

¿Cómo funciona el DNS?

El Sistema de Nombres de Dominio (DNS) es la guía telefónica de Internet. Los humanos acceden a la información en línea a través de nombres de dominio, como nytimes.com o espn.com. Los navegadores web interactúan a través de direcciones IP (Protocolo de Internet). El DNS traduce los nombres de dominio a direcciones IP para que los navegadores puedan cargar los recursos de Internet.

Visite los siguientes recursos para obtener más información:

- ¿Qué es el DNS?
- Cómo funciona el DNS (cómic)
- Entender los nombres de dominio
- DNS y su funcionamiento
- Registros DNS
- Miniserie completa sobre DNS

¿Qué es un nombre de dominio?

Un nombre de dominio es una dirección única y fácil de recordar que se utiliza para acceder a sitios web, como "google.com" y "facebook.com". Los usuarios pueden conectarse a sitios web utilizando nombres de dominio gracias al sistema DNS.

Visite los siguientes recursos para obtener más información:

- ¿Qué es un nombre de dominio?
- ¿Qué es un nombre de dominio? | Nombre de dominio frente a URL
- Guía para principiantes sobre el funcionamiento de los nombres de dominio

¿Qué es el alojamiento web?

El alojamiento web es un servicio en línea que le permite publicar los archivos de su sitio web en Internet. Así, cualquiera que tenga acceso a Internet podrá acceder a su sitio web.

Visite los siguientes recursos para obtener más información:

- ¿Qué es el alojamiento web? Explicación
- Explicación de los distintos tipos de alojamiento web
- Dónde alojar un proyecto Fullstack con poco presupuesto

- ¿Cuál es la diferencia entre página web, sitio web, servidor web y motor de búsqueda?
- ¿Qué es un servidor web?

Aprender un lenguaje de programación

Incluso si eres un principiante lo menos que deberías saber es que el Desarrollo Web se clasifica principalmente en dos facetas: Desarrollo Frontend y Desarrollo Backend. Y obviamente, ambos tienen su respectivo conjunto de herramientas y tecnologías. Por ejemplo, cuando hablamos de Desarrollo Frontend, siempre vienen 3 nombres en primer lugar - HTML, CSS y JavaScript.

Del mismo modo, cuando se trata de desarrollo web backend, necesitamos principalmente un lenguaje de programación backend (o del lado del servidor) para que el sitio web funcione junto con otras herramientas y tecnologías como bases de datos, frameworks, servidores web, etc.

Elija un lenguaje de la lista y asegúrese de aprender sus peculiaridades, los detalles básicos sobre su tiempo de ejecución, por ejemplo, la concurrencia, el modelo de memoria, etc.

Rust

Rust es un moderno lenguaje de programación de sistemas centrado en la seguridad, la velocidad y la concurrencia. Consigue estos objetivos siendo seguro en memoria sin utilizar la recolección de basura.

Visite los siguientes recursos para obtener más información:

- The Rust Programming Language - libro en línea
- Rust by Example - colección de ejemplos ejecutables
- Rust vs. Go: Por qué son mejores juntos
- Rust en cifras: El lenguaje de programación Rust en 2021

Java

Java es un lenguaje de propósito general, utilizado principalmente para aplicaciones basadas en Internet. Fue creado en 1995 por James Gosling en Sun Microsystems y es una de las opciones más populares para los desarrolladores de backend.

Visite los siguientes recursos para obtener más información:

- Visite la hoja de ruta dedicada a Java
- Sitio web de Java
- Codeacademy - Curso gratuito
- Tutoriales de W3 Schools
- Curso intensivo de Java
- Curso completo de Java

C#

C# (pronunciado "C sharp") es un lenguaje de programación de propósito general creado por Microsoft. Sirve para realizar diferentes tareas y puede utilizarse para crear aplicaciones web, juegos, aplicaciones para móviles, etc.

Visita los siguientes recursos para aprender más:

- Ruta de aprendizaje de C#
- C# en las escuelas W3
- Introducción a C#
- Tutoriales de C#

PHP

PHP es un lenguaje de programación de propósito general que suele utilizarse para crear páginas web dinámicas e interactivas. Fue creado originalmente por el programador danés-canadiense Rasmus Lerdorf en 1994. En la actualidad, la implementación de referencia de PHP está producida por The PHP Group y respaldada por PHP Foundation. PHP admite estilos de programación procedimentales y orientados a objetos, así como algunos elementos de programación funcional.

Visite los siguientes recursos para aprender más:

- Sitio web de PHP
- Aprenda PHP - W3Schools
- PHP - El camino correcto
- PHP para Principiantes
- PHP Para Principiantes Absolutos
- Tutorial Completo de PHP 8 - Aprenda PHP de la Manera Correcta en 2022

Go

Go es un lenguaje de programación de código abierto apoyado por Google. Go se puede utilizar para escribir servicios en la nube, herramientas CLI, se utiliza para el desarrollo de API, y mucho más.

Visite los siguientes recursos para obtener más información:

- Visita Dedicated Go Roadmap
- Un recorrido por Go - Conceptos básicos de Go
- Documentación de referencia de Go
- Go by Example - programas de ejemplo comentados
- Aprende Go | Codecademy
- Tutorial Go de W3Schools
- Creación de una API RESTful JSON en Go
- Go, el lenguaje de programación de la nube

JavaScript

Aparte de ser utilizado en el navegador, JavaScript también se utiliza en el backend, por ejemplo, utilizando Node.js o Deno para escribir código del lado del servidor en JavaScript.

Si eliges JavaScript para el Backend, mi recomendación personal sería aprender JavaScript y luego ir con Node.js ya que es la opción más popular y ampliamente utilizada. Además, recomendaría aprender TypeScript más adelante a medida que continúe con su viaje de desarrollo de backend; es un superconjunto de JavaScript y se utiliza en muchos proyectos.

Visita los siguientes recursos para aprender más:

- Visita Dedicated JavaScript Roadmap
- W3Schools - Tutorial de JavaScript
- Tutorial de JavaScript moderno
- Eloquent Javascript - Libro
- You Dont Know JS Yet (serie de libros)
- Codecademy - Aprende JavaScript
- Curso acelerado de JavaScript para principiantes
- Curso acelerado de Node.js
- Tutorial de Node.js para principiantes

Python

Python es un conocido lenguaje de programación fuertemente tipado y dinámicamente tipado. Al ser un lenguaje interpretado, el código se ejecuta tan pronto como se escribe y la sintaxis de Python permite escribir código de forma programática funcional, procedimental u orientada a objetos.

Visite los siguientes recursos para obtener más información:

- Visite la hoja de ruta dedicada a Python
- Sitio web de Python
- Introducción a Python
- Automatice las cosas aburridas
- FreeCodeCamp.org - ¿Cómo aprender Python?
- Principios de Python - Conceptos básicos de Python
- W3Schools - Tutorial de Python
- Curso acelerado de Python
- Codecademy - Aprende Python 2
- Introducción a Python para no programadores
- Introducción a Python e InfluxDB
- Python para principiantes - Aprende Python en 1 hora

Ruby

Ruby es un lenguaje de programación interpretado de alto nivel que mezcla Perl, Smalltalk, Eiffel, Ada y Lisp. Ruby se centra en la simplicidad y la productividad, junto con una sintaxis que se lee y escribe de forma natural. Ruby admite programación procedimental, funcional y orientada a objetos, y está tipado dinámicamente.

Visite los siguientes recursos para obtener más información:

- Sitio web de Ruby
- Aprende Ruby en 20 minutos
- Aprende Ruby | Codecademy
- Ruby, una introducción al mejor amigo del programador

Sistemas de control de versiones

Los sistemas de control de versiones y fuentes permiten a los desarrolladores seguir y controlar los cambios en el código a lo largo del tiempo. Estos servicios suelen incluir la posibilidad de realizar revisiones atómicas del código, bifurcarse a partir de puntos concretos y comparar versiones del código. Son útiles para determinar quién, qué, cuándo y por qué se hicieron cambios en el código.

Visita los siguientes recursos para obtener más información:

- Git
- ¿Qué es el control de versiones?

Git

Git es un sistema de control de versiones distribuido, gratuito y de código abierto, diseñado para gestionar con rapidez y eficacia desde proyectos pequeños a muy grandes.

Visita los siguientes recursos para obtener más información:

- Curso acelerado de Git y GitHub para principiantes
- Aprende Git con tutoriales, noticias y consejos - Atlassian
- Git Cheatsheet
- Aprende a ramificar en Git
- Tutorial de Git

Servicios de alojamiento de repositorios

Cuando se trabaja en equipo, a menudo se necesita un lugar remoto donde colocar el código para que otros puedan acceder a él, crear sus propias ramas y crear o revisar pull requests. Estos servicios suelen incluir funciones de seguimiento de incidencias, revisión de código e integración continua. Algunas opciones populares son GitHub, GitLab, BitBucket y AWS CodeCommit.

Visita los siguientes recursos para obtener más información:

- GitHub
- GitLab
- BitBucket
- Cómo elegir el mejor repositorio de código fuente

GitHub

GitHub es un proveedor de alojamiento en Internet para el desarrollo de software y el control de versiones mediante Git. Ofrece las funcionalidades de control de versiones distribuidas y gestión de código fuente de Git, además de sus propias características.

Visita los siguientes recursos para obtener más información:

- Sitio web de GitHub
- Documentación de GitHub
- Cómo utilizar Git en un equipo de desarrollo profesional
- ¿Qué es GitHub?
- Git frente a GitHub: ¿Cuál es la diferencia?
- Git y GitHub para principiantes
- Git y GitHub - CS50 más allá de 2019
- Aprende a hacer ramas en Git

GitLab

GitLab es un proveedor de alojamiento en Internet para el desarrollo de software y el control de versiones mediante Git. Ofrece las funcionalidades de control de versiones distribuidas y gestión de código fuente de Git, además de sus propias características.

Visita los siguientes recursos para obtener más información:

- Sitio web de GitLab
- Documentación de GitLab

Bitbucket

Bitbucket es un servicio de alojamiento y repositorio de código fuente basado en Git que es la alternativa de Atlassian a otros productos como GitHub, GitLab, etc.

Bitbucket ofrece opciones de alojamiento a través de Bitbucket Cloud (servidores de Atlassian), Bitbucket Server (local del cliente) o Bitbucket Data Centre (número de servidores en el entorno local o en la nube del cliente).

Visita los siguientes recursos para obtener más información:

- Sitio web de Bitbucket
- Primeros pasos con Bitbucket
- Usando Git con Bitbucket Cloud
- Una breve visión general de Bitbucket
- Tutorial de Bitbucket | Cómo usar Bitbucket Cloud
- Tutorial de Bitbucket | Bitbucket para principiantes

Sistema operativo y conocimientos generales

El sistema operativo es un programa que gestiona los recursos de un ordenador, especialmente la asignación de esos recursos entre otros programas. Los recursos típicos incluyen la unidad central de procesamiento (CPU), la memoria del ordenador, el almacenamiento de archivos, los dispositivos de entrada/salida (E/S) y las conexiones de red.

Visite los siguientes recursos para obtener más información:

- ¿Qué es un sistema operativo?
- Resumen del sistema operativo
- Sistemas operativos: Curso acelerado de informática nº 18
- Introducción al sistema operativo

Uso de terminales

Los terminales, también conocidos como líneas de comandos o consolas, nos permiten realizar y automatizar tareas en un ordenador sin necesidad de utilizar una interfaz gráfica de usuario.

Visite los siguientes recursos para obtener más información:

- Curso intensivo de línea de comandos
- Uso básico de la terminal - Cheatsheet para hacer la línea de comandos FÁCIL
- 50+ Comandos Linux que debes conocer

Conceptos básicos de POSIX

POSIX (Portable Operating System Interface) es una familia de estándares para mantener la compatibilidad entre sistemas operativos. Describe las utilidades, API y servicios que un sistema operativo compatible debe proporcionar al software, facilitando así la migración de programas de un sistema a otro.

Un ejemplo práctico: en un sistema operativo tipo Unix, existen tres *flujos estándar*, `stdin`, `stdout` y `stderr` - son conexiones de E/S con las que probablemente te encuentres al utilizar un terminal, ya que gestionan el flujo de la **entrada estándar** (stdin), la **salida estándar** (stdout) y el **error estándar** (stderr).

Así que, en este caso, cuando queremos interactuar con cualquiera de estos flujos (a través de un proceso, por ejemplo), la API del sistema operativo POSIX lo hace más fácil - por ejemplo, en la cabecera de C `<unistd.h>` donde stdin, stderr y stdout se definen como `STDIN_FILENO`, `STDERR_FILENO` y `STDOUT_FILENO`.

POSIX también añade un estándar para los códigos de salida, la semántica del sistema de ficheros y otras convenciones de la API de utilidades de línea de comandos.

Visite los siguientes recursos para obtener más información:

- Estándar POSIX por el IEEE
- Resumen de algunas implementaciones de POSIX

- Guía de POSIX

Comandos básicos de terminal

Trabajar en la terminal es una práctica común para cualquier desarrollador de Backend y existen muchos comandos y utilidades que pueden ayudarte a realizar tus tareas de forma más eficiente.

La mejor manera de aprender estos comandos es practicarlos en tu propia máquina/entorno. Específicamente, estos están relacionados con los comandos/utilidades de Linux que son los más prevalentes en el mercado.

Para entender estos comandos, lea las páginas del manual utilizando el comando `man`, por ejemplo, `man grep`, `man awk`, etc.

Después de una exposición y práctica suficientes a estos comandos, será más fácil utilizarlos en la práctica.

Visita los siguientes recursos para aprender más:

- 40 comandos básicos de Linux
- Una colección de alternativas modernas/rápidas/sanas a comandos comunes de UNIX
- Tutorial de línea de comandos
- Desafío de la línea de comandos
- Los 50 comandos más populares de Linux y Terminal (con marcas de tiempo)

Conocimientos de sistemas operativos

Funcionamiento general de los sistemas operativos

Un sistema operativo es un programa principal en el ordenador, que gobierna todas las demás aplicaciones. Le permite utilizar navegadores, jugar, imprimir documentos y ejecutar su programa favorito.

Visite los siguientes recursos para obtener más información:

- Sistema operativo - Visión general
- Conceptos del sistema operativo
- Conceptos básicos del sistema operativo

Gestión de la memoria

El término Memoria puede definirse como una colección de datos en un formato específico. Se utiliza para almacenar instrucciones y procesar datos. La memoria comprende una gran matriz o grupo de palabras o bytes, cada uno con su propia ubicación. El principal motivo de un sistema informático es ejecutar programas. Estos programas, junto con la información a la que acceden, deben estar en la memoria principal durante la ejecución. La CPU obtiene instrucciones de la memoria según el valor del contador de programa.

Para lograr cierto grado de multiprogramación y una utilización adecuada de la memoria, es importante la gestión de la misma. Existen varios métodos de gestión de memoria, que reflejan diversos enfoques, y la eficacia de cada algoritmo depende de la situación.

Visite los siguientes recursos para obtener más información:

- Desmitificación de la gestión de memoria en los lenguajes de programación modernos
- Gestión de la memoria en el sistema operativo

Comunicación entre procesos

La comunicación entre procesos (IPC) se refiere específicamente a los mecanismos que proporciona un sistema operativo para permitir que los procesos gestionen datos compartidos.

Visite los siguientes recursos para obtener más información:

- Comunicación entre procesos
- Comunicación entre procesos - Neso Academy

Gestión de E/S

Una de las tareas importantes de un Sistema Operativo es gestionar varios dispositivos de E/S incluyendo ratón, teclados, touchpad, unidades de disco, adaptadores de pantalla, dispositivos USB, pantallas con mapa de bits, LED, convertidor analógico-digital, interruptor de encendido/apagado, conexiones de red, E/S de audio, impresoras, etc.

Visite los siguientes recursos para obtener más información:

- Sistema operativo - Hardware de E/S
- Gestión de E/S
- Fundamentos del Sistema Operativo (Estructura de E/S)

Conceptos básicos de conexión en red

Las redes de ordenadores hacen referencia a dispositivos informáticos interconectados que pueden intercambiar datos y compartir recursos entre sí. Estos dispositivos en red utilizan un sistema de reglas, denominadas protocolos de comunicación, para transmitir información a través de tecnologías físicas o inalámbricas.

Visite los siguientes recursos para obtener más información:

- ¿Qué son las redes informáticas?

Hilos y concurrencia

Un hilo es la unidad más pequeña de procesamiento que se puede realizar en un sistema operativo. En la mayoría de los sistemas operativos modernos, un hilo existe dentro de un proceso, es decir, un único proceso puede contener varios hilos.

La concurrencia se refiere a la ejecución de múltiples hilos al mismo tiempo. Se produce en un sistema operativo cuando varios hilos de proceso se ejecutan simultáneamente. Estos hilos pueden

interactuar entre sí a través de la memoria compartida o el paso de mensajes. La concurrencia hace que se compartan recursos, lo que provoca problemas como bloqueos y escasez de recursos. Ayuda con técnicas como la coordinación de procesos, la asignación de memoria y la programación de la ejecución para maximizar el rendimiento.

Visite los siguientes recursos para obtener más información:

- ¿Cuál es la diferencia? Programas, Procesos e Hilos
- Concurrencia en el Sistema Operativo
- Introducción a Procesos e Hilos
- Introducción a la concurrencia
- Explicación de la concurrencia, los subprocesos y el paralelismo

Gestión de procesos

La gestión de procesos implica varias tareas como la creación, programación, terminación de procesos y un punto muerto. Un proceso es un programa en ejecución, que constituye una parte importante de los sistemas operativos actuales. El SO debe asignar recursos que permitan a los procesos compartir e intercambiar información. También protege los recursos de cada proceso de otros métodos y permite la sincronización entre procesos.

Visite los siguientes recursos para obtener más información:

- Sistema Operativo: Gestión de Procesos y Procesos
- Gestión de Procesos en el Sistema Operativo: PCB en el Sistema Operativo

Bases de datos relacionales

Una base de datos relacional es **un tipo de base de datos que almacena y proporciona acceso a puntos de datos que están relacionados entre sí**. Las bases de datos relacionales almacenan los datos en una serie de tablas. Las interconexiones entre las tablas se especifican como claves externas. Una clave externa es una referencia única de una fila de una tabla relacional a otra fila de una tabla, que puede ser la misma tabla, pero lo más habitual es que sea una tabla diferente.

Visite los siguientes recursos para obtener más información:

- Bases de datos relacionales
- 51 años de bases de datos relacionales
- Bases de datos y SQL
- Introducción a las Bases de Datos Relacionales
- ¿Qué es una base de datos relacional?

PostgreSQL

PostgreSQL, también conocido como Postgres, es un sistema de gestión de bases de datos relacionales gratuito y de código abierto que hace hincapié en la extensibilidad y el cumplimiento de SQL.

Visite los siguientes recursos para obtener más información:

- Visite Dedicated PostgreSQL DBA Roadmap (en inglés)
- Sitio web oficial
- Qué es PostgreSQL
- Aprenda PostgreSQL - Tutorial completo para principiantes
- Aprenda PostgreSQL Tutorial - Curso completo para principiantes
- Tutorial Postgres para Principiantes

MySQL

MySQL es un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto increíblemente popular. MySQL puede utilizarse como cliente independiente o junto con otros servicios para proporcionar conectividad a bases de datos. La **M** de LAMP stack significa MySQL; sólo eso ya debería dar una idea de su prevalencia.

Visite los siguientes recursos para obtener más información:

- Sitio web de MySQL
- W3Schools - Tutorial de MySQL
- Tutorial de MySQL para principiantes
- MySQL para desarrolladores
- Tutorial de MySQL

MariaDB

El servidor MariaDB es una bifurcación del servidor MySQL desarrollada por la comunidad. Iniciado por miembros del equipo original de MySQL, MariaDB trabaja activamente con desarrolladores

externos para ofrecer el servidor SQL abierto más completo, estable y con las licencias más razonables de la industria. MariaDB se creó con la intención de ser una versión más versátil y sustitutiva de MySQL.

Visite los siguientes recursos para obtener más información:

- Sitio web de MariaDB
- MariaDB vs MySQL
- W3Schools - Tutorial de MariaDB
- Tutorial de MariaDB para principiantes en una hora

MS SQL

MS SQL (o Microsoft SQL Server) es el sistema de gestión de bases de datos relacionales (RDBMS) desarrollado por Microsoft. MS SQL utiliza el lenguaje de consulta T-SQL (Transact-SQL) para interactuar con las bases de datos relacionales. Existen muchas versiones y ediciones diferentes de MS SQL

Visite los siguientes recursos para obtener más información:

- Sitio web de MS SQL
- Tutoriales de SQL Server
- Tutorial de SQL Server para principiantes

Oracle

Oracle Database Server o a veces llamado Oracle RDBMS o incluso simplemente Oracle es un sistema de gestión de bases de datos relacionales líder en el mundo producido por Oracle Corporation.

Visite los siguientes recursos para obtener más información:

- Sitio web oficial
- Documentos oficiales
- Tutorial de Oracle SQL para principiantes

Bases de datos NoSQL

Las bases de datos NoSQL ofrecen almacenamiento y recuperación de datos con un modelo diferente al de las bases de datos relacionales "tradicionales". Las bases de datos NoSQL suelen centrarse más en el escalado horizontal, la consistencia eventual, la velocidad y la flexibilidad, y se utilizan habitualmente para aplicaciones de big data y streaming en tiempo real. NoSQL se describe a menudo como un sistema BASE (Basically Available, Soft state, Eventual consistency), a diferencia de SQL/relacional, que suele centrarse en ACID (Atomicity, Consistency, Isolation, Durability). Las estructuras de datos NoSQL más comunes son el par clave-valor, la columna ancha, el grafo y el documento.

Visite los siguientes recursos para obtener más información:

- Explicación de NoSQL
- Cómo funcionan las bases de datos NoSQL
- SQL vs NoSQL Explicado

Bases de datos de documentos

MongoDB

MongoDB es un programa de base de datos orientada a documentos multiplataforma de código fuente disponible. Clasificado como un programa de base de datos NoSQL, MongoDB utiliza documentos tipo JSON con esquemas opcionales. MongoDB está desarrollado por MongoDB Inc. y licenciado bajo la Server Side Public License (SSPL).

Visite los siguientes recursos para obtener más información:

- Visite la hoja de ruta dedicada de MongoDB
- Sitio web de MongoDB
- Documentación de MongoDB
- Sandbox en línea de MongoDB
- Ruta de aprendizaje para desarrolladores de MongoDB
- Documentación de Dynamo DB
- Guía oficial para desarrolladores

CouchDB

Bases de datos en tiempo real

Una base de datos en tiempo real se define en términos generales como un almacén de datos diseñado para recoger, procesar y/o enriquecer una serie entrante de puntos de datos (es decir, un flujo de datos) en tiempo real, normalmente inmediatamente después de que se creen los datos.

Firebase

ReThinkDB

Bases de datos clave-valor

Una base de datos clave-valor (base de datos KV) es un tipo de base de datos que almacena datos como una colección de pares clave-valor. En una base de datos KV, cada dato se identifica con una clave única y el valor es el dato asociado a esa clave.

Las bases de datos KV están diseñadas para almacenar y recuperar datos de forma rápida y eficaz, y suelen utilizarse en aplicaciones que requieren un alto rendimiento y una baja latencia. Son especialmente adecuadas para almacenar grandes cantidades de datos no estructurados, como datos de registro y perfiles de usuario.

Algunas bases de datos KV populares son Redis, Memcached y LevelDB. Estas bases de datos se utilizan a menudo en combinación con otros tipos de bases de datos, como las bases de datos relacionales o las bases de datos de documentos, para proporcionar una solución de almacenamiento de datos completa y escalable.

Visite los siguientes recursos para obtener más información:

- Bases de datos clave-valor – Wikipedia

Redis

DynamoDB

Bases de datos de series temporales

InfluxDB

InfluxDB se construyó desde cero para ser una base de datos de series temporales específica, es decir, no se reutilizó para series temporales. El tiempo estaba incorporado desde el principio. InfluxDB forma parte de una plataforma integral que soporta la recopilación, almacenamiento, monitorización, visualización y alerta de datos de series temporales. Es mucho más que una base de datos de series temporales.

Visite los siguientes recursos para obtener más información:

- Sitio web de InfluxDB
- Base de datos de series temporales

TimeScale

Bases de datos de columnas

Una **base de datos de columnas anchas** (a veces denominada base de datos de columnas) es similar a una base de datos relacional. Almacena los datos en tablas, filas y columnas. Sin embargo, a diferencia de las bases de datos relacionales, cada fila puede tener su propio formato de columnas. Las bases de datos de columnas pueden verse como una base de datos bidimensional clave-valor. Uno de estos sistemas de bases de datos es **Apache Cassandra**.

Advertencia: ¡tenga en cuenta que "base de datos columnar" y "base de datos de columnas" son dos términos diferentes!

Visite los siguientes recursos para obtener más información:

- Apache Cassandra
- Base de datos Apache Cassandra - Curso completo para principiantes

Cassandra

HBase

Bases de datos gráficas

Una base de datos gráfica almacena nodos y relaciones en lugar de tablas o documentos. Los datos se almacenan de la misma forma que se esbozan ideas en una pizarra. Los datos se almacenan sin restringirlos a un modelo predefinido, lo que permite una forma muy flexible de concebirlos y utilizarlos.

Visite los siguientes recursos para obtener más información:

- ¿Qué es una base de datos gráfica?
- Bases de datos gráficas frente a bases de datos relacionales

Neo4j

Más sobre bases de datos

Una base de datos es una colección de datos útiles de una o varias organizaciones relacionadas entre sí, estructurada de forma que los datos se conviertan en un activo para la organización. Un sistema de gestión de bases de datos es un programa informático diseñado para ayudar a mantener y extraer grandes colecciones de datos en el momento oportuno.

Visite los siguientes recursos para obtener más información:

- Oracle: ¿Qué es una base de datos?
- Prisma.io: ¿Qué son las bases de datos?

ORMs

El mapeo objeto-relacional (ORM) es una técnica que permite consultar y manipular datos de una base de datos utilizando un paradigma orientado a objetos. Cuando se habla de ORM, la mayoría de la gente se refiere a una biblioteca que implementa la técnica de mapeo objeto-relacional, de ahí la frase "un ORM".

Visite los siguientes recursos para obtener más información:

- Mapeo Objeto-Relacional - Wikipedia
- ¿Qué es un ORM y cómo debo utilizarlo?
- ¿Qué es un ORM, cómo funciona y cómo debo usarlo?

ACID

ACID son las cuatro propiedades de los sistemas de bases de datos relacionales que ayudan a garantizar que podamos realizar las transacciones de forma fiable. Es un acrónimo que se refiere a la presencia de cuatro propiedades: atomicidad, consistencia, aislamiento y durabilidad.

Visite los siguientes recursos para obtener más información:

- ¿Qué es una base de datos conforme a ACID?
- ¿Qué es la Conformidad ACID? Atomicidad, Consistencia, Aislamiento
- Explicación de ACID: Atómica, Consistente, Aislada y Duradera

Transacciones

En pocas palabras, una transacción de base de datos es una secuencia de múltiples operaciones realizadas en una base de datos, y todas sirven como una única unidad lógica de trabajo, que se realiza en su totalidad o no se realiza en absoluto. En otras palabras, nunca se da el caso de que sólo se realicen la mitad de las operaciones y se guarden los resultados.

Visite los siguientes recursos para obtener más información:

- ¿Qué son las transacciones?

Problema de N más uno

El problema de la consulta N+1 se produce cuando el código ejecuta N sentencias de consulta adicionales para obtener los mismos datos que se podrían haber obtenido al ejecutar la consulta principal.

Visite los siguientes recursos para obtener más información:

- Explicación detallada del problema N+1

Normalización de bases de datos

La normalización de bases de datos es el proceso de estructurar una base de datos relacional de acuerdo con una serie de formas normales para reducir la redundancia de datos y mejorar su integridad. Fue propuesta por primera vez por Edgar F. Codd como parte de su modelo relacional.

La normalización consiste en organizar las columnas (atributos) y las tablas (relaciones) de una base de datos para garantizar que sus dependencias se cumplen correctamente mediante las restricciones de integridad de la base de datos. Se consigue aplicando algunas reglas formales, ya sea mediante un proceso de síntesis (creación de un nuevo diseño de base de datos) o de descomposición (mejora de un diseño de base de datos existente).

Visite los siguientes recursos para obtener más información:

- ¿Qué es la normalización en DBMS (SQL)? Base de datos 1NF, 2NF, 3NF, BCNF con ejemplo
- Normalización de bases de datos
- Concepto básico de normalización de bases de datos

Modos de fallo

Existen diferentes modos de fallo que pueden producirse en una base de datos, entre los que se incluyen:

- Contención de lectura: Se produce cuando varios clientes o procesos intentan leer datos de la misma ubicación de la base de datos al mismo tiempo, lo que puede provocar retrasos o errores.
- Contención de escritura: Se produce cuando varios clientes o procesos intentan escribir datos en la misma ubicación de la base de datos al mismo tiempo, lo que puede provocar retrasos o errores.
- Rebaño atronador: Se produce cuando un gran número de clientes o procesos intentan acceder simultáneamente al mismo recurso, lo que puede provocar el agotamiento de los recursos y reducir el rendimiento.
- Cascada: Se produce cuando un fallo en una parte del sistema de base de datos causa una reacción en cadena que provoca fallos en otras partes del sistema.
- Bloqueo: Se produce cuando dos o más transacciones esperan a que la otra libere un bloqueo sobre un recurso, lo que provoca una paralización.
- Corrupción: Se produce cuando los datos de la base de datos se corrompen, lo que puede provocar errores o resultados inesperados al leer o escribir en la base de datos.
- Fallo de hardware: Se produce cuando fallan componentes de hardware, como unidades de disco o memoria, lo que puede provocar la pérdida o corrupción de datos.

- Fallo de software: Se produce cuando fallan componentes de software, como el sistema de gestión de la base de datos o la aplicación, lo que puede provocar errores o resultados inesperados.
- Fallo de red: Se produce cuando se pierde la conexión de red entre la base de datos y el cliente, lo que puede provocar errores o tiempos de espera al intentar acceder a la base de datos.
- Ataque de denegación de servicio (DoS): Se produce cuando un actor malicioso intenta saturar la base de datos con peticiones, lo que provoca el agotamiento de los recursos y la reducción del rendimiento.

Perfilar el rendimiento

Existen varias formas de perfilar el rendimiento de una base de datos:

- Supervisar el rendimiento del sistema: Puedes utilizar herramientas como el Administrador de tareas de Windows o el comando top de Unix/Linux para supervisar el rendimiento de tu servidor de bases de datos. Estas herramientas permiten ver el uso general de CPU, memoria y disco del sistema, lo que puede ayudar a identificar cuellos de botella en los recursos.
- Utiliza herramientas específicas para bases de datos: La mayoría de los sistemas de gestión de bases de datos (SGBD) tienen sus propias herramientas para supervisar el rendimiento. Por ejemplo, Microsoft SQL Server tiene SQL Server Management Studio (SSMS) y la vista de gestión dinámica sys.dm_os_wait_stats, mientras que Oracle tiene Oracle Enterprise Manager y la vista v$waitstat. Estas herramientas permiten ver métricas de rendimiento específicas, como la cantidad de tiempo de espera en bloqueos o el número de lecturas y escrituras físicas.
- Utiliza herramientas de terceros: También existen varias herramientas de terceros que pueden ayudarle a perfilar el rendimiento de una base de datos. Algunos ejemplos son SolarWinds Database Performance Analyzer, Quest Software Foglight y Redgate SQL Monitor. Estas herramientas suelen proporcionar análisis de rendimiento más exhaustivos y pueden ayudarle a identificar problemas específicos o cuellos de botella.
- Analice las consultas lentas: Si tiene consultas específicas que se están ejecutando lentamente, puede utilizar herramientas como EXPLAIN PLAN o SHOW PLAN en MySQL o SQL Server para ver el plan de ejecución de la consulta e identificar cualquier problema potencial. También puede utilizar herramientas como el registro de consultas lentas de MySQL o SQL Server Profiler para capturar las consultas lentas y analizarlas con más detalle.
- Supervise el rendimiento de la aplicación: Si experimenta problemas de rendimiento con una aplicación específica que utiliza la base de datos, puede utilizar herramientas como Application Insights o New Relic para supervisar el rendimiento de la aplicación e identificar cualquier problema que pueda estar relacionado con la base de datos.

Consulte la documentación de la base de datos que esté utilizando.

Escalado de bases de datos

Una base de datos es una colección de datos útiles de una o varias organizaciones relacionadas entre sí, estructurada de forma que los datos se conviertan en un activo para la organización. Un sistema de gestión de bases de datos es un programa informático diseñado para ayudar a mantener y extraer grandes colecciones de datos en el momento oportuno.

Visite los siguientes recursos para obtener más información:

- Oracle: ¿Qué es una base de datos?
- Prisma.io: ¿Qué son las bases de datos?

Índices de bases de datos

Un índice es una estructura de datos que construyes y asignas encima de una tabla existente que básicamente mira a través de tu tabla e intenta analizar y resumir para poder crear accesos directos.

Visite los siguientes recursos para obtener más información:

- Una mirada en profundidad a la Indexación de Bases de Datos
- Explicación de la indexación de bases de datos

Replicación de datos

La replicación de datos es el proceso por el cual los datos que residen en un servidor físico/virtual o en una instancia en la nube (instancia primaria) se replican o copian continuamente en un servidor secundario o en una instancia en la nube (instancia en espera). Las organizaciones replican datos para respaldar la alta disponibilidad, las copias de seguridad y/o la recuperación ante desastres.

Visite los siguientes recursos para obtener más información:

- ¿Qué es la replicación de datos?

Estrategias de fragmentación

La estrategia de fragmentación es una técnica para dividir un gran conjunto de datos en fragmentos más pequeños (fragmentos lógicos) en los que distribuimos estos fragmentos en diferentes máquinas/nodos de base de datos con el fin de distribuir la carga de tráfico. Es un buen mecanismo para mejorar la escalabilidad de una aplicación. Muchas bases de datos soportan sharding, pero no todas.

Visita los siguientes recursos para obtener más información:

- Database Sharding - Concepto de entrevista sobre diseño de sistemas
- Wikipedia - Sharding en arquitecturas de bases de datos
- Cómo la fragmentación de una base de datos puede hacerla más rápida

Teorema CAP

CAP es un acrónimo de Consistencia, Disponibilidad y Tolerancia a la Partición. Según el teorema CAP, cualquier sistema distribuido sólo puede garantizar dos de las tres propiedades en un momento dado. No puede garantizar las tres propiedades a la vez.

Visite los siguientes recursos para obtener más información:

- ¿Qué es el teorema CAP?
- Teorema CAP - Wikipedia
- Demostración ilustrada del teorema CAP
- Teorema CAP y sus aplicaciones en bases de datos NoSQL
- ¿Qué es el Teorema CAP?

Más información sobre las API

API es el acrónimo de Application Programming Interface (interfaz de programación de aplicaciones), que es un intermediario de software que permite que dos aplicaciones se comuniquen entre sí.

Visite los siguientes recursos para obtener más información:

- ¿Qué es una API?
- ¿Qué es una API? (Video)

REST

REST, o REpresentational State Transfer, es un estilo arquitectónico para proporcionar estándares entre sistemas informáticos en la web, facilitando que los sistemas se comuniquen entre sí.

Visite los siguientes recursos para obtener más información:

- ¿Qué es REST?
- Fundamentos de REST
- ¿Qué es una API REST?
- Capítulo de la tesis doctoral de Roy Fieldings, Representational State Transfer (REST)
- Aprenda REST: Un tutorial RESTful

APIs JSON

JSON o JavaScript Object Notation es un esquema de codificación que está diseñado para eliminar la necesidad de un código ad-hoc para cada aplicación para comunicarse con los servidores que se comunican de una manera definida. El módulo API JSON expone una implementación para almacenes de datos y estructuras de datos, como tipos de entidad, paquetes y campos.

Visite los siguientes recursos para obtener más información:

- Sitio web oficial
- Documentación oficial
- API JSON: Explicado en 4 minutos

SOAP

Simple Object Access Protocol (SOAP) es un protocolo de mensajes para el intercambio de información entre sistemas y aplicaciones. Cuando se trata de interfaces de programación de aplicaciones (API), una API SOAP se desarrolla de forma más estructurada y formalizada. Los mensajes SOAP pueden transportarse a través de una variedad de protocolos de nivel inferior, incluido el Protocolo de Transferencia de Hipertexto (HTTP) relacionado con la web.

Visite los siguientes recursos para obtener más información:

- w3school Explicación de SOAP

gRPC

gRPC es un framework RPC universal de código abierto y alto rendimiento.

RPC significa Remote Procedure Call, hay un debate en curso sobre lo que la g significa. RPC es un protocolo que permite a un programa ejecutar un procedimiento de otro programa ubicado en otro ordenador. La gran ventaja es que el desarrollador no necesita codificar los detalles de la interacción remota. El procedimiento remoto se llama como cualquier otra función. Pero el cliente y el servidor pueden codificarse en lenguajes diferentes.

Visite los siguientes recursos para obtener más información:

- Sitio web de gRPC
- Documentación sobre gRPC
- ¿Qué es GRPC?
- ¿Qué es GRPC? (Video)

GraphQL

GraphQL es un lenguaje de consulta y un sistema de ejecución para API (interfaces de programación de aplicaciones). Está diseñado para ofrecer a los clientes una forma flexible y eficaz de solicitar datos a los servidores, y a menudo se utiliza como alternativa a las API REST (transferencia de estado representacional).

Una de las principales características de GraphQL es su capacidad para especificar exactamente los datos que se necesitan, en lugar de recibir un conjunto fijo de datos de un punto final. Esto permite a los clientes solicitar sólo los datos que necesitan y reduce la cantidad de datos que hay que transferir por la red.

GraphQL también permite definir la estructura de los datos que devuelve el servidor, lo que permite a los clientes solicitar datos de forma predecible y flexible. Esto facilita la creación y el mantenimiento de aplicaciones cliente que dependen de los datos del servidor.

GraphQL se utiliza ampliamente en aplicaciones web y móviles modernas, y cuenta con el apoyo de una comunidad de desarrolladores amplia y activa.

Visite los siguientes recursos para obtener más información:

- Sitio web oficial de GraphQL

HATEOAS

HATEOAS es el acrónimo de **H**ypermedia **A**s **T**he **E**ngine **O**f **A**pplication **S**tate (Hipermedia como motor del estado de la aplicación). Se trata del concepto de que cuando se envía información a través de una API RESTful, el documento recibido debe contener todo lo que el cliente necesita para analizar y utilizar los datos, es decir, no tiene que ponerse en contacto con ningún otro punto final que no se mencione explícitamente en el documento.

Visite los siguientes recursos para obtener más información:

- Oktane17: Diseño de API REST + JSON (3:56 - 5:57)

Especificación OpenAPI

La especificación OpenAPI (OAS) define una interfaz estándar e independiente del idioma para las API RESTful que permite tanto a humanos como a ordenadores descubrir y comprender las capacidades del servicio sin necesidad de acceder al código fuente, la documentación o mediante la inspección del tráfico de red. Cuando se define correctamente, un consumidor puede entender e interactuar con el servicio remoto con una cantidad mínima de lógica de implementación.

Una definición OpenAPI puede ser utilizada por herramientas de generación de documentación para mostrar la API, herramientas de generación de código para generar servidores y clientes en varios lenguajes de programación, herramientas de prueba y muchos otros casos de uso.

Visite los siguientes recursos para obtener más información:

- Sitio web de la especificación OpenAPI
- Open API Live Editor
- Guía de formación oficial
- OpenAPI 3.0: Cómo diseñar y documentar API con la última especificación OpenAPI 3.0

Autenticación

El proceso de autenticación de la API valida la identidad del cliente que intenta establecer una conexión mediante un protocolo de autenticación. El protocolo envía las credenciales del cliente remoto que solicita la conexión al servidor de acceso remoto en texto plano o cifrado. El servidor sabe entonces si puede conceder acceso a ese cliente remoto o no.

Aquí está la lista de formas comunes de autenticación:

- Autenticación JWT
- Autenticación basada en token
- Autenticación basada en sesión
- Autenticación básica
- OAuth - Autorización abierta
- SSO - Inicio de sesión único

Visite los siguientes recursos para obtener más información:

- Autenticación de usuarios: Conceptos básicos y consejos
- Una visión general sobre los métodos de autenticación
- SSO - Inicio de sesión único
- OAuth - Autorización abierta
- Autenticación JWT
- Autenticación basada en token
- Autenticación basada en sesión
- Autenticación básica

OAuth

OAuth significa **O**pen **A**uthorization y es un estándar abierto para la autorización. Funciona para autorizar dispositivos, API, servidores y aplicaciones utilizando tokens de acceso en lugar de credenciales de usuario, lo que se conoce como "acceso delegado seguro".

En su forma más simple, OAuth delega la autenticación en servicios como Facebook, Amazon o Twitter y autoriza a las aplicaciones de terceros a acceder a la cuenta del usuario **sin** tener que introducir su nombre de usuario y contraseña.

Se utiliza principalmente para REST/APIs y sólo proporciona un alcance limitado de los datos de un usuario.

Visite los siguientes recursos para obtener más información:

- Okta - ¿Qué diablos es OAuth?
- DigitalOcean - Introducción a OAuth 2
- En qué consiste realmente OAuth
- OAuth 2.0: Una visión general

Autenticación básica

Dado el nombre de "autenticación básica", no debes confundir la autenticación básica con la autenticación estándar de nombre de usuario y contraseña. La autenticación básica es parte de la especificación HTTP, y los detalles se pueden **encontrar en el RFC7617**.

Como forma parte de las especificaciones HTTP, todos los navegadores tienen soporte nativo para la "Autenticación Básica HTTP".

Visite los siguientes recursos para obtener más información:

- Autenticación básica HTTP
- Autenticación básica HTTP ilustrada

Autenticación por token

La autenticación basada en tokens es un protocolo que permite a los usuarios verificar su identidad y recibir a cambio un token de acceso único. Durante el periodo de validez del token, los usuarios acceden al sitio web o aplicación para el que se ha emitido el token, en lugar de tener que volver a introducir las credenciales cada vez que regresan a la misma página web, aplicación o cualquier recurso protegido con ese mismo token.

Los tokens de autenticación funcionan como un billete sellado. El usuario conserva el acceso mientras el token siga siendo válido. Una vez que el usuario cierra la sesión o sale de una aplicación, el token queda invalidado.

La autenticación basada en tokens es diferente de las técnicas tradicionales de autenticación basadas en contraseñas o servidores. Los tokens ofrecen una segunda capa de seguridad, y los administradores tienen un control detallado de cada acción y transacción.

Sin embargo, el uso de tokens requiere algunos conocimientos de programación. La mayoría de los desarrolladores aprenden las técnicas con rapidez, pero existe una curva de aprendizaje.

Visite los siguientes recursos para obtener más información:

- ¿Qué es la autenticación basada en tokens?

JWT

JWT son las siglas de JSON Web Token, un estándar/metodología abierta de encriptación basada en tokens que se utiliza para transferir información de forma segura como un objeto JSON. Clientes y servidores utilizan JWT para compartir información de forma segura, conteniendo el JWT objetos JSON codificados y reclamaciones. Los tokens JWT están diseñados para ser compactos, seguros de usar dentro de URLs, e ideales para contextos SSO.

Visite los siguientes recursos para obtener más información:

- Sitio web jwt.io
- Introducción a los tokens web JSON
- ¿Qué es JWT?
- Qué es JWT y por qué debería utilizar JWT
- ¿Qué es JWT? Explicación de los tokens web JSON

Autenticación basada en cookies

Las cookies son fragmentos de datos que se utilizan para identificar al usuario y sus preferencias. El navegador devuelve la cookie al servidor cada vez que se solicita la página. Las cookies específicas, como las cookies HTTP, se utilizan para realizar la autenticación basada en cookies para mantener la sesión de cada usuario.

Visite los siguientes recursos para obtener más información:

- ¿Cómo funciona la autenticación basada en cookies?

OpenID

OpenID es un protocolo que utiliza los mecanismos de autorización y autenticación de OAuth 2.0 y que actualmente está ampliamente adoptado por muchos proveedores de identidad en Internet. Resuelve el problema de la necesidad de compartir la información personal de los usuarios entre muchos servicios web diferentes (por ejemplo, tiendas en línea, foros de debate, etc.).

Visite los siguientes recursos para obtener más información:

- Sitio web oficial
- Qué es OpenID
- OAuth vs OpenID
- Guía ilustrada de OAuth y OpenID Connect
- OAuth 2.0 y OpenID Connect (en inglés)

Lenguaje de marcado de aserción de seguridad (SAML)

SAML son las siglas de Security Assertion Markup Language. Es un estándar basado en XML para el intercambio de datos de autenticación y autorización entre partes, en particular entre un proveedor de identidades (IdP) y un proveedor de servicios (SP). En un sistema basado en SAML, un usuario

solicita acceso a un recurso protegido. El proveedor de servicios solicita al proveedor de identidad que autentique al usuario y confirme si se le concede el acceso al recurso.

Ventajas de SAML

Algunas de las ventajas de utilizar SAML son:

- Inicio de sesión único (SSO): Los usuarios pueden iniciar sesión una vez en el IdP y acceder a múltiples proveedores de servicios sin necesidad de autenticarse de nuevo.
- Mayor seguridad: No es necesario que el proveedor de servicios almacene y gestione las contraseñas y credenciales de usuario, lo que reduce los posibles vectores de ataque.
- Mayor eficacia: Como los usuarios ya no necesitan mantener múltiples conjuntos de credenciales, la gestión del acceso se vuelve más fácil tanto para el usuario como para los administradores del sistema.
- Interoperabilidad: SAML permite que una amplia gama de aplicaciones trabaje juntas, independientemente de la tecnología o plataforma subyacente.

Componentes de SAML

En la arquitectura SAML intervienen tres componentes principales:

1. **Proveedor de identidades (IdP)**: La entidad que gestiona las identidades de los usuarios y los autentica proporcionando tokens de seguridad, también llamados aserciones.
2. **Proveedor de servicios (SP)**: La entidad que proporciona un servicio (como una aplicación web o API) y confía en el proveedor de identidades para autenticar a los usuarios y conceder/denegar el acceso a los recursos.
3. **Usuario/Principal**: El usuario final que desea acceder al servicio prestado por el proveedor de servicios.

Flujo de trabajo SAML

El proceso de autenticación SAML consta de los siguientes pasos:

- El usuario solicita acceso a un recurso protegido al proveedor de servicios.
- Si el usuario aún no está autenticado, el proveedor de servicios genera y envía una solicitud de autenticación SAML al proveedor de identidades.
- El proveedor de identidad autentica al usuario (utilizando, por ejemplo, un nombre de usuario y una contraseña, autenticación multifactor u otro método).
- El proveedor de identidad construye una respuesta SAML, que incluye detalles sobre el usuario y afirma si el usuario está autorizado a acceder al recurso solicitado.
- La respuesta SAML se envía de vuelta al proveedor de servicios, normalmente a través del navegador web o el cliente API del usuario.
- El proveedor de servicios procesa la respuesta SAML, extrae la información necesaria y concede o deniega el acceso al usuario basándose en la afirmación del proveedor de identidades.

Con SAML, puede agilizar la autenticación y autorización de usuarios en varias aplicaciones y sistemas, proporcionando una mejor experiencia de usuario y mejorando la seguridad general de su backend.

Almacenamiento en caché

El almacenamiento en caché es una técnica que consiste en guardar datos o información de uso frecuente en una memoria local, durante un periodo de tiempo determinado. Así, la próxima vez que el cliente solicite la misma información, en lugar de recuperarla de la base de datos, la obtendrá de la memoria local. La principal ventaja del almacenamiento en caché es que mejora el rendimiento al reducir la carga de procesamiento.

Almacenamiento en caché del lado del cliente

El almacenamiento en caché del lado del cliente es el almacenamiento de datos de red en una caché local para su futura reutilización. Cuando una aplicación obtiene datos de la red, los almacena en una caché local. Una vez que un recurso ha sido almacenado en caché, el navegador utiliza la caché en futuras peticiones de ese recurso para aumentar el rendimiento.

Visite los siguientes recursos para obtener más información:

- Todo lo que necesita saber sobre la caché HTTP

Almacenamiento en caché del lado del servidor

El almacenamiento en caché del lado del servidor almacena temporalmente archivos y datos web en el servidor de origen para reutilizarlos más tarde.

Cuando el usuario solicita la página web por primera vez, el sitio web sigue el proceso normal de recuperación de datos del servidor y genera o construye la página web del sitio. Una vez realizada la solicitud y recibida la respuesta, el servidor copia la página web y la almacena en la caché.

La próxima vez que el usuario vuelva a visitar el sitio web, cargará la copia ya guardada o en caché de la página web, haciéndola así más rápida.

Visite los siguientes recursos para saber más:

- Caché de servidor
- Caché del servidor y caché del cliente

Redis

Redis es un **almacén de estructuras de datos** en memoria de código abierto (con licencia BSD) utilizado como base de datos, caché, agente de mensajes y motor de streaming. Redis proporciona estructuras de datos como **cadenas de texto**, **hashes**, **listas**, **conjuntos**, **conjuntos ordenados** con consultas de rango, **mapas de bits**, **hyperlogs**, **índices geoespaciales** y **flujos**. Redis incorpora **replicación**, **scripts Lua**, **desalojo LRU**, **transacciones** y diferentes niveles de **persistencia en disco**, y proporciona alta disponibilidad a través de **Redis Sentinel** y particionamiento automático con **Redis Cluster**.

Visite los siguientes recursos para obtener más información:

- Sitio web de Redis
- Redis en 100 segundos

Memcached

Memcached es un sistema de caché de memoria distribuida de propósito general. Se utiliza a menudo para acelerar sitios web dinámicos basados en bases de datos, almacenando datos y objetos en la memoria RAM para reducir el número de veces que una fuente de datos externa (como una base de datos o API) debe ser leída. Memcached es un software gratuito y de código abierto, con licencia BSD revisada. Memcached funciona en sistemas operativos tipo Unix (Linux y macOS) y en Microsoft Windows. Depende de la biblioteca `libevent`.

Las API de Memcached proporcionan una tabla hash muy grande distribuida entre varias máquinas. Cuando la tabla está llena, las inserciones posteriores hacen que los datos más antiguos se purguen en el orden de uso menos reciente (LRU). Las aplicaciones que utilizan Memcached suelen almacenar las peticiones y adiciones en la RAM antes de recurrir a un almacén de respaldo más lento, como una base de datos.

Memcached no tiene un mecanismo interno para rastrear los fallos que puedan ocurrir. Sin embargo, algunas utilidades de terceros proporcionan esta funcionalidad.

Visite los siguientes recursos para obtener más información:

- Memcached, de Wikipedia
- Memcached, de Github oficial
- Tutorial de Memcached

CDN (Red de Entrega de Contenidos)

Un servicio de Red de Entrega de Contenidos (CDN) tiene como objetivo proporcionar una alta disponibilidad y mejorar el rendimiento de los sitios web. Esto se consigue con una entrega rápida de los activos y contenidos del sitio web, normalmente a través de puntos finales geográficamente más cercanos a las solicitudes de los clientes. Las CDN comerciales tradicionales (Amazon CloudFront, Akamai, CloudFlare y Fastly) ofrecen servidores en todo el mundo que pueden utilizarse para este fin. Servir activos y contenidos a través de una CDN reduce el ancho de banda en el alojamiento de sitios web, proporciona una capa adicional de almacenamiento en caché para reducir posibles interrupciones y también puede mejorar la seguridad del sitio web.

Visite los siguientes recursos para obtener más información:

- CloudFlare - ¿Qué es una CDN? | ¿Cómo funcionan las CDN?
- Wikipedia - Red de entrega de contenidos
- ¿Qué es una CDN en la nube?
- ¿Qué es una Red de Entrega de Contenidos (CDN)?

Conocimientos de seguridad web

La seguridad web se refiere a las medidas de protección tomadas por los desarrolladores para proteger las aplicaciones web de amenazas que podrían afectar al negocio.

Visite los siguientes recursos para obtener más información:

- Por qué es importante HTTPS
- Wikipedia - OWASP
- Lista de comprobación de seguridad de aplicaciones web OWASP
- Los 10 principales riesgos de seguridad OWASP
- Cheatsheet OWASP
- Política de seguridad de contenidos (CSP)

Algoritmos hash

MD5

MD5 (Message-Digest Algorithm 5) es una función hash cuyo uso se desaconseja actualmente debido a sus numerosas vulnerabilidades. Se sigue utilizando como suma de comprobación para verificar la integridad de los datos.

Visite los siguientes recursos para obtener más información:

- Wikipedia - MD5
- ¿Qué es MD5?
- ¿Por qué MD5 no es seguro?

Familia SHA

SHA (Secure Hash Algorithms) es una familia de funciones hash criptográficas creada por el NIST (National Institute of Standards and Technology). La familia incluye:

- SHA-0: Publicado en 1993, es el primer algoritmo de la familia. Poco después de su lanzamiento, dejó de utilizarse debido a un fallo importante no revelado.
- SHA-1: Creado para sustituir a SHA-0 y que se parece a MD5, este algoritmo se considera inseguro desde 2010.
- SHA-2: No es un algoritmo, sino un conjunto de ellos, siendo SHA-256 y SHA-512 los más populares. SHA-2 sigue siendo seguro y ampliamente utilizado.
- SHA-3: Nacido en una competición, es el miembro más nuevo de la familia. SHA-3 es muy seguro y no presenta los mismos defectos de diseño que sus hermanos.

Visite los siguientes recursos para obtener más información:

- Wikipedia - SHA-1
- Wikipedia - SHA-2
- Wikipedia - SHA-3

Scrypt

Scrypt (pronunciado "ess crypt") es una función de hash de contraseñas (como bcrypt). Está diseñado para utilizar mucho hardware, lo que dificulta los ataques de fuerza bruta. Scrypt se utiliza principalmente como algoritmo de prueba de trabajo para criptomonedas.

Visite los siguientes recursos para obtener más información:

- Wikipedia – Scrypt

Bcrypt

bcrypt es una función de hash de contraseñas, que ha demostrado ser fiable y segura desde su lanzamiento en 1999. Se ha implementado en los lenguajes de programación más utilizados.

Visita los siguientes recursos para obtener más información:

- Paquete npm bcrypts
- Comprender bcrypt
- Explicación de bcrypt

Seguridad API

HTTPS

HTTPS es una forma segura de enviar datos entre un servidor web y un navegador.

Una comunicación a través de HTTPS comienza con la fase de "handshake", durante la cual el servidor y el cliente acuerdan cómo encriptar la comunicación, en concreto eligen un algoritmo de encriptación y una clave secreta. Tras el apretón de manos, toda la comunicación entre el servidor y el cliente se cifrará utilizando el algoritmo y la clave acordados.

La fase de "apretón de manos" utiliza un tipo particular de criptografía, llamada criptografía asimétrica, para comunicarse de forma segura, aunque el cliente y el servidor aún no hayan acordado una clave secreta. Después de la fase de apretón de manos, la comunicación HTTPS se cifra con criptografía simétrica, que es mucho más eficiente, pero requiere que tanto el cliente como el servidor conozcan la clave secreta.

Visite los siguientes recursos para obtener más información:

- ¿Qué es HTTPS?
- Por qué es importante HTTPS
- Activar HTTPS en sus servidores
- Cómo funciona HTTPS (cómic)
- Explicación de SSL, TLS, HTTP y HTTPS
- HTTPS - Historias del terreno
- HTTPS explicado con palomas mensajeras

Riesgos de seguridad OWASP

OWASP o Open Web Application Security Project es una comunidad en línea que produce artículos, metodologías, documentación, herramientas y tecnologías de libre acceso en el campo de la seguridad de las aplicaciones web.

Visite los siguientes recursos para obtener más información:

- Wikipedia - OWASP
- Estándar de verificación de seguridad de aplicaciones OWASP
- Los 10 mayores riesgos de seguridad OWASP
- Cheatsheets OWASP

CORS

Cross-Origin Resource Sharing (CORS) es un mecanismo basado en cabeceras HTTP que permite a un servidor indicar cualquier origen (dominio, esquema o puerto) distinto del suyo desde el que un navegador debe permitir la carga de recursos. Visite los siguientes recursos para obtener más información:

- Compartición de recursos entre orígenes (CORS)
- CORS en 100 segundos
- CORS en 6 minutos

SSL/TLS

Secure Sockets Layer (SSL) y Transport Layer Security (TLS) son protocolos criptográficos utilizados para proporcionar seguridad en las comunicaciones por Internet. Estos protocolos cifran los datos que se transmiten a través de la red, de modo que cualquiera que intente interceptar los paquetes no podrá interpretar los datos. Una diferencia que es importante conocer es que SSL está ahora obsoleto debido a fallos de seguridad, y la mayoría de los navegadores web modernos ya no lo soportan. Sin embargo, TLS sigue siendo seguro y está ampliamente soportado, por lo que es preferible utilizar TLS.

Visite los siguientes recursos para obtener más información:

- Wikipedia - SSL/TLS
- Cloudflare - ¿Qué es SSL?

Política de seguridad de contenidos

La Política de Seguridad de Contenidos es un estándar de seguridad informática introducido para prevenir ataques de cross-site scripting, clickjacking y otros ataques de inyección de código resultantes de la ejecución de contenido malicioso en el contexto de una página web de confianza.

Visite los siguientes recursos para obtener más información:

- MDN - Política de seguridad de contenidos (CSP)
- Google Devs - Política de seguridad de contenidos (CSP)

Seguridad del servidor

Infórmese sobre la seguridad de su servidor y cómo protegerlo. Éstos son algunos de los temas que se me ocurren:

- Utilizar un cortafuegos: Una de las formas más eficaces de proteger un servidor es utilizar un cortafuegos para bloquear todo el tráfico entrante innecesario. Para ello, puedes utilizar iptables en sistemas Linux o un cortafuegos de hardware.
- Cierre los puertos innecesarios: Asegúrese de cerrar todos los puertos que no sean necesarios para que su servidor funcione correctamente. Esto reducirá la superficie de ataque de su servidor y dificultará el acceso de los atacantes.
- Utilice contraseñas seguras: Utilice contraseñas largas y complejas para todas sus cuentas, y considere la posibilidad de utilizar un gestor de contraseñas para almacenarlas de forma segura.
- Mantén tu sistema actualizado: Asegúrate de mantener tu sistema operativo y software actualizados con los últimos parches de seguridad. Esto ayudará a evitar que los atacantes se aprovechen de las vulnerabilidades.
- Utilice SSL/TLS para la comunicación: Utilice Secure Sockets Layer (SSL) o Transport Layer Security (TLS) para cifrar la comunicación entre su servidor y los dispositivos cliente. Esto le ayudará a protegerse contra los ataques de intermediario y otros tipos de ciberamenazas.
- Utilice un sistema de detección de intrusos (IDS): un IDS supervisa el tráfico de la red y le alerta de cualquier actividad sospechosa, lo que puede ayudarle a identificar y responder a tiempo a posibles amenazas.
- Active la autenticación de dos factores: La autenticación de dos factores añade una capa adicional de seguridad a tus cuentas al requerir una segunda forma de autenticación, como un código enviado a tu teléfono, además de tu contraseña.

Aprenda también sobre OpenSSL y a crear su propia PKI, así como a gestionar certificados, renovaciones y autenticación mutua de clientes con certificados x509.

Pruebas

Una de las claves para crear software que cumpla los requisitos sin defectos son las pruebas (testing). Las pruebas de software ayudan a los desarrolladores a saber que están creando el software correcto. Cuando las pruebas se ejecutan como parte del proceso de desarrollo (a menudo con herramientas de integración continua), generan confianza y evitan regresiones en el código.

Visite los siguientes recursos para obtener más información:

- ¿Qué son las pruebas de software?
- Pirámide de pruebas

Pruebas de integración

Las pruebas de integración son una amplia categoría de pruebas en las que se integran varios módulos de software y se prueban como un grupo. Su objetivo es probar la interacción entre varios servicios, recursos o módulos. Por ejemplo, la interacción de una API con un servicio backend, o de un servicio con una base de datos.

Visite los siguientes recursos para obtener más información:

- Pruebas de integración
- Cómo integrar y probar su pila de tecnología
- ¿Qué es la prueba de integración?

Pruebas unitarias

Las pruebas unitarias consisten en comprobar las unidades individuales (módulos, funciones/métodos, rutinas, etc.) del software para garantizar su corrección. Estas pruebas de bajo nivel garantizan que los componentes más pequeños funcionen correctamente y alivian la carga de las pruebas de alto nivel. Generalmente, un desarrollador escribe estas pruebas durante el proceso de desarrollo y se ejecutan como pruebas automatizadas.

Visite los siguientes recursos para obtener más información:

- Tutorial de pruebas unitarias
- ¿Qué son las pruebas unitarias?

Pruebas funcionales

Las pruebas funcionales consisten en comprobar que el software cumple los requisitos funcionales. Por lo general, es una forma de prueba de caja negra en la que el probador no tiene conocimiento del código fuente; la prueba se realiza proporcionando la entrada y comparando la salida esperada/real. Contrasta con las pruebas no funcionales, que incluyen pruebas de rendimiento, carga, escalabilidad y penetración.

Visite los siguientes recursos para obtener más información:

- ¿Qué es la Prueba Funcional?
- Pruebas funcionales frente a pruebas no funcionales

CI/CD

CI/CD (Integración Continua/Despliegue Continuo) es la práctica de automatizar la creación, las pruebas y el despliegue de aplicaciones con el objetivo principal de detectar problemas de forma temprana y proporcionar versiones más rápidas al entorno de producción.

Visite los siguientes recursos para obtener más información:

- DevOps CI/CD explicado en 100 segundos por Fireship
- Automatice sus flujos de trabajo con acciones de GitHub
- ¿Qué es CI/CD?
- Introducción: Integración continua y entrega continua (CI/CD)
- 3 formas de utilizar la automatización en los procesos de CI/CD
- Artículos sobre CI/CD

Principios de diseño y desarrollo

En esta sección, discutiremos algunos principios esenciales de diseño y desarrollo a seguir mientras se construye el backend de cualquier aplicación. Estos principios asegurarán que el backend sea eficiente, escalable y mantenible.

1. Separación de intereses (SoC)

La separación de intereses es un principio fundamental que establece que las diferentes funcionalidades de un sistema deben ser lo más independientes posible. Este enfoque mejora la capacidad de mantenimiento y la escalabilidad al permitir a los desarrolladores trabajar en componentes separados sin afectarse mutuamente. Divide tu backend en módulos y capas claras, como almacenamiento de datos, lógica de negocio y comunicación de red.

2. Reutilización

La reutilización es la capacidad de utilizar componentes, funciones o módulos en varios lugares sin duplicar el código. Al diseñar el backend, busque oportunidades en las que pueda reutilizar código existente. Utilice técnicas como la creación de funciones de utilidad, clases abstractas e interfaces para fomentar la reutilización y reducir la redundancia.

3. Mantenlo simple y estúpido (KISS)

El principio KISS afirma que cuanto más sencillo es un sistema, más fácil es entenderlo, mantenerlo y ampliarlo. Cuando diseñes el backend, intenta que la arquitectura y el código sean lo más sencillos posible. Utiliza convenciones de nomenclatura claras y estructuras modulares, y evita el exceso de ingeniería y la complejidad innecesaria.

4. No se repita (DRY)

No duplique código o funcionalidades en su backend. La duplicación puede provocar incoherencias y problemas de mantenimiento. En su lugar, céntrate en crear componentes, funciones o módulos reutilizables, que puedan compartirse en diferentes partes del backend.

5. Escalabilidad

Un sistema escalable es aquel que puede gestionar eficazmente un número creciente de usuarios, solicitudes o datos. Diseña el backend teniendo en cuenta la escalabilidad, considerando factores como el almacenamiento de datos, el almacenamiento en caché, el equilibrio de carga y el escalado horizontal (añadir más instancias del servidor backend).

6. Seguridad

La seguridad es una de las principales preocupaciones a la hora de desarrollar cualquier aplicación. Siga siempre las mejores prácticas para evitar fallos de seguridad, como proteger los datos sensibles, utilizar protocolos de comunicación seguros (por ejemplo, HTTPS), implementar mecanismos de autenticación y autorización, y sanear las entradas de usuario.

7. Pruebas

Las pruebas son cruciales para garantizar la fiabilidad y estabilidad del backend. Implemente una estrategia de pruebas exhaustiva, que incluya pruebas unitarias, de integración y de rendimiento.

Utilice herramientas de pruebas automatizadas y establezca canalizaciones de integración continua (CI) y despliegue continuo (CD) para agilizar el proceso de pruebas y despliegue.

8. Documentación

Una documentación adecuada ayuda a los desarrolladores a comprender y mantener la base de código del backend. Escriba documentación clara y concisa para su código, explicando el propósito, la funcionalidad y cómo usarlo. Además, utilice comentarios y convenciones de nomenclatura adecuadas para que el propio código sea más legible y autoexplicativo.

Siguiendo estos principios de diseño y desarrollo, estarás en el buen camino para crear un backend eficiente, seguro y mantenible para tus aplicaciones.

Patrones de diseño

Los patrones de diseño son soluciones típicas a problemas habituales en el diseño de software. Pueden dividirse en tres categorías:

- Patrones Creativos para la creación de objetos
- Patrones estructurales para proporcionar relaciones entre objetos
- Patrones de comportamiento para ayudar a definir cómo interactúan los objetos.

Visite los siguientes recursos para obtener más información:

- Patrones de diseño para humanos
- Patrones de diseño GOF
- Patrones de diseño

Diseño orientado al dominio

El diseño orientado al dominio (DDD) es un enfoque de diseño de software que se centra en modelar el software para que coincida con un dominio de acuerdo con las aportaciones de los expertos de ese dominio.

En términos de programación orientada a objetos, significa que la estructura y el lenguaje del código del software (nombres de clases, métodos de clases, variables de clases) deben coincidir con el dominio del negocio. Por ejemplo, si un software procesa solicitudes de préstamo, puede tener clases como LoanApplication y Customer, y métodos como AcceptOffer y Withdraw.

DDD conecta la implementación a un modelo en evolución y se basa en los siguientes objetivos:

- Poner el foco principal del proyecto en el dominio central y la lógica del dominio;
- Basar los diseños complejos en un modelo del dominio;
- Iniciar una colaboración creativa entre expertos técnicos y del dominio para perfeccionar iterativamente un modelo conceptual que aborde problemas concretos del dominio.

Visite los siguientes recursos para obtener más información:

- Diseño orientado al dominio de forma rápida

Desarrollo basado en pruebas

El desarrollo dirigido por pruebas (TDD) es el proceso de escribir pruebas para los requisitos del software que fallarán hasta que el software se desarrolle para cumplir esos requisitos. Una vez superadas las pruebas, el ciclo se repite para refactorizar el código o desarrollar otra característica o requisito. En teoría, esto garantiza que el software se escriba para cumplir los requisitos de la forma más sencilla, y evita defectos en el código.

Visite los siguientes recursos para obtener más información:

- ¿Qué es el desarrollo basado en pruebas (TDD)?
- Desarrollo basado en pruebas
- Agile en la práctica: Desarrollo basado en pruebas

CQRS

CQRS, o segregación de responsabilidad de consulta de comandos, define un patrón arquitectónico en el que el objetivo principal es separar el enfoque de las operaciones de lectura y escritura para un almacén de datos. CQRS también se puede utilizar junto con el patrón Event Sourcing para persistir el estado de la aplicación como una secuencia ordenada de eventos, lo que hace posible restaurar los datos a cualquier punto en el tiempo.

Visite los siguientes recursos para obtener más información:

- Patrón CQRS

Generación de eventos

La fuente de eventos es un patrón de diseño en el que el estado de un sistema se representa como una secuencia de eventos que han ocurrido a lo largo del tiempo. En un sistema basado en eventos, los cambios en el estado del sistema se registran como eventos y se almacenan en un almacén de eventos. El estado actual del sistema se obtiene reproduciendo los eventos del almacén de eventos.

Una de las principales ventajas de la fuente de eventos es que proporciona un historial claro y auditable de todos los cambios que se han producido en el sistema. Esto puede ser útil para depurar y seguir la evolución del sistema a lo largo del tiempo.

El aprovisionamiento de eventos se utiliza a menudo junto con otros patrones, como la segregación de responsabilidad de consulta de comandos (CQRS) y el diseño orientado al dominio, para construir sistemas escalables y con capacidad de respuesta con una lógica empresarial compleja. También es útil para construir sistemas que necesitan soportar la funcionalidad de deshacer/rehacer o que necesitan integrarse con sistemas externos.

Visite los siguientes recursos para obtener más información:

- Event Sourcing - Martin Fowler

Patrones arquitectónicos

Un patrón arquitectónico es una solución general y reutilizable a un problema común de arquitectura de software dentro de un contexto determinado. Los patrones arquitectónicos abordan diversas cuestiones de la ingeniería de software, como las limitaciones de rendimiento del hardware informático, la alta disponibilidad y la minimización de un riesgo empresarial.

Visite los siguientes recursos para obtener más información:

- Patrones arquitectónicos en pocas palabras

Aplicaciones monolíticas

La arquitectura monolítica es un patrón en el que una aplicación gestiona las peticiones, ejecuta la lógica de negocio, interactúa con la base de datos y crea el HTML para el front-end. En términos más sencillos, esta aplicación hace muchas cosas. Sus componentes internos están altamente acoplados y se despliegan como una unidad.

Visite los siguientes recursos para obtener más información:

- Patrón: Arquitectura monolítica
- Arquitectura monolítica - Ventajas y desventajas

Microservicios

La arquitectura de microservicios es un patrón en el que se desarrollan, mantienen y despliegan por separado servicios altamente cohesionados y débilmente acoplados. Cada componente gestiona una función individual y, cuando se combinan, la aplicación gestiona una función empresarial global.

Visite los siguientes recursos para obtener más información:

- Patrón: Arquitectura de microservicios
- ¿Qué son los microservicios?
- Microservicios 101
- Introducción: Explicación de los microservicios
- Artículos sobre microservicios

SOA

SOA, o arquitectura orientada a servicios, define una forma de hacer reutilizables los componentes de software mediante interfaces de servicio. Estas interfaces utilizan estándares de comunicación comunes de tal forma que pueden incorporarse rápidamente a nuevas aplicaciones sin tener que realizar cada vez una integración profunda.

Visite los siguientes recursos para obtener más información:

- Fundación de Arquitectura de Referencia para la Arquitectura Orientada a Servicios

Serverless

Serverless es una arquitectura en la que un desarrollador crea y ejecuta aplicaciones sin aprovisionar ni gestionar servidores. Con la computación en nube/serverless, los servidores existen, pero son gestionados por el proveedor de la nube. Los recursos se utilizan a medida que se necesitan, bajo demanda y a menudo utilizando auto escalado.

Visite los siguientes recursos para obtener más información:

- Serverless
- Servicios de AWS
- Informática serverless en 100 segundos

Malla de servicios

Una malla de servicios es una red de microservicios conectados mediante una malla de proxies inteligentes interconectados. Se utiliza para gestionar y asegurar la comunicación entre microservicios, y proporciona funciones como el equilibrio de carga, el descubrimiento de servicios y la observabilidad.

En una malla de servicios, cada microservicio suele estar representado por una instancia de un proxy ligero y transparente llamado "enviado". Los enviados gestionan la comunicación entre microservicios y proporcionan funciones como el equilibrio de carga, el enrutamiento y la seguridad.

Las mallas de servicios suelen implementarse utilizando un patrón sidecar, en el que los enviados se despliegan junto a los microservicios de los que son responsables. Esto permite desacoplar la malla de servicios de los microservicios y facilita su gestión y actualización.

Las mallas de servicios se utilizan habitualmente en arquitecturas nativas de la nube y a menudo se gestionan mediante un plano de control, que se encarga de configurar y gestionar los enviados. Algunas implementaciones populares de mallas de servicios incluyen Istio y Linkerd.

Visite los siguientes recursos para obtener más información:

- ¿Qué es una malla de servicios?

Aplicaciones de doce factores

Twelve-Factor App (Aplicación de doce factores) es una metodología para crear aplicaciones de software como servicio (SaaS) escalables y mantenibles. Se basa en un conjunto de buenas prácticas que los autores de la metodología identificaron como esenciales para crear aplicaciones modernas nativas de la nube.

La metodología Twelve-Factor App consta de los siguientes principios:

- Código base: Debe haber una única base de código para la aplicación, con múltiples despliegues.
- Dependencias: La aplicación debe declarar y aislar explícitamente sus dependencias.
- Configuración: La aplicación debe almacenar la configuración en el entorno.
- Servicios de respaldo: La aplicación debe tratar los servicios de respaldo como recursos adjuntos.

- Construir, liberar, ejecutar: La aplicación debe construirse, liberarse y ejecutarse como una unidad aislada.
- Procesos: La aplicación debe ejecutarse como uno o varios procesos sin estado.
- Vinculación de puertos: La aplicación debe exponer sus servicios a través de la vinculación de puertos.
- Concurrencia: La aplicación debe escalarse añadiendo más procesos, no añadiendo hilos.
- Desechabilidad: La aplicación debe estar diseñada para iniciarse y detenerse rápidamente.
- Paridad entre desarrollo y producción: Los entornos de desarrollo, preparación y producción deben ser lo más similares posible.
- Registros: La aplicación debe tratar los registros como flujos de eventos.
- Procesos de administración: La aplicación debe ejecutar las tareas de administración/mantenimiento como procesos únicos.

La metodología Twelve-Factor App es ampliamente adoptada por los desarrolladores de aplicaciones SaaS, y se considera una práctica recomendada para crear aplicaciones nativas de la nube que sean escalables, mantenibles y fáciles de desplegar.

Visite los siguientes recursos para obtener más información:

- La aplicación de doce factores

Motores de búsqueda

Los motores de búsqueda son una parte esencial de cualquier aplicación web, responsables de proporcionar resultados de búsqueda eficientes y relevantes para los usuarios. Almacenan y recuperan datos basándose en índices únicos, que permiten realizar búsquedas rápidas y precisas. Como desarrollador backend, es crucial comprender las funcionalidades de los motores de búsqueda y cómo integrarlos en su aplicación web.

Tipos de motores de búsqueda

Existen dos tipos principales de motores de búsqueda:

1. **Motores de búsqueda de texto completo**: Están diseñados específicamente para buscar y analizar documentos de texto. Pueden indexar de forma eficiente grandes volúmenes de texto y proporcionar resultados relevantes basados en palabras clave o frases. Algunos ejemplos populares de motores de búsqueda de texto completo son **Elasticsearch**, **Solr** y **Amazon CloudSearch**.
2. **Motores de búsqueda de bases de datos**: Los motores de bases de datos son funciones integradas en la mayoría de las bases de datos. Proporcionan capacidades de búsqueda dentro de los datos almacenados en la base de datos. Algunos ejemplos son **MySQL FULLTEXT search** y **PostgreSQL Full-Text Search**.

Conceptos clave

Cuando se trata de motores de búsqueda, es importante entender estos conceptos clave:

- **Indexación**: El proceso de analizar y almacenar datos en un formato optimizado para una búsqueda y recuperación rápidas.
- **Tokenización**: Descomposición del texto en palabras o términos individuales (también conocidos como tokens), para una indexación y búsqueda eficientes.
- **Consulta**: El acto de buscar los datos indexados formulando una pregunta específica o solicitando información basada en palabras clave o frases.
- **Puntuación de relevancia**: Puntuación asignada a cada resultado de búsqueda que indica su grado de coincidencia con la consulta, basada en algoritmos y modelos de relevancia.

Integración

Para integrar un motor de búsqueda en tu aplicación web, normalmente seguirías estos pasos:

1. **Elija el motor de búsqueda**: identifique el motor de búsqueda que mejor se adapte a las necesidades de su aplicación, teniendo en cuenta factores como la escalabilidad, el rendimiento y la facilidad de integración.
2. **Indexe sus datos**: Analice y almacene sus datos utilizando el motor de búsqueda elegido. Este proceso puede implicar la creación de un índice, la especificación de campos y la definición de cómo se deben tokenizar y analizar los datos.
3. **Implementar la funcionalidad de búsqueda**: Desarrolle el código backend para gestionar las solicitudes de búsqueda, como el envío de consultas al motor de búsqueda y el análisis sintáctico de las respuestas. Además, asegúrese de gestionar las entradas del usuario, como palabras clave, frases y filtros.
4. **Mostrar los resultados de la búsqueda**: Diseña el frontend de tu aplicación para mostrar los resultados de la búsqueda de una manera fácil de usar, incluyendo paginación, ordenación y filtros.

Elasticsearch

Elastic search es un motor de búsqueda orientado a documentos. Es una base de datos basada en documentos que le permite INSERTAR, BORRAR, REGRESAR e incluso realizar análisis de los registros guardados. Pero, Elastic Search es diferente a cualquier otra base de datos de propósito general con la que haya trabajado, en el pasado. Es esencialmente un motor de búsqueda y ofrece un arsenal de características que puede utilizar para recuperar los datos almacenados en ella, según sus criterios de búsqueda. Y todo ello a la velocidad del rayo.

Visite los siguientes recursos para obtener más información:

- Sitio web de Elasticsearch
- Documentación de Elasticsearch

Solr

Solr es altamente fiable, escalable y tolerante a fallos, y ofrece indexación distribuida, replicación y consultas con equilibrio de carga, recuperación y conmutación por error automatizadas, configuración centralizada y mucho más. Solr impulsa las funciones de búsqueda y navegación de muchos de los sitios de Internet más grandes del mundo.

Visite los siguientes recursos para obtener más información:

- Sitio web oficial
- Documentación oficial

Intermediarios de mensajes

Los corredores de mensajes son una tecnología de comunicación entre aplicaciones para ayudar a construir un mecanismo de integración común para soportar arquitecturas de nube nativa, basadas en microservicios, sin servidor y de nube híbrida. Dos de los corredores de mensajes más famosos son `RabbitMQ` y `Apache Kafka`.

Visita los siguientes recursos para obtener más información:

- Introducción a los corredores de mensajes

RabbitMQ

Con decenas de miles de usuarios, RabbitMQ es uno de los corredores de mensajes de código abierto más populares. RabbitMQ es ligero y fácil de desplegar en las instalaciones y en la nube. Soporta múltiples protocolos de mensajería. RabbitMQ puede desplegarse en configuraciones distribuidas y federadas para satisfacer requisitos de gran escala y alta disponibilidad.

Visite los siguientes recursos para obtener más información:

- Tutoriales RabbitMQ
- Tutorial RabbitMQ - Colas de mensajes y sistemas distribuidos

Kafka

Apache Kafka es una plataforma de streaming de eventos distribuidos de código abierto utilizada por miles de empresas para canalizaciones de datos de alto rendimiento, análisis de streaming, integración de datos y aplicaciones de misión crítica.

Visite los siguientes recursos para obtener más información:

- Introducción rápida a Apache Kafka
- Fundamentos de Apache Kafka

Contenedores frente a virtualización

Los contenedores y las máquinas virtuales son los dos enfoques más populares para configurar una infraestructura de software para su organización.

Visite los siguientes recursos para obtener más información:

- Contenedores frente a virtualización: Todo lo que necesita saber
- Contenirezación o virtualización - Las diferencias

Kubernetes

Kubernetes es una plataforma de gestión de contenedores de **código abierto**, y el producto dominante en este espacio. Con Kubernetes, los equipos pueden desplegar imágenes en varios hosts subyacentes, definiendo la disponibilidad deseada, la lógica de despliegue y la lógica de escalado en YAML. Kubernetes evolucionó a partir de Borg, una plataforma interna de Google utilizada para aprovisionar y asignar recursos informáticos. (similar a los sistemas Autopilot y Aquaman de Microsoft Azure).

La popularidad de Kubernetes la ha convertido en una habilidad cada vez más importante para el Ingeniero DevOps y ha desencadenado la creación de equipos de Plataforma en toda la industria. Estos equipos de ingeniería de plataformas a menudo existen con el único propósito de hacer que Kubernetes sea accesible y utilizable para sus colegas de desarrollo de productos.

Visite los siguientes recursos para obtener más información:

- Sitio web de Kubernetes
- Documentación de Kubernetes
- Curso acelerado de Kubernetes para principiantes absolutos
- Introducción: Cómo surgió Kubernetes, qué es y por qué debería importarle
- Kubernetes: Una visión general

Docker

Docker es una plataforma para trabajar con aplicaciones en contenedores. Entre sus características se encuentran un demonio y un cliente para gestionar e interactuar con contenedores, registros para almacenar imágenes y una aplicación de escritorio para empaquetar todas estas características.

Visite los siguientes recursos para obtener más información:

- Documentación de Docker
- Qué es Docker | AWS
- Curso completo de Docker - ¡De PRINCIPIANTE a PROFESIONAL!
- Tutorial de Docker
- Docker simplificado en 55 segundos

LXC

LXC es una abreviatura utilizada para Linux Containers que es un sistema operativo que se utiliza para ejecutar múltiples sistemas Linux virtualmente en un host controlado a través de un único

kernel Linux. LXC es una interfaz de espacio de usuario para las funciones de contención del núcleo Linux. Mediante una potente API y herramientas sencillas, permite a los usuarios de Linux crear y gestionar fácilmente contenedores de sistemas o aplicaciones.

Visite los siguientes recursos para obtener más información:

- Documentación de LXC
- ¿Qué es LXC?
- Introducción a los contenedores Linux (LXC)
- Introducción a los contenedores LXC

GraphQL

GraphQL es un lenguaje de consulta para API y un tiempo de ejecución para realizar esas consultas con los datos existentes. GraphQL proporciona una descripción completa y comprensible de los datos de su API, ofrece a los clientes la posibilidad de pedir exactamente lo que necesitan y nada más, facilita la evolución de las API a lo largo del tiempo y habilita potentes herramientas para desarrolladores.

Visite los siguientes recursos para obtener más información:

- Introducción a GraphQL
- Tutorial Fullstack para GraphQL
- Tutoriales de GraphQL
- Curso de GraphQL para principiantes

Apollo

Apollo es una plataforma para construir un gráfico unificado, una capa de comunicación que te ayuda a gestionar el flujo de datos entre los clientes de tu aplicación (como aplicaciones web y nativas) y tus servicios back-end.

Visite los siguientes recursos para obtener más información:

- Sitio web de Apollo
- Documentación oficial
- Canal oficial de YouTube
- Tutorial de GraphQL con React - Cliente Apollo

Relé Moderno

Relay es un cliente JavaScript utilizado en el navegador para obtener datos GraphQL. Es un framework JavaScript desarrollado por Facebook para gestionar y obtener datos en aplicaciones React. Está construido pensando en la escalabilidad para alimentar aplicaciones complejas como Facebook. El objetivo final de GraphQL y Relay es ofrecer interacciones instantáneas de interfaz de usuario-respuesta.

Visita los siguientes recursos para obtener más información:

- Sitio web oficial
- Introducción a Relay modern

Web sockets

Los Web sockets se definen como una comunicación bidireccional entre los servidores y los clientes, lo que significa que ambas partes se comunican e intercambian datos al mismo tiempo. Este protocolo define una comunicación full duplex desde la base. Los web sockets suponen un paso adelante en la incorporación de funcionalidades de escritorio a los navegadores web.

Visite los siguientes recursos para obtener más información:

- Introducción a WebSockets
- Guía para principiantes sobre WebSockets
- Biblioteca Socket.io Comunicación bidireccional y de baja latencia para todas las plataformas

Eventos enviados por el servidor

Server-Sent Events (SSE) es una tecnología que permite a un servidor web enviar datos a un cliente en tiempo real. Utiliza una conexión HTTP para enviar un flujo de datos del servidor al cliente, y el cliente puede escuchar estos eventos y actuar cuando los recibe.

SSE es útil para aplicaciones que requieren actualizaciones en tiempo real, como sistemas de chat, teletipos de bolsa y feeds de redes sociales. Es una forma sencilla y eficaz de establecer una conexión duradera entre un cliente y un servidor, y es compatible con la mayoría de los navegadores web modernos.

Para utilizar SSE, el cliente debe crear un objeto EventSource y especificar la URL del script del servidor que enviará los eventos. A continuación, el servidor puede enviar los eventos escribiéndolos en el flujo de respuesta con el formato adecuado.

Visite los siguientes recursos para obtener más información:

- Eventos Enviados por el Servidor – MDN

Servidores web

Los servidores web pueden ser de hardware o de software, o quizás una combinación de ambos.

Lado hardware:

Un servidor web de hardware es un ordenador que aloja software de servidor web y los archivos que componen un sitio web (por ejemplo, documentos HTML, imágenes, hojas de estilo CSS y archivos JavaScript). Un servidor web establece una conexión a Internet y facilita el intercambio físico de datos con otros dispositivos conectados a la web.

Lado software:

Un servidor web de software tiene una serie de componentes de software que regulan la forma en que los usuarios en línea acceden a los archivos alojados. Se trata, como mínimo, de un servidor HTTP. El software que conoce y entiende HTTP y las URL (direcciones web) se conoce como servidor HTTP (el protocolo que utiliza su navegador para ver páginas web). El contenido de estos sitios web alojados se envía al dispositivo del usuario final a través de un servidor HTTP, al que se puede acceder a través de los nombres de dominio de los sitios web que alberga.

Básicamente, un navegador realiza una petición HTTP cada vez que desea un archivo almacenado en un servidor web. El servidor web (hardware) correspondiente recibe la petición, que es aceptada por el servidor HTTP (software) apropiado, que localiza el contenido solicitado y lo devuelve al navegador a través de HTTP. (Si el servidor no puede localizar la página solicitada, responde con un error 404).

Visite los siguientes recursos para obtener más información:

- Qué es un servidor Web
- Conceptos y ejemplos de servidores web

Nginx

NGINX es un potente servidor web y utiliza una arquitectura sin hilos y basada en eventos que le permite superar a Apache si se configura correctamente. También puede hacer otras cosas importantes, como equilibrar la carga, almacenar HTTP en caché o utilizarse como proxy inverso.

Visite los siguientes recursos para obtener más información:

- Sitio web oficial
- NGINX explicado en 100 segundos

Apache

Apache es un servidor HTTP gratuito y de código abierto, disponible en muchos sistemas operativos, pero utilizado principalmente en distribuciones Linux. Es una de las opciones más populares para los desarrolladores web, ya que representa más del 30% de todos los sitios web, según estimaciones de W3Techs.

Visite los siguientes recursos para obtener más información:

- Sitio web del servidor Apache

- ¿Qué es el servidor web Apache?

Caddy

El servidor web Caddy es un servidor web extensible, multiplataforma y de código abierto escrito en Go. Tiene algunas características realmente buenas como SSL/HTTPs automático y un archivo de configuración realmente fácil.

Visita los siguientes recursos para saber más:

- Sitio web oficial
- Empezando con Caddy el Servidor Web HTTPS desde cero

MS IIS

Internet Information Services (IIS) para Windows® Server es un servidor Web flexible, seguro y manejable para alojar cualquier cosa en la Web.

Visite los siguientes recursos para obtener más información:

- Sitio Web oficial
- Aprenda Windows Web Server IIS

Construir a escala

En términos generales, la escalabilidad es la capacidad de un sistema para gestionar una cantidad creciente de trabajo añadiéndole recursos.

Un software concebido con una arquitectura escalable en mente, es un sistema que soportará mayores cargas de trabajo sin cambios fundamentales en él, pero no te engañes, esto no es magia. Sólo llegarás hasta cierto punto con un pensamiento inteligente sin añadirle más fuentes.

Para que un sistema sea escalable, hay ciertas cosas a las que debes prestar atención, como:

- Acoplamiento
- Observabilidad
- Evolucionabilidad
- Infraestructura

Cuando piensas en la infraestructura de un sistema escalable, tienes dos formas principales de construirla: utilizando recursos on-premise o aprovechando todas las herramientas que te puede dar un proveedor de cloud.

La principal diferencia entre los recursos locales y los de la nube será la FLEXIBILIDAD: con los proveedores de la nube no es necesario planificar con antelación, puede actualizar su infraestructura con un par de clics, mientras que con los recursos locales necesitará un cierto nivel de planificación.

Visite los siguientes recursos para obtener más información:

- Arquitectura escalable: Definición y guía práctica
- Escalado de sistemas distribuidos - Introducción a la arquitectura de software

Instrumentación, monitorización y telemetría

La instrumentación se refiere a la medición del rendimiento de un producto, con el fin de diagnosticar errores y escribir información de rastreo. La instrumentación puede ser de dos tipos: instrumentación de origen e instrumentación binaria.

La monitorización del backend permite al usuario ver el rendimiento de la infraestructura, es decir, los componentes que ejecutan una aplicación web. Entre ellos se incluyen el servidor HTTP, el middleware, la base de datos, los servicios API de terceros, etc.

La telemetría es el proceso de recopilación continua de datos de diferentes componentes de la aplicación. Estos datos ayudan a los equipos de ingeniería a solucionar problemas en los servicios y a identificar las causas. En otras palabras, los datos de telemetría potencian la observabilidad de sus aplicaciones distribuidas.

Visite los siguientes recursos para obtener más información:

- ¿Qué es la instrumentación?
- ¿Qué es la monitorización?
- ¿Qué es la telemetría?

Estrategias de mitigación

Esta sección se refiere principalmente a los patrones de diseño de la nube que ayudan a crear soluciones escalables. Eche un vistazo a los documentos sobre **patrones de diseño de la nube** de Microsoft y a este vídeo sobre los **patrones de estrangulamiento, reintento y disyuntor**.

Aprenda a realizar migraciones de bases de datos de forma eficaz. Especialmente las migraciones de esquemas multifase sin tiempo de inactividad. En lugar de realizar todos los cambios a la vez, realice cambios incrementales más pequeños para permitir que el código antiguo y el nuevo funcionen con la base de datos al mismo tiempo, antes de eliminar el código antiguo y, por último, eliminar las partes del esquema de la base de datos que ya no se utilizan.

Visite los siguientes recursos para obtener más información:

- Las bases de datos como reto para la entrega continua

Degradación gradual

La degradación gradual es un principio de diseño según el cual un sistema debe estar diseñado para seguir funcionando, aunque algunos de sus componentes o funciones no estén disponibles. En el contexto del desarrollo web, la degradación gradual se refiere a la capacidad de una página o aplicación web para seguir funcionando, incluso si el navegador o dispositivo del usuario no es compatible con ciertas características o tecnologías.

La degradación gradual se utiliza a menudo como alternativa a la mejora progresiva, un principio de diseño que establece que un sistema debe diseñarse para aprovechar las funciones y tecnologías avanzadas si están disponibles.

Visite los siguientes recursos para obtener más información:

- ¿Qué es la degradación gradual y por qué es importante?
- Cuatro consideraciones a la hora de diseñar sistemas para la degradación gradual
- El arte de la degradación gradual

Estrangulamiento

El estrangulamiento es un patrón de diseño que se utiliza para limitar la velocidad a la que se puede utilizar un sistema o componente. Se suele utilizar en entornos de computación en nube para evitar el uso excesivo de recursos, como la potencia de cálculo, el ancho de banda de red o la capacidad de almacenamiento.

Hay varias formas de implementar el estrangulamiento en un entorno de nube:

- Limitación de velocidad: Se trata de establecer un número máximo de peticiones que se pueden realizar a un sistema o componente en un periodo de tiempo determinado.
- Asignación de recursos: Esto implica asignar una cantidad fija de recursos a un sistema o componente, y luego limitar el uso de esos recursos si se exceden.
- Cubo de fichas: Consiste en utilizar un "cubo" de fichas para representar los recursos disponibles y permitir que cada solicitud "consuma" un determinado número de fichas. Cuando el cubo está vacío, se deniegan las solicitudes adicionales hasta que haya más tokens disponibles.

El estrangulamiento es un aspecto importante del diseño de la nube, ya que ayuda a garantizar que los recursos se utilizan de forma eficiente y que el sistema permanece estable y disponible. A menudo se utiliza junto con otros patrones de diseño, como el autoescalado y el equilibrio de carga, para proporcionar un entorno de nube escalable y resistente.

Visite los siguientes recursos para obtener más información:

- Estrangulamiento - Framework bien diseñado de AWS

Contrapresión

La contrapresión es un patrón de diseño que se utiliza para gestionar el flujo de datos a través de un sistema, especialmente en situaciones en las que la tasa de producción de datos supera la tasa de consumo de datos. Se utiliza habitualmente en entornos de computación en nube para evitar la sobrecarga de recursos y garantizar que los datos se procesan de forma oportuna y eficiente.

Hay varias formas de aplicar la contrapresión en un entorno de nube:

- Almacenamiento en búfer: Consiste en almacenar los datos entrantes en un búfer hasta que puedan procesarse, lo que permite al sistema seguir recibiendo datos, aunque temporalmente no pueda procesarlos.
- Procesamiento por lotes: consiste en agrupar los datos entrantes en lotes y procesar los lotes en secuencia, en lugar de procesar cada dato individualmente.
- Control de flujo: Consiste en utilizar mecanismos como señales de control de flujo o ventanas para regular la velocidad de transmisión de datos entre sistemas.

La contrapresión es un aspecto importante del diseño de la nube, ya que ayuda a garantizar que los datos se procesan de forma eficiente y que el sistema permanece estable y disponible. A menudo se utiliza junto con otros patrones de diseño, como el autoescalado y el equilibrio de carga, para proporcionar un entorno de nube escalable y resistente.

Visite los siguientes recursos para obtener más información:

- Backpressure - Framework bien diseñado de AWS

Cambio de carga

El cambio de carga es un patrón de diseño que se utiliza para gestionar la carga de trabajo de un sistema cambiando la carga a diferentes componentes o recursos en diferentes momentos. Se utiliza habitualmente en entornos de computación en nube para equilibrar la carga de trabajo de un sistema y optimizar el uso de los recursos.

Hay varias formas de implementar el cambio de carga en un entorno de nube:

- Programación: Se trata de programar la ejecución de tareas o cargas de trabajo para que se produzcan en momentos o intervalos específicos.
- Equilibrio de carga: Esto implica la distribución de la carga de trabajo de un sistema a través de múltiples recursos, como servidores o contenedores, para garantizar que la carga de trabajo esté equilibrada y que los recursos se utilicen de manera eficiente.
- Autoescalado: Consiste en ajustar automáticamente el número de recursos disponibles para un sistema en función de la carga de trabajo, permitiendo que el sistema aumente o disminuya según sea necesario.

El cambio de carga es un aspecto importante del diseño de la nube, ya que ayuda a garantizar que los recursos se utilizan de manera eficiente y que el sistema permanece estable y disponible. A menudo se utiliza junto con otros patrones de diseño, como el estrangulamiento y la contrapresión, para proporcionar un entorno de nube escalable y resistente.

Visite los siguientes recursos para obtener más información:

- Load Shifting - AWS Well-Architected Framework (en inglés)

Disyuntor

El patrón de diseño del disyuntor es una forma de proteger un sistema de fallos o carga excesiva deteniendo temporalmente determinadas operaciones si se considera que el sistema se encuentra en un estado de fallo o sobrecarga. Se utiliza habitualmente en entornos de computación en nube para evitar fallos en cascada y mejorar la resistencia y disponibilidad de un sistema.

Un disyuntor consta de tres estados: cerrado, abierto y semiabierto. En el estado cerrado, el disyuntor permite que las operaciones se desarrollen con normalidad. Si el sistema sufre un fallo o se sobrecarga, el disyuntor pasa al estado abierto y todas las operaciones posteriores se detienen inmediatamente. Después de un periodo de tiempo especificado, el disyuntor pasa al estado semiabierto, y se permite que un pequeño número de operaciones continúen. Si estas operaciones tienen éxito, el interruptor vuelve al estado cerrado; si fallan, el interruptor vuelve al estado abierto.

El patrón de diseño del disyuntor es útil para proteger un sistema de fallos o de una carga excesiva, proporcionando una forma de detener temporalmente ciertas operaciones y permitir que el sistema se recupere. A menudo se utiliza junto con otros patrones de diseño, como reintentos y fallbacks, para proporcionar un entorno de nube más robusto y resistente.

Visite los siguientes recursos para obtener más información:

- Disyuntor - Framework bien diseñado de AWS

Escalado horizontal/vertical

El escalado horizontal es un cambio en el número de un recurso. Por ejemplo, aumentar el número de máquinas virtuales que procesan mensajes en una cola. El escalado vertical es un cambio en el tamaño/potencia de un recurso. Por ejemplo, aumentar la memoria o el espacio en disco disponible para una máquina. El escalado puede aplicarse a bases de datos, recursos en la nube y otras áreas de la informática.

Visite los siguientes recursos para obtener más información:

- Escalado horizontal frente a escalado vertical
- Escalado en bases de datos
- Conceptos básicos de diseño de sistemas: Escalado horizontal frente a escalado vertical
- Diseño de sistemas 101

Observabilidad

Registro de métricas y otros elementos observables que pueden ayudar a depurar y resolver problemas cuando las cosas van mal.

En el desarrollo de software, la observabilidad es la medida de lo bien que podemos entender un sistema a partir del trabajo que hace, y cómo mejorarlo.

¿Qué hace que un sistema sea "observable"? Es su capacidad de producir y recopilar métricas, registros y trazas para que podamos entender lo que ocurre bajo el capó e identificar problemas y cuellos de botella más rápidamente.

Por supuesto, puedes implementar todas estas funciones por ti mismo, pero hay muchos programas que pueden ayudarte, como Datadog, Sentry y CloudWatch.

Visite los siguientes recursos para obtener más información:

- Documentación de DataDog
- Documentación de AWS CloudWatch
- Documentación de Sentry
- AWS re:Invent 2017: Mejora de la observabilidad de microservicios y sin servidor con Monitor
- Observabilidad e instrumentación: Qué son y por qué son importantes

Este roadmap.sh ha sido traducido por rortegag.com

Los enlaces del documento son de roadmap.sh y no están traducidos