Resultados de Investigación: Arquitectura Cliente-Servidor, Next.js, protocolo SOAP y Spring Boot

Luisa Lorena Parra Nivia
Fabio Luis Buitrago Ochoa
Erick Santiago Garavito Villamil

Arquitectura de Software

Marzo, 2024

Tabla de contenido

Introducción

El propósito de este documento es realizar una exploración y comprensión de una serie de conceptos, patrones, estilos, principios, tecnologías y frameworks relevantes en el ámbito de la informática y el desarrollo de software. En este contexto, se llevará a cabo una investigación detallada sobre el patrón arquitectónico cliente-servidor, Next.js/TypeScript, SOAP, Spring Boot/Java y MongoDB.

Para alcanzar este objetivo, hemos realizado una investigación sobre la historia y la evolución de los temas mencionados, con el fin de comprender su desarrollo a lo largo del tiempo. Esto nos ha permitido identificar cómo han surgido inicialmente, cómo han evolucionado en respuesta a las necesidades cambiantes del mercado y cómo se han utilizado en diversas aplicaciones y contextos.

Además, hemos profundizado en la posición de estas tecnologías en el mercado actual, tanto en términos de su popularidad y adopción según los rankings de uso, como en relación con los rangos salariales asociados a los roles donde se implementan. Esto nos ha proporcionado una comprensión más completa de su relevancia y demanda en la industria tecnológica actual.

Para enriquecer nuestra comprensión, exploraremos los principios SOLID asociados a cada tema, destacando cómo estos principios fundamentales de diseño de software se aplican en la práctica. También analizaremos los atributos de calidad pertinentes, como la escalabilidad, la mantenibilidad, la seguridad y el rendimiento, para entender cómo estos conceptos se traducen en características deseables en los sistemas de software.

Por último, para ilustrar la aplicación efectiva de los conceptos, patrones, tecnologías y frameworks mencionados en situaciones concretas, presentaremos un caso de estudio práctico que integra todas estas herramientas en un proyecto de desarrollo de software.

1. Definición, historia y evolución

Arquitectura cliente servidor

Historia

La arquitectura cliente-servidor ha tenido un papel fundamental en la evolución de los sistemas informáticos, desde sus primeras etapas hasta la llegada de la computación en la nube. Esta arquitectura ha experimentado un desarrollo significativo a lo largo del tiempo, adaptándose a las necesidades cambiantes de la industria y las tecnologías emergentes.

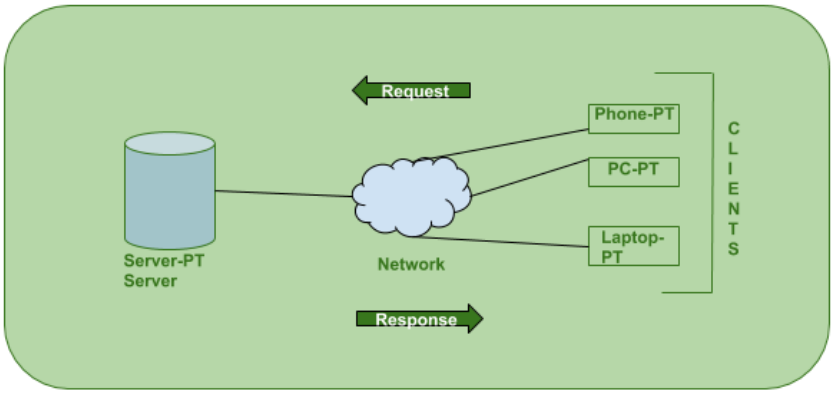
La historia de la arquitectura cliente-servidor se remonta a principios de la década de 1980 en los Estados Unidos, cuando la informática comenzó a transitar de los grandes mainframes hacia un modelo de procesamiento distribuido utilizando múltiples estaciones de trabajo o computadoras personales. Las corporaciones adoptaron rápidamente los sistemas cliente-servidor, los cuales se convirtieron en la columna vertebral de su infraestructura de automatización de oficinas y comunicación.

Uno de los hitos en la evolución de la arquitectura cliente-servidor fue el desarrollo del sistema operativo UNIX por Ken Thompson y Dennis Ritchie en 1969 en Bell Labs. UNIX se convirtió rápidamente en un sistema operativo ampliamente utilizado en minicomputadoras. Si bien las estaciones de trabajo que ejecutaban sistemas operativos UNIX existían desde principios de la década de 1980, fue en la segunda mitad de la década de 1980 cuando las corporaciones japonesas comenzaron a desarrollar y comercializar estaciones de trabajo UNIX. Sin embargo, el crecimiento del mercado de estaciones de trabajo se desaceleró a principios de la década de 1990 debido al aumento de la potencia y el rendimiento de las computadoras personales, y para mediados de la década de 1990, el mercado de estaciones de trabajo estaba en declive. A medida que la demanda de estaciones de trabajo disminuía, algunos de los modelos de gama alta comenzaron a utilizarse como servidores, lo que eventualmente se convirtió en la aplicación principal de las estaciones de trabajo UNIX.

Dado que los servidores estaban dirigidos principalmente a aplicaciones empresariales que involucraban conexiones con múltiples clientes, no solo tenían que ofrecer potencia de procesamiento, sino también estabilidad, tolerancia a fallos y escalabilidad comparable a la de los mainframes. Los servidores se construyeron para admitir actualizaciones a arquitecturas redundantes o incluso cuádruples redundantes y utilizaron múltiples discos duros en matrices RAID. Los sistemas tolerantes a fallos también se organizaron con sistemas operativos de funcionamiento continuo para aplicaciones que requerían operación ininterrumpida.

Definición

El modelo cliente-servidor es una arquitectura de software que divide las aplicaciones en dos partes distintas: el cliente, que solicita recursos o servicios, y el servidor, que proporciona esos recursos o servicios. Estas partes interactúan entre sí a través de una red, utilizando un protocolo de comunicación estándar, como HTTP, TCP/IP o UDP.



Tomado de: https://www.geeksforgeeks.org/client-server-model/

- **Cliente**

El cliente, en el contexto del modelo cliente-servidor, se define como una aplicación o dispositivo que busca acceder a servicios o recursos proporcionados por un servidor. Esta aplicación puede adoptar diversas formas, como una aplicación de escritorio instalada en una computadora, una aplicación web accesible a través de un navegador, una aplicación móvil instalada en un dispositivo móvil, o incluso otros tipos de software especializado.

Independientemente de su forma, el cliente inicia la interacción comunicándose con el servidor mediante el envío de peticiones. Estas solicitudes pueden variar en función de las necesidades específicas de la aplicación, como recuperar información, almacenar datos, procesar transacciones, entre otras posibles funciones.

El cliente puede ofrecer una interfaz gráfica de usuario (GUI) o una interfaz de línea de comandos (CLI), dependiendo de cómo esté diseñado y de las preferencias del usuario. La interfaz de usuario permite a los usuarios interactuar con la aplicación cliente de manera intuitiva, facilitando la navegación, la entrada de datos y la visualización de resultados. La elección entre una GUI y una CLI dependerá del contexto de uso y de los requisitos de la aplicación en cuestión.

- **Servidor**

El servidor, dentro del modelo cliente-servidor, se define como una aplicación o sistema encargado de responder a las solicitudes realizadas por los clientes. Este servidor puede manifestarse en forma de un hardware físico dedicado o bien como una instancia virtual alojada en la nube, dependiendo de las necesidades y la infraestructura disponible. Sea cual sea su forma, el servidor ejecuta un software especializado diseñado para gestionar las peticiones entrantes de los clientes de manera eficiente.

El servidor permanece en un estado de espera activa, listo para recibir y procesar las solicitudes entrantes de los clientes. Una vez que recibe una solicitud, el servidor la analiza y ejecuta las acciones correspondientes según lo especificado por la solicitud. Estas acciones pueden incluir recuperar datos, procesar transacciones, realizar cálculos o cualquier otra operación requerida por el cliente.

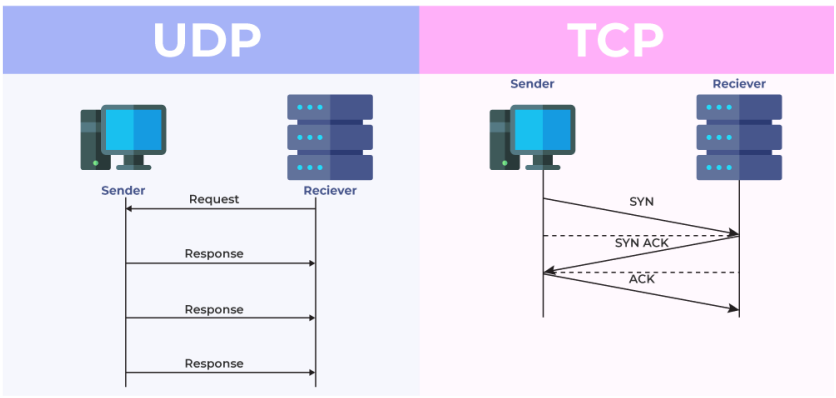
Una característica fundamental del servidor es su capacidad para gestionar múltiples conexiones de clientes de forma simultánea. Esto significa que puede atender a varios clientes al mismo tiempo, sin comprometer la eficiencia o la calidad del servicio proporcionado. Esta capacidad es crucial en entornos donde se esperan grandes volúmenes de tráfico o donde múltiples usuarios acceden al sistema simultáneamente.

- **Comunicación**

La comunicación entre el cliente y el servidor se establece a través de una red, que puede ser tanto internet como una red local. Esta red sirve como el medio a través del cual se transmiten los datos entre el cliente y el servidor, permitiendo así la interacción entre ambas partes.

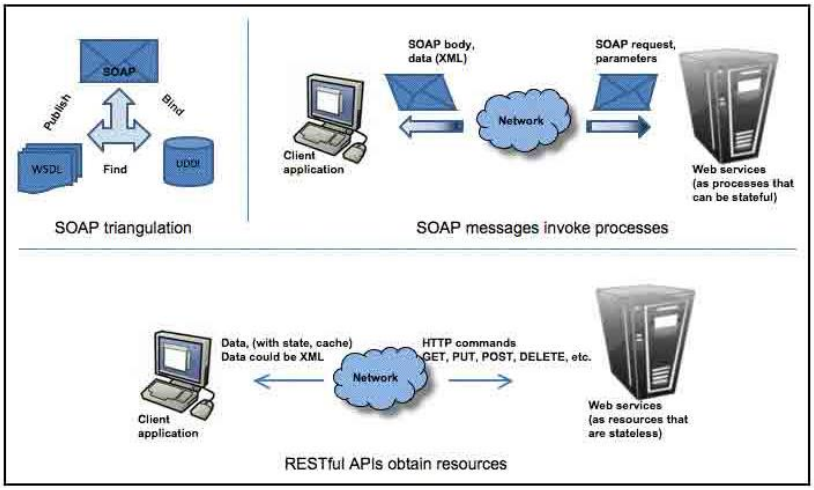
Para facilitar esta comunicación de manera eficiente y segura, se emplea un protocolo de comunicación estándar. Este protocolo define las reglas y el formato de los datos que se intercambian entre el cliente y el servidor, asegurando una comunicación coherente y comprensible para ambas partes.

Existen varios protocolos comunes utilizados para diferentes propósitos. TCP/IP (Transmission Control Protocol/Internet Protocol) es otro protocolo comúnmente utilizado para comunicaciones confiables, garantizando la entrega ordenada y sin errores de los datos entre el cliente y el servidor. Por otro lado, UDP (User Datagram Protocol) se utiliza para comunicaciones no confiables donde la velocidad y la simplicidad son más importantes que la integridad de los datos, como en la transmisión de video en tiempo real o en videojuegos en línea.



Tomado de: https://www.geeksforgeeks.org/differences-between-tcp-and-udp/

Por ejemplo, HTTP (Hypertext Transfer Protocol) es ampliamente utilizado para la comunicación en la web, permitiendo la transferencia de recursos como páginas web, imágenes y otros archivos entre el cliente y el servidor. Dentro del contexto de HTTP (Hypertext Transfer Protocol), se encuentran dos enfoques principales para la comunicación entre cliente y servidor: SOAP (Simple Object Access Protocol) y REST (Representational State Transfer). Ambos se basan en HTTP como el protocolo subyacente para la comunicación, pero difieren en su enfoque y estructura.



Tomado de: https://www.programacion.com.py/web/web-services-rest-vs-soap

- **Modelo de Interacción**

La interacción entre el cliente y el servidor sigue un proceso definido en el cual el cliente envía una solicitud al servidor, detallando el tipo de servicio o recurso requerido. Esta solicitud puede incluir información específica sobre la acción deseada o los datos necesarios para llevar a cabo la operación solicitada.

Una vez que el servidor recibe la solicitud, procede a procesarla utilizando el software especializado que se ejecuta en él. Durante este proceso, el servidor puede realizar

diversas acciones, como acceder a bases de datos, ejecutar algoritmos o interactuar con otros sistemas. Posteriormente, el servidor genera una respuesta que se envía de vuelta al cliente.

Esta respuesta puede contener los datos solicitados por el cliente o simplemente indicar el resultado de la operación realizada en el servidor. En algunos casos, la respuesta puede incluir información adicional, como códigos de estado para indicar si la operación fue exitosa o si se produjo algún error.

Es importante destacar que la interacción entre el cliente y el servidor puede ser síncrona o asíncrona, dependiendo de los requisitos específicos de la aplicación. En una interacción síncrona, el cliente espera activamente la respuesta del servidor después de enviar la solicitud, lo que significa que el proceso de la aplicación cliente se detiene temporalmente hasta que se recibe la respuesta del servidor. Por otro lado, en una interacción asíncrona, el cliente puede continuar con otras tareas mientras espera la respuesta del servidor, lo que permite una mayor flexibilidad y capacidad de respuesta en la aplicación.

Next.js

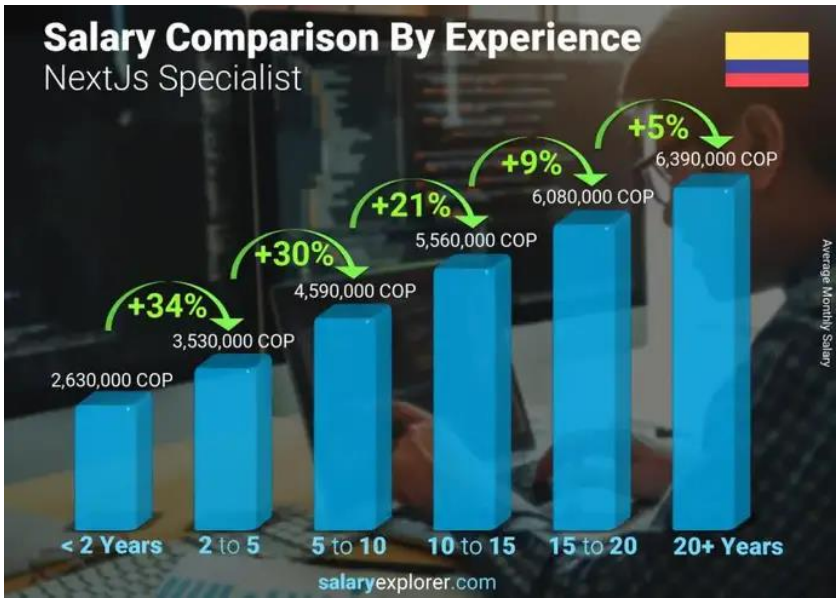
Desde su lanzamiento en 2016, Next.js ha ganado popularidad debido a su integración con React, una de las tecnologías más populares en JavaScript. Aunque no se destacó en encuestas como la Stack Overflow Survey hasta 2022 y en el Jetbrains Developer Ecosystem Language hasta 2023, gradualmente ha ganado impulso en el desarrollo web. Según el Jetbrains Developer Ecosystem Language, Next.js ocupó el tercer lugar con un 27% de audiencia entre las tecnologías y frameworks más utilizados, después de React con un 57% y Vue.js con un 32%.

Información salarial

Ahora bien, si entramos al ámbito laboral, se tienen buenas oportunidades, según SalaryExplorer en Colombia hay un salario promedio de $4.460.000 COP.

Tomado de https://www.salaryexplorer.com/es/average-salary-wage-comparison-colombia-nextjs-specialist-c47j19491?expand_article=1

Adicional a esto se encuentra información acerca de los salarios de una persona experta en Next.Js según la experiencia que se tenga, con un salario base de $2.630.000 SOP mensuales para personas con menos de 2 años de experiencia.



Tomado de https://www.salaryexplorer.com/es/average-salary-wage-comparison-colombia-nextjs-specialist-c47j19491?expand_article=1

Y lastimosamente los estudios demuestran que en Colombia los hombres suelen ganar 10% más que las mujeres en promedio según Salary explorer. Ahora, en un ámbito más internacional tenemos los salarios promedio asociados a el nivel de experiencia con datos de Estados Unidos junto con las 10 ciudades mejor pagas.

Experience Level	Average Annual Salary Range
Mid-Level Developer	$110,000 - $142,000
Senior Developer	$140,000 - $175,000
Junior Developer	$80,000 - $115,000

Tomado de https://clouddevs.com/next/annual-salaries/

City	Annual Salary	Monthly Pay	Weekly Pay	Hourly Wage
San Francisco, CA	$167,385	$13,948	$3,218	$80.47
Fremont, CA	$161,433	$13,452	$3,104	$77.61
San Jose, CA	$160,755	$13,396	$3,091	$77.29
Oakland, CA	$158,579	$13,214	$3,049	$76.24
Antioch, CA	$155,307	$12,942	$2,986	$74.67
Lebanon, NH	$154,327	$12,860	$2,967	$74.20
Hayward, CA	$153,562	$12,796	$2,953	$73.83
Seattle, WA	$153,333	$12,777	$2,948	$73.72
Vallejo, CA	$153,249	$12,770	$2,947	$73.68
Staten Island, NY	$152,472	$12,706	$2,932	$73.30

Tomado de https://clouddevs.com/next/annual-salaries/

Historia

Next.js es un framework de código abierto con licencia MIT, lanzado el 25 de octubre de 2016 por la compañía ZEIT, que más tarde se convirtió en Vercel en 2020. Este framework fue concebido para abordar las necesidades de creación de aplicaciones web universales en JavaScript. Basándose en tecnologías establecidas como React, Webpack y Babel, Next.js busca simplificar el desarrollo de aplicaciones web, permitiendo el renderizado tanto en el servidor como en el cliente. Su objetivo principal era superar las limitaciones de las tecnologías existentes en el ámbito de las aplicaciones web en JavaScript.

El framework ha evolucionado continuamente desde su lanzamiento en 2016, mejorando tanto la experiencia de usuario (UX) como la de desarrollo (DX). Se han implementado mejoras significativas en el rendimiento y la funcionalidad, como se demostró en la versión Next.js 11, lanzada en junio de 2021. En esta versión, se logró una mejora del 24% en el tiempo de inicio, desde que se crea una nueva sesión de desarrollo hasta que se visualiza el resultado en pantalla. Adicionalmente, el tiempo de procesamiento de cambios también mejoró en un 40%, desde que se guarda un cambio en el archivo hasta que se visualiza en la pantalla.

Definición

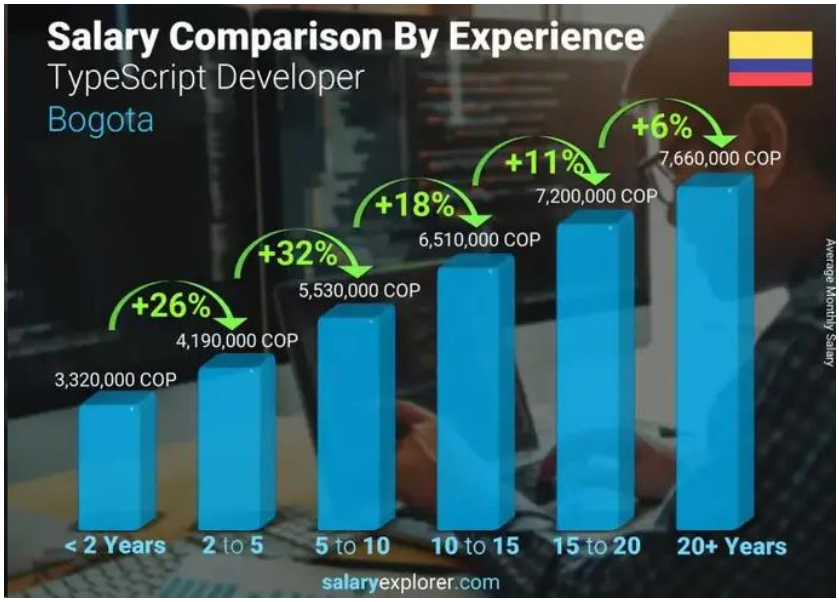
Next.js es un framework de código abierto con licencia MIT diseñado para crear aplicaciones web universales en JavaScript. Basado en tecnologías como React, Webpack y Babel, Next.js permite el renderizado tanto en el servidor como en el cliente, simplificando el desarrollo de aplicaciones web y abordando las limitaciones de las tecnologías existentes en este ámbito. Su integración con React ha contribuido a su popularidad entre los desarrolladores, ofreciendo mejoras continuas en rendimiento y funcionalidad desde su lanzamiento.

TypeScript

La popularidad de TypeScript se refleja en las encuestas; fue votado como el segundo lenguaje más querido en la Encuesta de Stack Overflow y utilizado por el 78% de los encuestados en el Estado de JS 2020, con un 93% expresando su intención de volver a usarlo. Además, TypeScript ha recibido reconocimiento como la tecnología más adoptada en términos de crecimiento año tras año. Su presencia se observa en frameworks populares como Angular, Next.js y AdonisJs, entre otros. Aunque JavaScript sigue siendo el lenguaje más utilizado debido a su arraigo histórico, TypeScript ha mantenido una posición significativa en las encuestas. En la encuesta de Stack Overflow de 2023, por ejemplo, TypeScript fue el cuarto lenguaje más utilizado, superado por SQL, Python y HTML/CSS, pero manteniendo un sólido 38.78% de uso. Esto evidencia su continua relevancia y adopción en el campo del desarrollo de software.

Información salarial

En cuanto a los salarios para desarrollados en TypeScript se tiene un salario base de $3.320.000COP en Colombia, a continuación, se muestra la gráfica de salarios dada la experiencia laborar de la persona en Colombia.



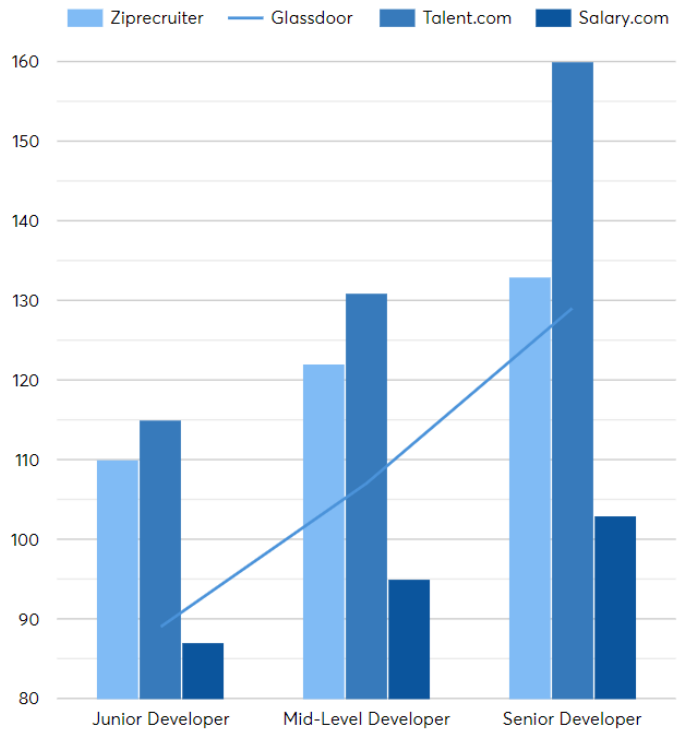
Tomado de https://www.salaryexplorer.com/average-salary-wage-comparison-bogota-typescript-developer-t397j19765?expand_article=1

En el ámbito internacional se cuentan con salarios anuales promedio por región, siendo norte américa la región con mejor promedio salarial

Region	Average Annual Salary (USD)
North America	$100,000 - $135,000
South America	$60,000 - $85,000
Western Europe	$80,000 - $110,000
Eastern Europe	$55,000 - $75,000
Australia	$95,000 - $125,000
Asia	$50,000 - $65,000
Africa	$40,000 - $55,000

Tomado de https://clouddevs.com/typescript/annual-salaries/

Adicional a esto podemos ver el promedio salarial según la experiencia según buscadores salariales reconocidos como Ziprecuiter, glassdor, talent.com y salary.com y las ciudades con mejor salario de Estados Unidos.



Tomado de https://clouddevs.com/typescript/annual-salaries/

City	Annual Salary	Monthly Pay	Weekly Pay	Hourly Wage
Berkeley, CA	$156,597	$13,049	$3,011	$75.29
New York City, NY	$150,575	$12,547	$2,895	$72.39
Renton, WA	$150,124	$12,510	$2,887	$72.18
Santa Monica, CA	$149,136	$12,428	$2,868	$71.70
Bend, OR	$148,954	$12,412	$2,864	$71.61
Daly City, CA	$148,197	$12,349	$2,849	$71.25
Boston, MA	$144,996	$12,083	$2,788	$69.71
Wausau, WI	$144,395	$12,032	$2,776	$69.42
San Mateo, CA	$144,323	$12,026	$2,775	$69.39
Richmond, CA	$143,934	$11,994	$2,767	$69.20

Tomado de https://www.ziprecruiter.com/Salaries/Typescript-Developer-Salary

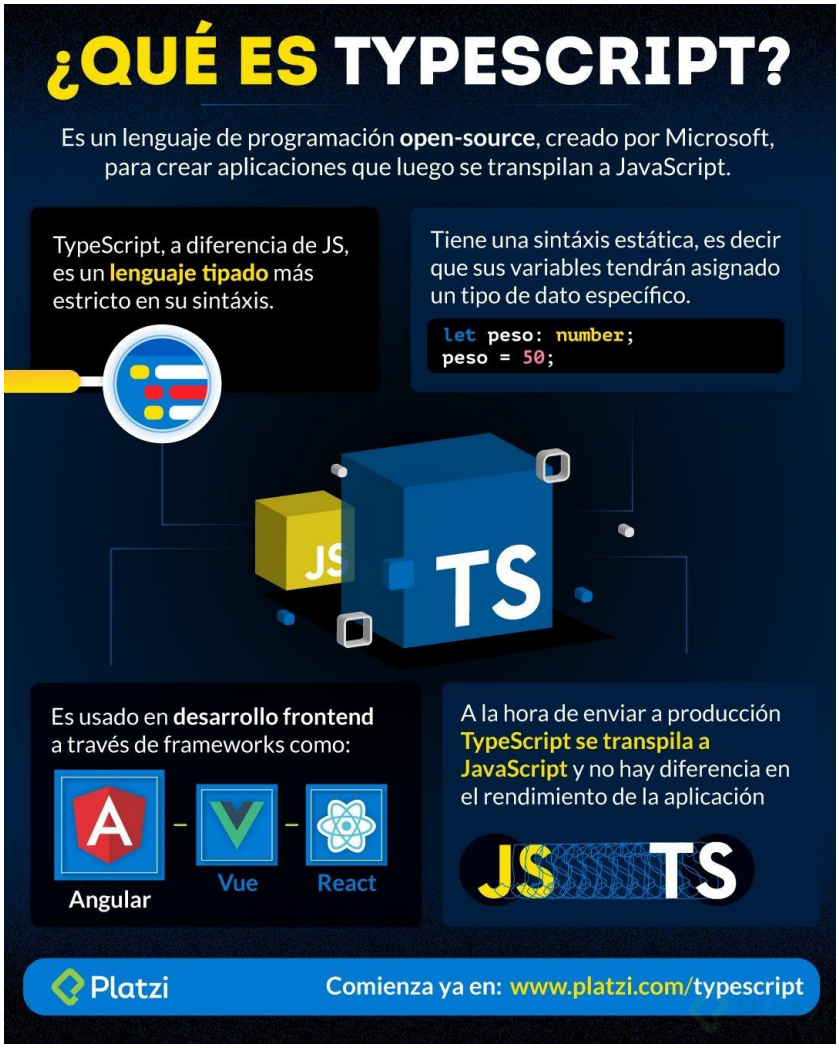
Historia

Su lanzamiento inicial fue en octubre de 2012. Aunque inicialmente la comunidad de desarrolladores mostró escepticismo, las características propuestas demostraron su valía con el tiempo. Conforme TypeScript evolucionaba, se consolidaba como una herramienta poderosa para el desarrollo de aplicaciones web complejas. Aunque inicialmente recibió críticas, la comunidad de TypeScript fue creciendo gradualmente, impulsada por la participación de desarrolladores dispuestos a experimentar y contribuir al proyecto.

La visión original del equipo detrás de TypeScript se ha mantenido constante a lo largo de los años, centrada en proporcionar herramientas avanzadas para mejorar la experiencia de desarrollo en JavaScript. A medida que TypeScript ganaba popularidad, se posicionaba como una herramienta esencial en el ecosistema de desarrollo de software, siendo ampliamente utilizada por millones de desarrolladores en todo el mundo. Con una mirada hacia el futuro, TypeScript sigue evolucionando y contribuyendo al panorama tecnológico actual.

Definición

TypeScript es un lenguaje de programación de alto nivel, de código abierto y desarrollado por Microsoft, bajo la licencia Apache 2.0. Se basa en JavaScript y propone ser un lenguaje compilado, fuertemente tipado e implementando conceptos de interfaces y orientación a objetos. Surgió como una respuesta a las deficiencias percibidas en JavaScript, ofreciendo verificación de tipos, detección de errores y funciones de edición avanzadas.



Tomado de: https://platzi.com/clases/1580-typescript-angular/20216-resumen-de-typescript/

SOAP

A pesar de los desafíos y críticas que ha enfrentado a lo largo de su historia, SOAP ha demostrado su durabilidad y utilidad en numerosos escenarios de aplicación. De hecho, según el State of the API Report 2023, SOAP sigue siendo uno de los protocolos de comunicación más utilizados, ocupando el cuarto lugar con un 23% de adopción. Este dato refleja la continua relevancia y preferencia por parte de los desarrolladores y empresas que confían en SOAP para la implementación de sus servicios web y sistemas distribuidos.

Historia

SOAP, el Protocolo de Acceso a Objetos Simples (Simple Object Access Protocol en inglés), surgió en 1998 como una iniciativa liderada por Dave Winer, Don Box, Bob Atkinson y Mohsen Al-Ghosein, quienes en ese momento trabajaban en Microsoft. Este protocolo fue concebido como una solución para el intercambio de información en entornos distribuidos y descentralizados. Los mensajes SOAP sirven como medio para transmitir datos desde el remitente hasta el destinatario, y pueden combinarse para formar patrones de solicitud y respuesta.

SOAP se basa en XML, lo que lo hace altamente interoperable y compatible con una amplia gama de plataformas y tecnologías. Surgió en un momento en el que no existía un lenguaje de esquema ni un sistema de tipos para XML, siendo XML 1.0 una Recomendación reciente en ese momento. En sus primeras etapas, SOAP se centró en definir un sistema de tipos que permitiera la representación de datos de manera estructurada, así como la definición de operaciones o métodos para su manipulación.

Una de las razones principales que motivaron la creación de SOAP fue la necesidad de superar las limitaciones de los sistemas existentes en ese momento, que se basaban en tecnologías como COM o eran similares a EDI, lo cual resultaba poco adecuado para una comunidad de desarrollo en constante evolución. Por lo tanto, SOAP se propuso alcanzar un punto óptimo que satisficiera las necesidades del 80% de los casos comunes, pero que también pudiera adaptarse al 20% restante, brindando así una solución flexible y escalable.

Aunque SOAP fue concebido en 1998, su adopción y evolución se vieron afectadas por diversos factores, incluida la política interna de Microsoft en ese momento. Los esfuerzos iniciales para impulsar SOAP dentro de Microsoft se encontraron con obstáculos debido a desacuerdos entre grupos internos de la empresa, lo que retrasó su desarrollo y adopción en la industria.

Sin embargo, a medida que avanzaba hacia la fase 2 en 1999 y 2000, SOAP comenzó a integrar el trabajo del Grupo de Trabajo del Esquema del W3C, aprovechando sus avances

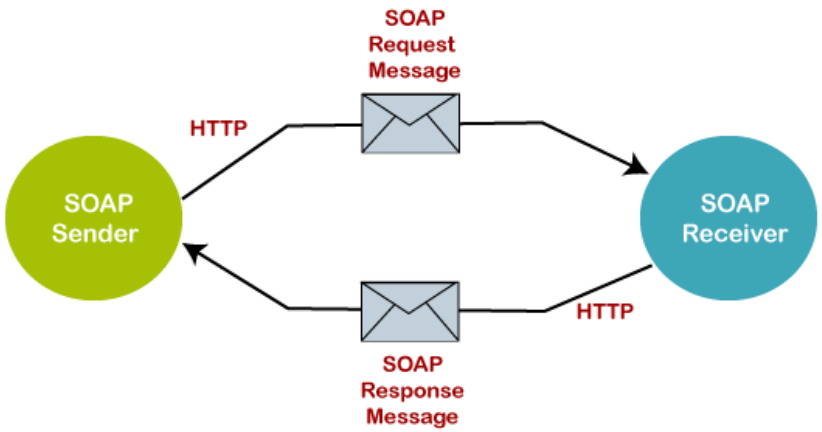
en la definición de tipos primitivos y compuestos en XML. A pesar de los desafíos técnicos y las guerras de proveedores que surgieron durante este período, SOAP continuó evolucionando y consolidándose como un protocolo fundamental para la comunicación en entornos distribuidos.

En la era post-SOAP, que se inició alrededor de 2001, SOAP logró un mayor grado de estabilidad y aceptación en la industria. La especificación del esquema XML se convirtió en una recomendación propuesta, lo que brindó una base sólida para la definición de tipos y estructuras de datos en XML. Además, la creación de un Grupo de Trabajo sobre Protocolo XML en el W3C proporcionó un marco para la evolución continua de SOAP y su integración con estándares relacionados.

En la actualidad, a pesar de la creciente popularidad del protocolo de comunicación REST, SOAP continúa siendo ampliamente utilizado y sigue siendo una pieza fundamental en el desarrollo de sistemas distribuidos y aplicaciones heterogéneas. Su capacidad para facilitar el intercambio de datos de manera interoperable y su flexibilidad para adaptarse a una variedad de entornos y tecnologías lo convierten en una herramienta invaluable en la era digital.

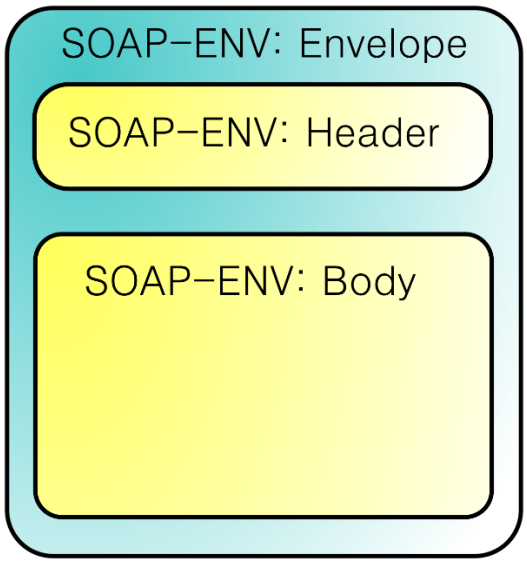
Definición

SOAP, el Protocolo de Acceso a Objetos Simples (Simple Object Access Protocol en inglés), es un protocolo de comunicación utilizado para el intercambio de información en entornos distribuidos y descentralizados. En esencia, SOAP define un conjunto de reglas y estándares para la estructuración y el envío de mensajes entre aplicaciones a través de redes de computadoras, como internet. Estos mensajes SOAP son creados y enviados por una aplicación (llamada "remitente") y recibidos y procesados por otra aplicación (llamada "destinatario").



Tomado de: https://www.javatpoint.com/soap-and-rest-web-services

Los mensajes SOAP están basados en XML (Lenguaje de Marcado Extensible), lo que significa que siguen una estructura jerárquica definida por etiquetas, similar a la estructura de una página web. Esta estructura XML permite que los mensajes SOAP sean fácilmente legibles y procesables tanto por humanos como por máquinas. Un mensaje SOAP contiene tres elementos principales:



Tomado de: https://es.wikipedia.org/wiki/Simple_Object_Access_Protocol

Envelope (Sobre): Define el principio y el final del mensaje SOAP y contiene información sobre cómo procesar el mensaje.

- **Header (Encabezado):** Opcionalmente, contiene información adicional sobre el mensaje, como datos de autenticación, metadatos u otros detalles relevantes para su procesamiento
- **Body (Cuerpo):** Contiene los datos principales del mensaje, es decir, la información que se está transmitiendo desde el remitente hasta el destinatario.

SOAP se utiliza en una variedad de aplicaciones y escenarios, especialmente en sistemas distribuidos donde las aplicaciones necesitan comunicarse entre sí de manera confiable y segura. Por ejemplo, se utiliza comúnmente en servicios web (web services) para permitir la comunicación entre diferentes aplicaciones a través de internet utilizando estándares abiertos y basados en texto como HTTP (Protocolo de Transferencia de Hipertexto) o SMTP (Protocolo Simple de Transferencia de Correo).

Java

Desde 2017, Java se mantiene dentro de los primeros 3 programming languages según The State of Developer Ecosystem. También se encuentra dentro de los primeros 10 en la encuesta de Stack Overflow.

Información salarial

En Colombia el salario promedio de un desarrollador web es de $4.355.000 COP mensuales, en base a 1008 salarios registrados y se consideraría un buen trabajo segú este salario.

Tomado de: https://co.talent.com/salary?job=desarrollador+java

Para la parte laboral internacionalmente se tienen estos salarios anuales en dólares por país obtenidos de clouddevs:

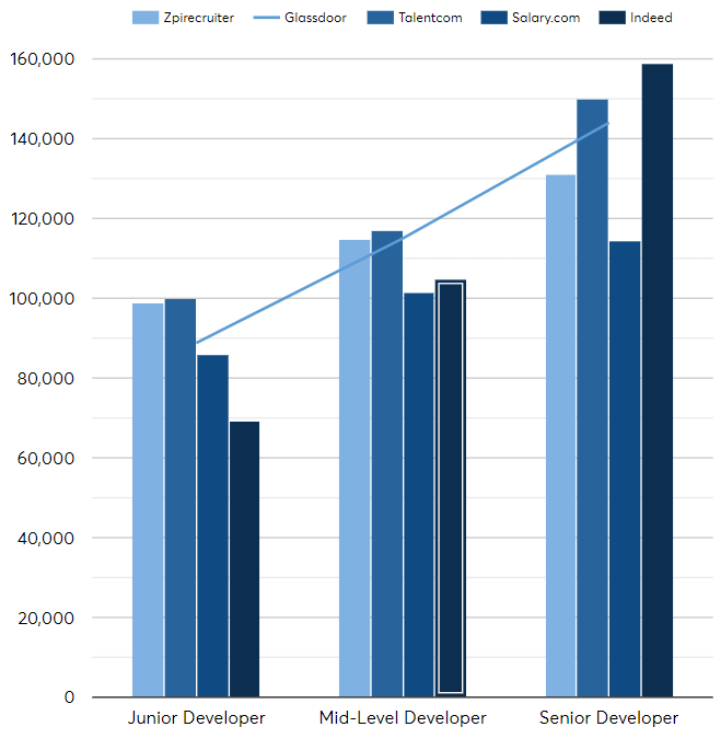
Country	Average Annual Salary (USD)
United States	$114,720
Canada	$102,213
United Kingdom	$92,812
Germany	$83,568
Australia	$105,427
Switzerland	$98,923
Netherlands	$68,564
Belgium	$52,745
Singapore	$63,998
India	$14,692
Kenya	$8,957

Tomado de https://clouddevs.com/java/annual-salaries/

Este es el promedio de lo que gana un desarrollador en java en Estados Unidos anualmente:

Tomado de https://www.ziprecruiter.com/Salaries/Java-Developer-Salary#Yearly

Estos son los promedios salariales según la experticia según buscadores salariales reconocidos como Ziprecuiter, glassdor, talent.com y salary.com



Historia

La historia de Java comienza con el Equipo Verde, también conocido como Green Team, que inició el proyecto para desarrollar un lenguaje para dispositivos digitales en la década de los 90. Originalmente, Java estaba destinado para la televisión interactiva, pero su tecnología avanzada superaba las capacidades de la industria de la televisión por cable digital en ese momento. Sin embargo, pronto se descubrió que Java era más adecuado para la programación en Internet y fue incorporado por Netscape.

Los principios fundamentales de Java, como ser simple, robusto, portátil, independiente de la plataforma y seguro, fueron establecidos por James Gosling y su equipo de ingenieros de Sun Microsystems, quienes iniciaron el proyecto en junio de 1991. Inicialmente, Java fue concebido para sistemas pequeños integrados en dispositivos electrónicos como decodificadores, y fue llamado "Greentalk" antes de ser renombrado como "Oak", como parte del proyecto Green.

La transición de Oak a Java ocurrió en 1995 debido a problemas de marca registrada. El nombre "Java" fue elegido en una búsqueda por algo que reflejara la esencia revolucionaria y dinámica de la tecnología. La elección se inspiró en una isla de Indonesia donde se producía café, reflejando la singularidad y vitalidad de la tecnología Java.

Java se lanzó oficialmente en 1995 y pronto se convirtió en uno de los productos más destacados del año, según la revista Time. Desde entonces, cada nueva versión de Java ha agregado características y funcionalidades adicionales, expandiendo su uso en una variedad de aplicaciones, incluidas aplicaciones web, empresariales, móviles y más.

Hoy Java sigue siendo una de las tecnologías más utilizadas en el mundo del desarrollo de software, con muchas aplicaciones y una comunidad de desarrolladores activa y comprometida que sigue impulsando su evolución.

Definición

Java es un lenguaje de programación de alto nivel y orientado a objetos que se caracteriza por ser simple, robusto, portátil, independiente de la plataforma y seguro. Java se distingue por su capacidad de escribir un programa una vez y ejecutarlo en cualquier dispositivo que tenga un intérprete de Java instalado, lo que lo hace altamente portable.

Además de ser un lenguaje de programación, Java también es una plataforma de software que proporciona un entorno de ejecución (Java Runtime Environment - JRE) y una amplia biblioteca de clases (Java API) que facilita el desarrollo de aplicaciones. La plataforma Java es utilizada en una variedad de campos, desde aplicaciones de escritorio hasta aplicaciones empresariales, aplicaciones web, dispositivos móviles, sistemas embebidos y más.

Tomado de: https://co.pinterest.com/pin/288582288623281688/

Spring Boot

Según el informe de Stack Overflow de 2023, Spring Boot ocupa el puesto número 12 en popularidad entre todas las tecnologías y frameworks web, con un 11.95% de uso entre todos los encuestados. Entre los desarrolladores profesionales, Spring Boot es utilizado por aproximadamente el 13,54%, mientras que entre aquellos que están aprendiendo a codificar, su uso es ligeramente menor, alrededor del 6,82%.

Información salarial

Para Springboot se tiene información de un buen rango salarial al año en Colombia con $63.000.000 COP en promedio y en Estados unidos se tiene un salario promedio de $126.750 USD

Tomado de: https://co.talent.com/salary?job=desarrollador+springboot

Tomado de: https://co.talent.com/salary?job=desarrollador+springboot

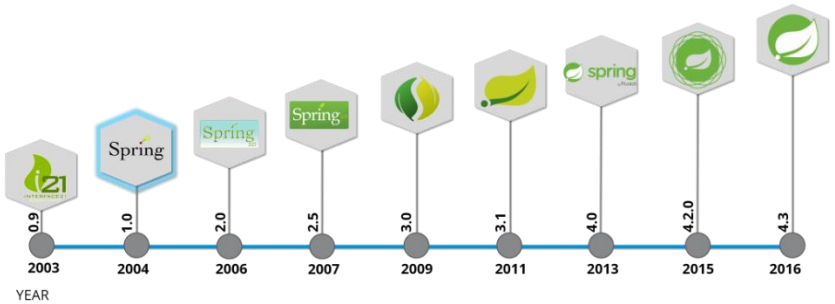
A continuación, compartimos la información salarial de las 10 ciudades mejor pagas para desarrolladores de Springboot:

City	Annual Salary	Monthly Pay	Weekly Pay	Hourly Wage
San Jose, CA	$165,801	$13,816	$3,188	$79.71
Oakland, CA	$162,151	$13,512	$3,118	$77.96
Hayward, CA	$161,877	$13,489	$3,113	$77.83
Antioch, CA	$161,766	$13,480	$3,110	$77.77
Seattle, WA	$160,988	$13,415	$3,095	$77.40
Santa Cruz, CA	$156,612	$13,051	$3,011	$75.29
Seaside, CA	$156,319	$13,026	$3,006	$75.15
Concord, CA	$154,927	$12,910	$2,979	$74.48
Sunnyvale, CA	$154,733	$12,894	$2,975	$74.39
Livermore, CA	$154,650	$12,887	$2,974	$74.35

Tomado de https://www.ziprecruiter.com/Salaries/Mongodb-Developer-Salary

Historia

Spring, un framework de aplicación para el desarrollo de aplicaciones Java empresariales, fue iniciado por Rod Johnson en el año 2000. La primera versión oficial, la 1.0, se lanzó en 2004 y fue gradualmente aceptada por la comunidad de desarrolladores Java. En los años siguientes, evolucionó con versiones como la 2.0 en 2006, que simplificaba la configuración XML y brindaba soporte para Java 5, y la 3.0 en 2009, que introdujo soporte para Java EE6. En 2013, Spring Framework fue trasladado a Pivotal y se lanzó la versión 4.0, con soporte para Java 8. En 2014, se presentó Spring Boot 1.0, que automatizaba la configuración para facilitar el desarrollo rápido de aplicaciones Spring. La evolución continuó con Spring Boot 2.0 en 2017, brindando soporte para Spring 5 y JDK9, junto con otras actualizaciones. En resumen, Spring ha pasado de ser una solución inicial diseñada por Rod Johnson a un marco de trabajo ampliamente adoptado que simplifica el desarrollo de aplicaciones empresariales Java.



Tomado de: https://javadesde0.com/un-poco-de-historia-sobre-spring/

Antecedentes en Spring Framework (2002-2013): En 2002, Rod Johnson creó Spring Framework como una alternativa más ligera y flexible a las tecnologías Java Enterprise Edition (Java EE) de la época. Introdujo conceptos innovadores como la inversión de control (IoC) y la inyección de dependencias (DI), simplificando el desarrollo de aplicaciones empresariales Java. Durante la década siguiente, experimentó un rápido crecimiento y adopción en la comunidad de desarrollo Java.

Aparición de Spring Boot (2014): En 2014, Spring Boot se lanzó como una extensión del popular Spring Framework, con el objetivo de simplificar el desarrollo de aplicaciones Java mediante la adopción de convenciones sobre la configuración y la reducción del código repetitivo. Ofreció configuraciones predeterminadas inteligentes y automatizadas, eliminando la necesidad de configuración manual, lo que permitió a los desarrolladores empezar rápidamente en sus proyectos.

Adopción y crecimiento (2014-2017): Desde su lanzamiento, Spring Boot experimentó una rápida adopción en la comunidad de desarrollo Java. Su enfoque en la simplicidad y la productividad resonó fuertemente entre los desarrolladores, que buscaban formas más eficientes de crear aplicaciones empresariales. Spring Boot se convirtió en una herramienta fundamental para una amplia gama de aplicaciones, desde microservicios hasta aplicaciones monolíticas.

Integración con Spring Cloud y avances en la nube (2013-2018): En 2013, SpringSource (la empresa detrás de Spring) se convirtió en Pivotal Software, y Spring Framework se trasladó a Pivotal. Con esta transición, Spring Boot continuó evolucionando, integrándose estrechamente con Spring Cloud para admitir el desarrollo de aplicaciones en la nube y arquitecturas de microservicios. Spring Boot 2.0, lanzado en 2018, ofreció soporte mejorado para Java 8 y tecnologías emergentes.

Innovaciones y lanzamientos posteriores (2018-2023): En los años siguientes, Spring Boot continuó innovando y mejorando. Se lanzaron nuevas versiones con características avanzadas, como soporte para Java 9 y 11, integración con bases de datos NoSQL, mejoras en la seguridad y el rendimiento, y herramientas para la implementación en la nube.

Definición

Spring Boot es un marco de desarrollo de aplicaciones Java que se basa en los principios de Spring Framework. Se diseñó con el objetivo de simplificar el proceso de desarrollo de aplicaciones Java, eliminando la necesidad de configuraciones manuales y código repetitivo. Con Spring Boot, los desarrolladores pueden comenzar rápidamente a construir aplicaciones empresariales robustas sin la complejidad asociada a la configuración tradicional.

Esta herramienta se destaca por su enfoque en la convención sobre la configuración, lo que significa que proporciona configuraciones predeterminadas inteligentes y configuraciones automáticas para muchos aspectos del desarrollo de aplicaciones. Esto permite a los desarrolladores centrarse más en la lógica de negocio y menos en la configuración de la infraestructura. Spring Boot también se integra estrechamente con otras tecnologías y proyectos de Spring, como Spring Data, Spring Security y Spring Cloud, lo que facilita la creación de aplicaciones en la nube, microservicios y sistemas distribuidos.

MongoDB

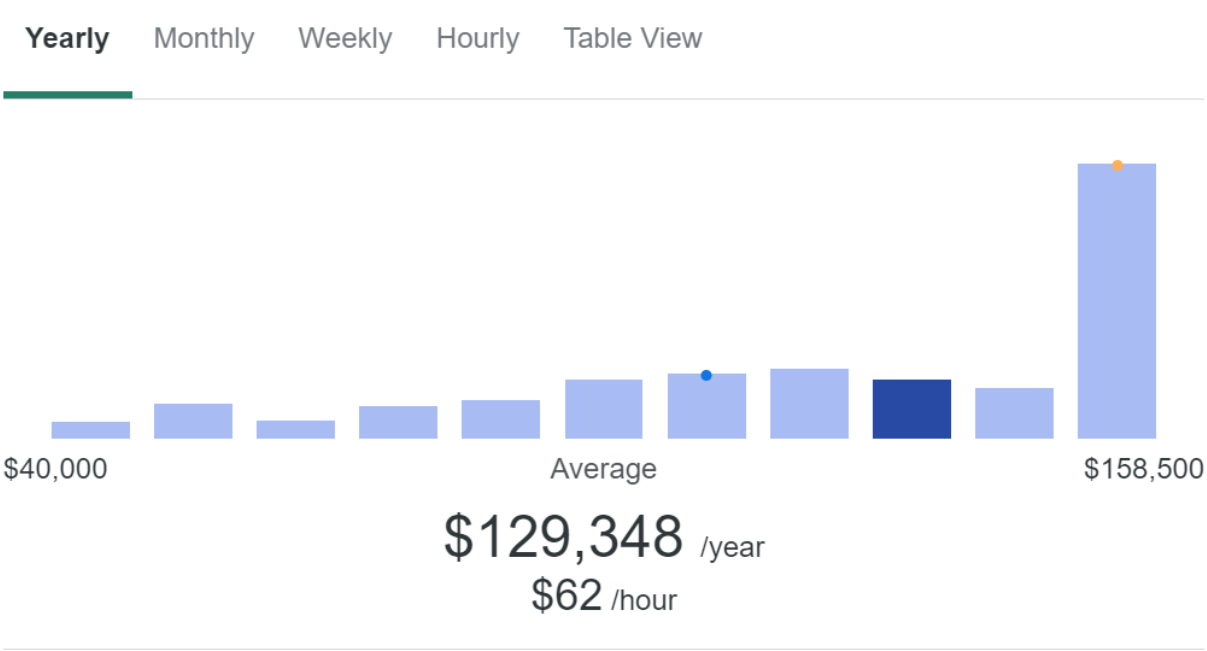
Según los datos recopilados por Stack Overflow en 2023, MongoDB ocupa el cuarto lugar en popularidad entre todos los desarrolladores, con un 25.52% de uso. Sin embargo, es importante destacar que esta posición puede variar según el grupo específico de desarrolladores. Por ejemplo, entre los profesionales desarrolladores, MongoDB ocupa el quinto lugar con un 25.66%, mientras que entre los que están aprendiendo a codificar, MongoDB también está en el quinto lugar con un 28.15%.

Por otro lado, según el sistema de clasificación Red9 en marzo de 2024, MongoDB ocupa el tercer lugar en popularidad entre los sistemas de gestión de bases de datos. Si bien MongoDB ha experimentado un cambio negativo en su posición en comparación con el año anterior, con una disminución de 5 lugares en el ranking de Red9, sigue siendo una opción popular en el mercado.

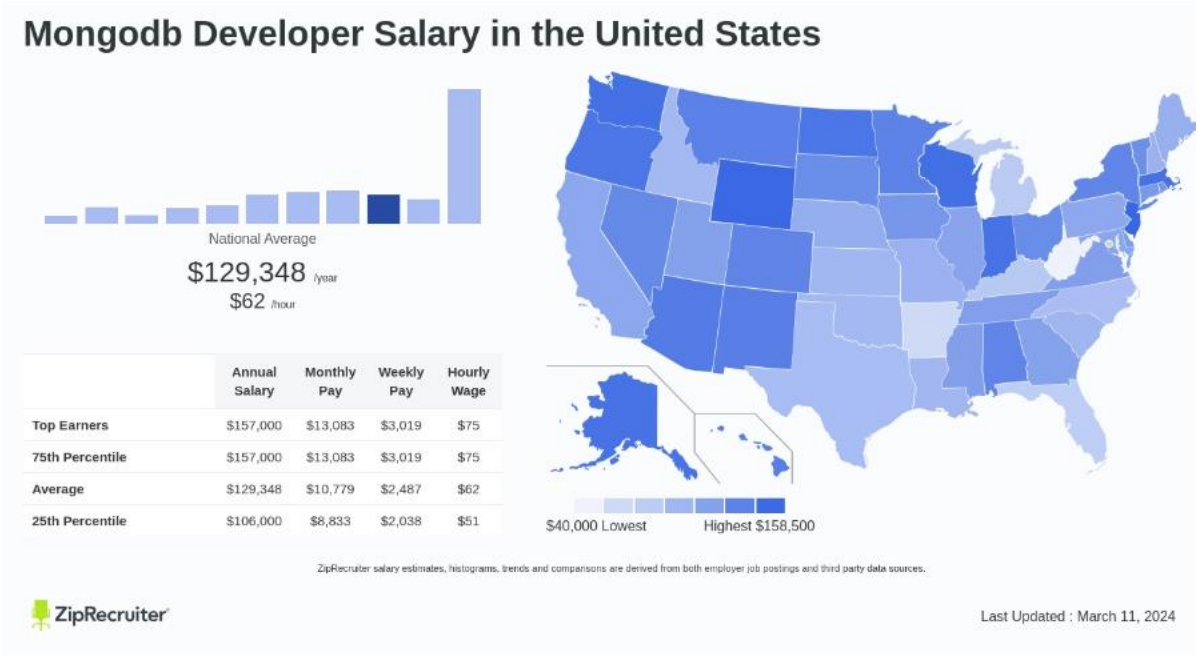
Información Salarial

Información salarial

Como información salarial, un desarrollador de mongodb gana en promedio$129.348USD al año en Estados Unidos, adicionalmente se muestran los estados de dicho país en el que se retratan las zonas en las que se gana más dinero.



Tomado de https://www.ziprecruiter.com/Salaries/Mongodb-Developer-Salary



Tomado de https://www.ziprecruiter.com/Salaries/Mongodb-Developer-Salary

Historia

MongoDB fue creado por Dwight Merriman y Eliot Horowitz en 2007, como respuesta a los desafíos de escalabilidad y agilidad que enfrentaron al desarrollar aplicaciones web en

DoubleClick, una empresa de publicidad en línea que ahora es propiedad de Google. En DoubleClick, el equipo se encontró con problemas al utilizar las bases de datos relacionales tradicionales para manejar grandes cantidades de datos. Inspirados por esta experiencia, decidieron crear una base de datos que pudiera abordar estos desafíos.

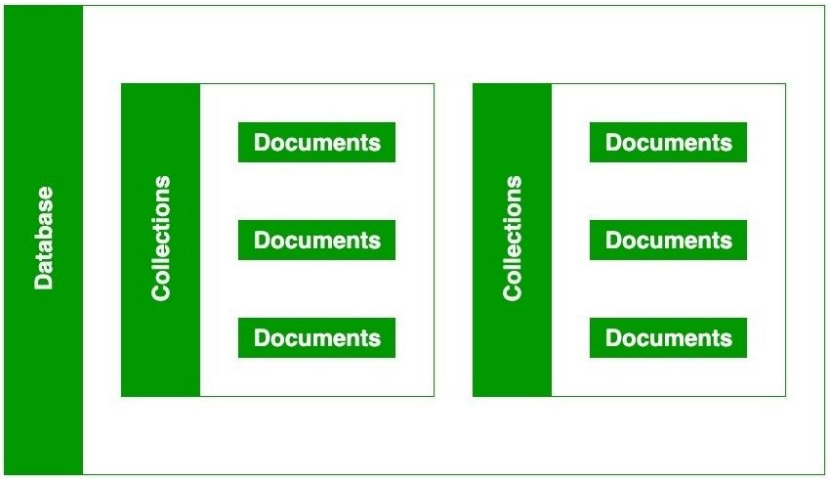
La palabra "Mongo" en MongoDB proviene de "humongous", que significa enorme, reflejando la capacidad de la base de datos para manejar grandes volúmenes de datos. El equipo fundó 10gen Inc. en 2007 para comercializar MongoDB y otros softwares relacionados. Posteriormente, en 2013, la empresa cambió su nombre a MongoDB Inc.

MongoDB, lanzado en 2009 como software de código abierto bajo la Licencia Pública General Affero (AGPL), ofrece también licencias comerciales. Ha sido utilizado por diversas organizaciones para una amplia gama de aplicaciones. La versión estable inicial, 1.4, se lanzó en marzo de 2010. Desde entonces, ha seguido evolucionando, con la última versión estable en ese momento, la 2.4.9, lanzada el 10 de enero de 2014.

En 2017, MongoDB Inc. se hizo pública bajo el símbolo de cotización "MDB" en NASDAQ, lo que marcó un hito significativo en la historia de la empresa. Desde entonces, MongoDB ha experimentado un crecimiento significativo, especialmente con el lanzamiento de MongoDB Atlas en 2016, un servicio de base de datos en la nube que ha ganado una amplia adopción y contribuido en gran medida al éxito continuo de la empresa.

Definición

MongoDB es una base de datos NoSQL (Not Only SQL) de código abierto y orientada a documentos. En lugar de utilizar tablas y filas como las bases de datos relacionales tradicionales, MongoDB almacena datos en documentos flexibles similares a JSON con un esquema dinámico (también conocidos como documentos BSON). A continuación, se muestran las entidades de almacenamiento de datos en MongoDB:



Tomado de: https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/

La flexibilidad del esquema de MongoDB permite a los desarrolladores almacenar y modificar datos de manera rápida y sencilla, facilitando la adaptación a cambios en los requisitos de la aplicación. Es ideal para aplicaciones que manejan grandes volúmenes de datos, necesitan escalabilidad horizontal y requieren un alto rendimiento de lectura y escritura. Algunas características clave incluyen:

- **Esquema flexible**: Los documentos en MongoDB no requieren un esquema fijo y pueden contener diferentes campos y estructuras de datos dentro de la misma colección.
- **Escalabilidad horizontal:** MongoDB puede distribuir datos a través de múltiples servidores, lo que permite un escalado horizontal para manejar grandes volúmenes de datos y cargas de trabajo.
- **Consultas avanzadas:** MongoDB admite consultas complejas utilizando su propio lenguaje de consulta (MongoDB Query Language o MQL), que permite realizar operaciones de filtrado, agregación y búsqueda avanzadas.
- **Alta disponibilidad y tolerancia a fallos:** MongoDB ofrece capacidades integradas de replicación y fragmentación automática para garantizar la disponibilidad continua de los datos y la recuperación ante fallos.

2. Relación entre los temas

Dado que el uso de estas tecnologías en conjunto no es tan común, estas tecnologías son ampliamente utilizadas para el desarrollo web, aunque unas sean más populares o fácil de adoptar que otras.

En primer lugar, la adopción de TypeScript en el desarrollo web ha ganado popularidad debido a su capacidad para agregar tipado estático al código JavaScript, lo que hace que las aplicaciones sean más seguras y fáciles de mantener. Esta integración con TypeScript es particularmente relevante en el contexto de Next.js, un framework de React para el desarrollo de aplicaciones web del lado del cliente y del servidor.

La arquitectura cliente-servidor, es bastante usada para desarrollos web. Next.js se puede utilizar tanto en el lado del cliente como en el servidor, lo que lo hace compatible con esta arquitectura. Por otro lado, SOAP, un protocolo de comunicación ampliamente utilizado para intercambiar información estructurada en la implementación de servicios web, también se relaciona con esta arquitectura al permitir la comunicación entre clientes y servidores de manera efectiva y segura.

Cuando se trata del servidor, Spring Boot, un framework de aplicación Java, es una opción común. Spring Boot facilita el desarrollo de aplicaciones Java con su enfoque en la configuración automática y la convención sobre la configuración, lo que lo hace ideal para implementaciones de servidor en arquitecturas cliente-servidor.

Finalmente, MongoDB, una base de datos NoSQL completa esta relación al proporcionar una solución de almacenamiento de datos flexible y escalable que puede integrarse fácilmente con los diferentes componentes de esta arquitectura. MongoDB es compatible con múltiples frameworks y tecnologías, lo que la hace adecuada para su uso en conjunción con Next.js, Spring Boot y otros componentes de la pila tecnológica mencionada.

Por último, es importante destacar que, si bien algunas de estas tecnologías no se limitan exclusivamente al desarrollo web y pueden aplicarse en diversos contextos, es el ámbito web el que las une y las hace relevantes en esta discusión.

3. Situaciones/problemas de aplicación

Cliente Servidor

- **Aplicaciones web:** En la mayoría de los casos, las aplicaciones web siguen el modelo cliente-servidor. El navegador actúa como el cliente, mientras que el servidor web maneja las solicitudes del cliente y responde con los recursos solicitados, como páginas HTML, imágenes, scripts, etc.
- **Sistemas de mensajería instantánea**: Aplicaciones de mensajería como WhatsApp, Telegram y Facebook Messenger utilizan la arquitectura cliente-servidor. Los clientes se conectan a servidores centrales que gestionan el envío y la recepción de mensajes entre los usuarios.
- **Redes sociales:** Plataformas como Facebook, Twitter e Instagram suelen utilizar la arquitectura cliente-servidor. Los clientes (usuarios) interactúan con servidores centrales que gestionan las publicaciones, la interacción social, la autenticación de usuarios y otras funciones.
- **Sistemas de correo electrónico:** Los clientes de correo electrónico, como Outlook o Gmail, se conectan a servidores de correo que almacenan y gestionan los correos electrónicos de los usuarios. Los clientes envían solicitudes para leer, enviar y gestionar correos electrónicos, y el servidor responde en consecuencia.

Next.js

- **Aplicaciones de renderizado del lado del servidor (SSR):** Next.js es excepcionalmente bueno en el renderizado del lado del servidor, lo que lo hace ideal para aplicaciones que requieren una carga inicial rápida, una indexación SEO sólida y una experiencia de usuario receptiva.
- **Aplicaciones de comercio electrónico:** En el comercio electrónico, la velocidad y la usabilidad son críticas. Next.js proporciona un entorno ideal para construir experiencias de compra en línea rápidas y fluidas, gracias a su capacidad para manejar el enrutamiento dinámico, la previsualización de páginas y la gestión eficiente del estado de la aplicación.
- **Aplicaciones de la vida real:** Netflix utiliza Next.js para partes de su plataforma web, proporcionando un rendimiento rápido y una experiencia de usuario fluida para sus usuarios. También, Twitch utiliza Next.js para su aplicación web, lo que permite una carga rápida de contenido y una navegación suave entre páginas. Y otras más, como Notion, TikTok, Ticketmaster, entre otros.

TypeScript

- **Desarrollo web:** TypeScript es comúnmente utilizado en el desarrollo web para crear aplicaciones tanto del lado del cliente como del servidor. Puede ser aplicado en la construcción de aplicaciones web complejas, aplicaciones de una sola página (SPA), entre otros.
- **Desarrollo de API:** TypeScript puede ser empleado en la creación de API RESTful o GraphQL, proporcionando tipado estático para los datos de entrada y salida, lo que ayuda a evitar errores comunes durante el desarrollo.
- **Procesamiento de datos en el servidor:** TypeScript es útil en el desarrollo de aplicaciones en el lado del servidor utilizando Node.js. Proporciona ventajas como la detección de errores y la autocompletación en el código, lo que facilita el desarrollo de aplicaciones escalables y mantenibles.
- **Aplicaciones específicas:** Angular, es un framework de desarrollo de aplicaciones web y móviles de Google. Está escrito en TypeScript y permite la creación de aplicaciones web dinámicas y de una sola página (SPA). También, NestJS, un framework de desarrollo de aplicaciones backend Node.js altamente modular y escalable. NestJS utiliza TypeScript para permitir la creación de aplicaciones server-side robustas, con inyección de dependencias, decoradores, entre otros.

SOAP

- **Transferencia de datos estructurados:** Si necesita enviar datos estructurados entre aplicaciones de manera confiable y segura, SOAP proporciona un formato XML estándar para este propósito.
- **Aplicaciones empresariales:** En entornos empresariales donde se necesita una comunicación segura y fiable entre sistemas y aplicaciones, SOAP es una opción común debido a su soporte para mecanismos de seguridad como WS-Security.
- **Procesos de negocio automatizados:** SOAP se utiliza en aplicaciones de negocio para invocar operaciones remotas y automatizar procesos de negocio a través de la comunicación entre sistemas distribuidos.
- **Transacciones financieras:** En aplicaciones financieras donde la seguridad y la integridad de los datos son críticas, SOAP puede utilizarse para la comunicación entre diferentes instituciones financieras y sistemas.
- **Integración con sistemas legados:** En entornos donde existen sistemas heredados que no son compatibles con los estándares modernos de comunicación, SOAP puede actuar como un puente para integrar estos sistemas con aplicaciones más nuevas.
- **Interacción con servicios de terceros:** Cuando se necesita interactuar con servicios proporcionados por terceros, como servicios de pago, servicios de envío, etc., SOAP puede ser utilizado para establecer comunicación segura y estructurada.
- **Sistemas de gestión empresarial (ERP):** Muchos sistemas ERP utilizan SOAP para comunicarse entre módulos y con sistemas externos, garantizando una transferencia segura de datos estructurados entre diferentes partes del sistema.
- **Integración de sistemas de gestión de clientes (CRM):** Las aplicaciones CRM a menudo utilizan SOAP para intercambiar datos con otros sistemas dentro de la organización, como sistemas ERP, sistemas de contabilidad, etc., garantizando una comunicación segura y fiable.

Java

- **Desarrollo de aplicaciones web:** Java es ampliamente utilizado en el desarrollo de aplicaciones web, ya sea mediante el uso de tecnologías como Java Servlets y JavaServer Pages (JSP), o utilizando frameworks modernos como Spring o Java EE para construir aplicaciones empresariales.

- **Desarrollo de aplicaciones de escritorio:** Con la plataforma Java SE, puedes desarrollar aplicaciones de escritorio multiplataforma utilizando bibliotecas como JavaFX o Swing.
- **Desarrollo de aplicaciones móviles:** Aunque ha perdido popularidad frente a otras plataformas como Android (que utiliza principalmente Kotlin ahora), Java todavía se utiliza para desarrollar aplicaciones móviles Android.
- **Desarrollo de aplicaciones de servidor:** Java es muy utilizado en el desarrollo de aplicaciones de servidor debido a su robustez y escalabilidad. Puede usarse para construir servidores de aplicaciones, servicios web, sistemas de gestión de bases de datos, etc.
- **Aplicaciones específicas:** En 2004, Java se convirtió en el primer lenguaje de programación utilizado en la exploración del espacio cuando los científicos de la NASA en el Laboratorio de Propulsión a Chorro (JPL) utilizaron el Maestro Science Activity Planner, basado en Java, para controlar el Spirit Mars Exploration Rover durante su misión en Marte. Esta aplicación marcó un hito histórico en la aplicación de Java en la exploración espacial.

Spring boot

- **Desarrollo de aplicaciones web:** Spring Boot es ideal para desarrollar aplicaciones web. Puede manejar fácilmente la lógica de negocio, la gestión de usuarios, la seguridad y la integración con bases de datos.
- **Microservicios:** Spring Boot es una opción popular para construir y desplegar microservicios. Permite a los desarrolladores crear, desplegar y gestionar fácilmente pequeños servicios independientes que funcionan juntos para formar una aplicación completa.
- **API RESTful:** Spring Boot facilita la creación de API RESTful para exponer funcionalidades de tu aplicación a otros sistemas o clientes. Puedes usar Spring MVC o Spring WebFlux para desarrollar y gestionar tus endpoints REST.
- **Algunos casos de aplicación:** JPMorgan Chase utiliza Spring Boot para sus sistemas de banca online y móvil. También, Udemy utiliza Spring Boot para gestionar y escalar su oferta de cursos y las interacciones de los usuarios de manera eficiente. Por otra parte, está Trivago que emplea Spring Boot para acortar el tiempo necesario en el desarrollo de aplicaciones web y consolidar las consultas de grandes volúmenes de usuarios y datos de múltiples sitios de reservas de hoteles.

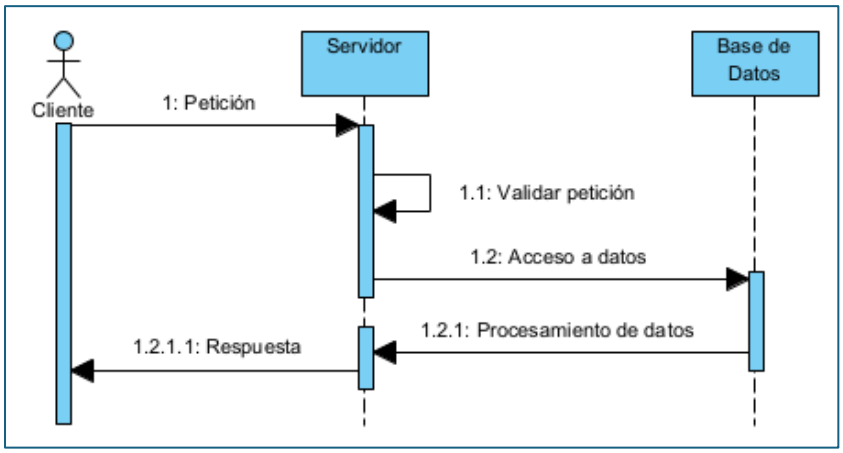
MongoDB

- **Aplicaciones web:** MongoDB es ideal para el desarrollo de aplicaciones web, especialmente aquellas que manejan grandes cantidades de datos no estructurados o semi-estructurados, como redes sociales, sistemas de gestión de contenidos (CMS), comercio electrónico, etc.
- **Big Data y análisis de datos:** MongoDB puede ser utilizado en entornos de big data donde se necesite una base de datos escalable y flexible para almacenar y analizar grandes volúmenes de datos, como registros de eventos, registros de servidores, datos de sensores, etc.
- **Gestión de contenido:** Para sistemas de gestión de contenido que manejan contenido multimedia, documentos, metadatos, etc., MongoDB es una opción popular debido a su capacidad para manejar datos no estructurados y semiestructurados de manera eficiente.
- **Sistemas de recomendación:** En sistemas de recomendación que analizan el comportamiento del usuario y generan recomendaciones personalizadas, MongoDB puede utilizarse para almacenar perfiles de usuario, historiales de comportamiento, entre otros.
- **IoT (Internet de las cosas):** Para aplicaciones de IoT que recopilan datos de una variedad de dispositivos y sensores, MongoDB proporciona una solución escalable y flexible para almacenar y analizar estos datos en tiempo real. Un ejemplo de esto es Bosch, está desarrollando una aplicación de IoT que captura datos del vehículo para realizar diagnósticos de mantenimiento preventivo.
- **Aplicaciones de análisis de redes sociales:** Para aplicaciones que analizan datos de redes sociales, MongoDB proporciona una solución eficiente para almacenar y analizar grandes volúmenes de datos no estructurados.
- **Algunos casos de aplicación:** MetLife utiliza MongoDB para crear un repositorio central que ofrece una visión integral de sus más de cien millones de clientes. También, Expedia creó Scratchpad, una aplicación basada en MongoDB que permite a los usuarios almacenar y personalizar sus búsquedas de viajes, ofreciendo ofertas especiales en tiempo real basadas en el comportamiento del usuario. Por otro lado, Forbes utiliza MongoDB para todo su sistema de gestión de contenidos, así como para analítica en tiempo real para detectar tendencias virales y adaptar su contenido en consecuencia.

4. Representación UML

Arquitectura cliente-servidor

Diagrama de Secuencia



Este diagrama de secuencia UML ilustra la interacción entre un cliente y un servidor durante una consulta de datos.

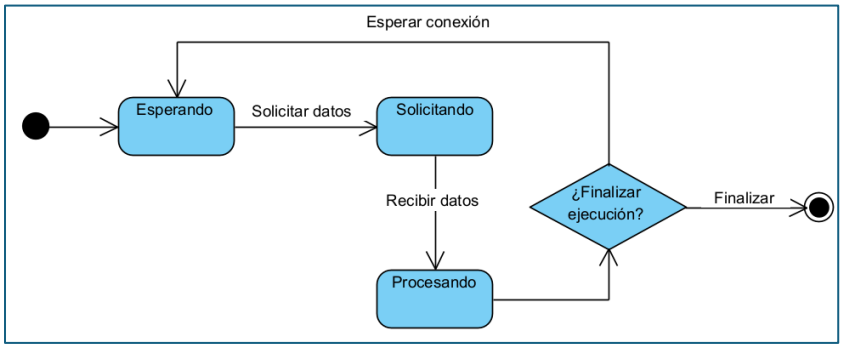
- Participantes
  - **Cliente:** Representa la aplicación o usuario que inicia la solicitud.
  - **Servidor:** Representa el sistema que procesa la solicitud y devuelve los datos.

- Flujo de mensajes
  - **Inicio:** El cliente envía un mensaje al servidor solicitando datos específicos, incluyendo el identificador de la consulta o los parámetros de búsqueda.

- **Validación:** El servidor recibe la solicitud y la valida. Si es válida, procede al siguiente paso.
- **Acceso a datos:** El servidor accede a la base de datos o almacén de datos para recuperar la información solicitada.
- **Procesamiento:** Los datos son procesados según sea necesario por el servidor, como filtrar, ordenar o transformarlos.
- **Respuesta:** El servidor envía un mensaje al cliente con los datos recuperados y procesados.
- **Recepción:** El cliente recibe los datos y los procesa si es necesario.
- **Finalización:** El cliente termina la interacción con el servidor.

Diagrama de máquina de estados

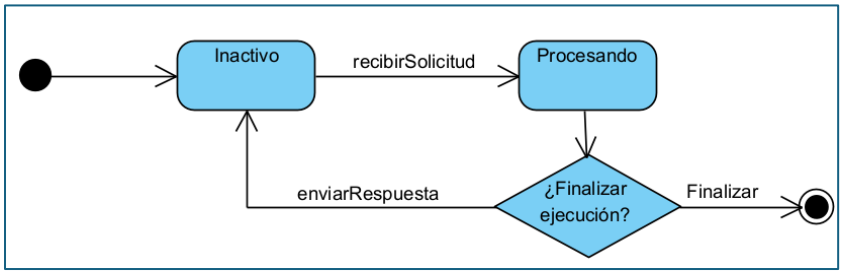
Cliente



- Estados
  - Esperando
    - **Descripción:** Este es el estado inicial del cliente, donde está inactivo y espera alguna acción para realizar.
    - **Evento:** solicitarDatos
    - **Transición:** Cuando el cliente decide solicitar datos, pasa al estado Solicitando.
  - Solicitando
    - **Descripción:** En este estado, el cliente ha enviado una solicitud de datos y está esperando recibirlos.
    - **Evento:** recibirDatos
    - **Transición:** Cuando el cliente recibe los datos solicitados, pasa al estado Procesando.
  - Procesando

        - **Descripción:** El cliente ha recibido los datos y está ocupado procesándolos.
        - **Decisión:** ¿Finalizar ejecución?
            - **Evento:** finalizar
            - **Transición:** Cuando el cliente ha completado el procesamiento de los datos y decide terminar su ejecución.
            - **Evento:** esperarConexión
            - **Transición:** Cuando el cliente ha completado el procesamiento de los datos, vuelve al estado Esperando para estar listo para la próxima acción.

Servidor



- Estados
    - Inactivo
        - **Descripción:** Este es el estado inicial del servidor, donde está pasivamente esperando una solicitud para procesar.
        - **Evento:** recibirSolicitud
        - **Transición:** Cuando el servidor recibe una solicitud, pasa al estado Procesando.
    - Procesando
        - **Descripción:** En este estado, el servidor está ocupado procesando la solicitud recibida.
            - **Decisión:** ¿Finalizar ejecución?
                - **Evento:** finalizar
                - **Transición:** Cuando el cliente ha completado el procesamiento de los datos y decide terminar su ejecución.
                - **Evento:** enviarRespuesta
                - **Transición:** Una vez que el servidor ha procesado la solicitud y está listo para enviar una respuesta, realiza una transición al estado Inactivo.

5. Ventajas y desventajas

Arquitectura cliente-servidor

Ventajas

- Escalabilidad: La capacidad de agregar más clientes o servidores según sea necesario facilita la gestión de un mayor número de solicitudes, lo que permite que el sistema crezca de manera flexible.
- Seguridad robusta: Al tener un punto centralizado de acceso y control, es más sencillo implementar y gestionar medidas de seguridad más sólidas, protegiendo los datos y recursos del sistema de manera efectiva.
- Mantenimiento simplificado: La capacidad de realizar cambios y actualizaciones en el servidor sin afectar a los clientes facilita el mantenimiento del sistema, ya que se pueden aplicar parches y mejoras de manera centralizada.

Desventajas

- Dependencia de la red: La comunicación entre cliente y servidor está sujeta a la disponibilidad y calidad de la red. Si la conexión falla, se interrumpe la comunicación, lo que puede afectar la funcionalidad del sistema.
- Cuellos de botella: Cuando hay un gran flujo de clientes solicitando recursos al servidor, puede generarse un cuello de botella que afecte el rendimiento del sistema, reduciendo la capacidad de respuesta y provocando tiempos de espera prolongados.
- Latencia: Debido al tiempo que tardan las solicitudes y respuestas en viajar a través de la red, puede haber un retraso en la comunicación entre cliente y servidor, lo que puede impactar la experiencia del usuario, especialmente en aplicaciones sensibles al tiempo de respuesta.

Next.js

Ventajas

- Renderización del lado del servidor (SSR) y generación de sitios estáticos (SSG): Next.js permite renderizar tanto en el servidor como de manera estática, lo que mejora el rendimiento y la SEO de las aplicaciones web al proporcionar contenido pre-renderizado.
- Enrutamiento automático: Next.js proporciona enrutamiento automático basado en la estructura de archivos del proyecto, lo que simplifica la configuración y la gestión de las rutas en la aplicación.
- Recarga automática: El servidor de desarrollo de Next.js proporciona recarga automática, lo que significa que los cambios en el código se reflejan instantáneamente en el navegador durante el desarrollo, mejorando la productividad del desarrollador.
- Optimización de imágenes: Next.js ofrece herramientas integradas para optimizar el manejo de imágenes, como la compresión automática y el soporte para formatos de imagen modernos, lo que mejora el rendimiento de la aplicación.
- Soporte para CSS en línea y CSS modular: Next.js permite el uso de CSS en línea y CSS modular, lo que facilita la creación y el mantenimiento de estilos en la aplicación.

Desventajas

- Curva de aprendizaje: Para los desarrolladores nuevos en React o en frameworks basados en React, como Next.js, puede haber una curva de aprendizaje significativa, especialmente para comprender conceptos avanzados como SSR y SSG.
- Configuración inicial compleja: Aunque Next.js simplifica muchas tareas de desarrollo web, la configuración inicial puede resultar compleja, especialmente para proyectos más grandes o personalizados que requieren ajustes específicos.
- Tamaño de la aplicación: Dependiendo de las características utilizadas y de cómo se configure la aplicación, el tamaño del bundle puede ser considerable, lo que podría afectar el tiempo de carga inicial de la aplicación, especialmente en conexiones de red más lentas.

TypeScript

Ventajas

- Detección de errores en tiempo de compilación: TypeScript es un lenguaje de programación tipado estáticamente, lo que significa que permite detectar errores de

sintaxis, de tipo y otros errores comunes durante la fase de compilación. Esto ayuda a reducir la cantidad de errores en el código y a mejorar la calidad del software.

- Mejor mantenibilidad del código: Gracias a la tipificación estática y a las herramientas de IDE que ofrecen sugerencias y autocompletado basadas en tipos, el código escrito en TypeScript tiende a ser más legible y mantenible. La tipificación explícita hace que sea más fácil entender la intención del código y colaborar con otros desarrolladores en proyectos grandes.
- Refactorización segura: La tipificación estática de TypeScript permite realizar refactorizaciones de manera segura. Los cambios en el código pueden ser refactorizados con confianza, ya que el compilador de TypeScript detectará cualquier error de tipo que pueda surgir como resultado de la refactorización.
- Mejora la productividad del equipo: TypeScript puede mejorar la productividad del equipo de desarrollo al reducir el tiempo dedicado a depurar errores de tipo y a comprender el código escrito por otros miembros del equipo. Además, las características como las interfaces y los tipos de unión permiten expresar de manera más clara la intención del código, lo que facilita la comunicación entre los miembros del equipo.
- Ecosistema de herramientas sólido: TypeScript cuenta con un ecosistema de herramientas robusto que incluye IDEs como Visual Studio Code y WebStorm, así como herramientas de construcción como webpack y parcel. Estas herramientas proporcionan soporte avanzado para TypeScript, lo que facilita el desarrollo de aplicaciones web modernas y escalables.

Desventajas

- Curva de aprendizaje inicial: Para aquellos que no están familiarizados con TypeScript o que están acostumbrados a trabajar solo con JavaScript, puede haber una curva de aprendizaje inicial significativa. TypeScript introduce nuevos conceptos, como tipos estáticos y anotaciones de tipo, que pueden requerir tiempo para entender completamente.
- Documentación menos abundante: Aunque TypeScript ha ganado popularidad en los últimos años, su documentación y recursos de aprendizaje pueden ser menos abundantes en comparación con JavaScript puro. Esto puede hacer que sea más difícil encontrar soluciones a problemas específicos o aprender mejores prácticas.
- Posible sobrecarga de tipos: En algunos casos, especialmente en proyectos pequeños o en partes específicas de un proyecto, el uso excesivo de tipos en TypeScript puede resultar en una sobrecarga de tipos, lo que significa que el

beneficio de tener tipos estáticos puede no compensar el esfuerzo adicional requerido para mantenerlos.

SOAP

Ventajas

- Interoperabilidad: SOAP permite la comunicación entre sistemas heterogéneos, ya que está basado en estándares abiertos como XML, HTTP y otros protocolos de Internet. Esto facilita la integración de sistemas desarrollados en diferentes tecnologías y plataformas.
- Seguridad: SOAP proporciona soporte integrado para estándares de seguridad como SSL/TLS para la encriptación de datos y WS-Security para la autenticación y autorización de usuarios, lo que garantiza la confidencialidad, integridad y autenticidad de los mensajes transmitidos.
- Robustez: SOAP es un protocolo robusto y bien establecido que ofrece características como manejo de errores, reintentos automáticos y transacciones atómicas, lo que garantiza la fiabilidad y la consistencia de las operaciones realizadas entre sistemas distribuidos.
- Soporte para servicios web complejos: SOAP es adecuado para la implementación de servicios web complejos que requieren funcionalidades avanzadas como transacciones distribuidas, coordinación de servicios y gestión de transacciones, lo que lo hace ideal para entornos empresariales.
- Compatibilidad con estándares industriales: SOAP es compatible con una amplia variedad de estándares industriales como WSDL (Web Services Description Language) para la descripción de servicios web, UDDI (Universal Description, Discovery, and Integration) para la publicación y descubrimiento de servicios, y otros estándares relacionados.

Desventajas

- Complejidad: SOAP puede resultar complejo de implementar y entender debido a su naturaleza basada en XML y a la cantidad de especificaciones y estándares asociados. Esto puede aumentar la sobrecarga de desarrollo y dificultar la depuración de problemas.
- Overhead de datos: SOAP utiliza XML para representar los mensajes, lo que puede generar un overhead de datos significativo en comparación con protocolos más ligeros como REST. Esto puede afectar el rendimiento de las aplicaciones, especialmente en entornos con ancho de banda limitado.

- Velocidad de desarrollo: Debido a su complejidad, SOAP puede requerir más tiempo y esfuerzo para desarrollar y mantener en comparación con alternativas más simples como REST. Esto puede ser una limitación en proyectos donde se prioriza la velocidad de desarrollo sobre otros aspectos.
- Menor flexibilidad: SOAP tiende a ser menos flexible que REST en términos de manipulación de datos y elección de protocolos de transporte. Esto puede ser una desventaja en situaciones donde se requiere una mayor flexibilidad y agilidad en el intercambio de datos entre sistemas.
- Menos soporte en dispositivos móviles y navegadores: SOAP puede tener menos soporte en dispositivos móviles y navegadores en comparación con REST, lo que puede limitar su utilidad en entornos donde la interoperabilidad con aplicaciones cliente es una consideración importante.

Java

Ventajas

- Portabilidad: Una de las características más destacadas de Java es su capacidad de ser ejecutado en cualquier plataforma que tenga una máquina virtual Java (JVM) disponible. Esto significa que el código Java puede ejecutarse en diferentes sistemas operativos sin necesidad de recompilación, lo que facilita la creación de aplicaciones multiplataforma.
- Orientación a objetos: Java es un lenguaje orientado a objetos, lo que permite una programación modular y un diseño de software más estructurado y reutilizable. La encapsulación, la herencia y el polimorfismo son características fundamentales de Java que facilitan el desarrollo de aplicaciones complejas.
- Gran comunidad y ecosistema: Java cuenta con una gran comunidad de desarrolladores y una amplia variedad de bibliotecas, frameworks y herramientas disponibles. Esto facilita el desarrollo rápido de aplicaciones y proporciona soluciones para una amplia gama de problemas comunes.
- Seguridad: Java está diseñado con un enfoque en la seguridad, con características como el sandbox de seguridad que ayuda a prevenir la ejecución de código malicioso. Además, la verificación de tipos estática de Java ayuda a detectar errores de programación antes de que el código se ejecute, lo que contribuye a la seguridad y estabilidad del software.
- Rendimiento: Aunque Java es un lenguaje interpretado, la JVM utiliza técnicas de optimización, como la compilación en tiempo de ejecución (JIT), para mejorar el rendimiento del código. Java también cuenta con una gestión eficiente de la

memoria a través del recolector de basura, que automatiza la gestión de la memoria y reduce el riesgo de fugas de memoria.

Desventajas

- Consumo de recursos: Aunque la gestión automática de la memoria en Java facilita el desarrollo de aplicaciones al evitar fugas de memoria, también puede resultar en un mayor consumo de recursos en comparación con lenguajes de programación con gestión manual de la memoria, como C++.
- Tiempo de arranque: Las aplicaciones Java pueden tener un tiempo de arranque más lento en comparación con aplicaciones nativas debido al tiempo requerido para cargar la JVM y el bytecode Java.
- Limitaciones de rendimiento en aplicaciones de tiempo real: Aunque Java ofrece un buen rendimiento en la mayoría de los casos, puede no ser la mejor opción para aplicaciones de tiempo real que requieren tiempos de respuesta extremadamente rápidos y predecibles.
- Problemas de seguridad en versiones antiguas: Aunque Java está diseñado con un enfoque en la seguridad, las versiones antiguas de Java han sido objeto de vulnerabilidades de seguridad conocidas. Es importante mantenerse actualizado con las últimas versiones de Java y aplicar parches de seguridad para mitigar estos riesgos.

Spring Boot

Ventajas

- Facilidad de configuración: Spring Boot utiliza convenciones sobre configuración, lo que significa que proporciona configuraciones predeterminadas sensatas para muchas características. Esto reduce la cantidad de configuración manual necesaria y facilita el inicio rápido de un proyecto.
- Integración con Spring Framework: Spring Boot se basa en el sólido ecosistema de Spring Framework, lo que significa que hereda todas las ventajas de Spring, como la inversión de control (IoC), la inyección de dependencias y el desarrollo orientado a aspectos (AOP).
- Desarrollo rápido: Spring Boot proporciona una gran cantidad de características listas para usar, como manejo de solicitudes HTTP, acceso a bases de datos, seguridad y mucho más. Esto acelera el desarrollo de aplicaciones al permitir a los desarrolladores centrarse en la lógica de negocio en lugar de en la infraestructura.

- Gestión automática de dependencias: Spring Boot utiliza Maven o Gradle para la gestión de dependencias, lo que facilita la inclusión y gestión de bibliotecas y frameworks externos en un proyecto.
- Soporte para microservicios: Spring Boot es muy adecuado para el desarrollo de aplicaciones basadas en microservicios debido a su enfoque en la modularidad y la escalabilidad. También proporciona integraciones con tecnologías como Spring Cloud para simplificar aún más el desarrollo de arquitecturas de microservicios.

Desventajas

- Curva de aprendizaje inicial: Aunque Spring Boot simplifica muchos aspectos del desarrollo de aplicaciones, aún puede haber una curva de aprendizaje inicial para aquellos que no están familiarizados con el ecosistema de Spring.
- Configuración compleja en proyectos grandes: A medida que un proyecto Spring Boot crece en tamaño y complejidad, la gestión de la configuración puede volverse más complicada. Esto puede requerir una comprensión más profunda de las características de Spring Boot y la personalización de la configuración predeterminada.
- Complejidad para proyectos pequeños: Para proyectos pequeños o simples, Spring Boot puede ser demasiado potente y complejo, lo que puede resultar en un exceso de abstracción y complejidad innecesaria.

MongoDB

Ventajas

- Modelo de datos flexible: MongoDB utiliza un modelo de datos flexible basado en documentos BSON (formato binario JSON). Esto permite almacenar datos de forma estructurada o no estructurada, lo que proporciona flexibilidad para adaptarse a los cambios en los requisitos de la aplicación.
- Escalabilidad horizontal: MongoDB está diseñado para escalar horizontalmente de manera eficiente, lo que significa que puedes agregar más servidores para manejar un mayor volumen de datos y tráfico sin sacrificar el rendimiento.
- Alta disponibilidad y tolerancia a fallos: MongoDB ofrece características integradas para la replicación y el failover automático, lo que garantiza la alta disponibilidad de los datos y minimiza el tiempo de inactividad en caso de fallos del servidor.
- Índices y consultas eficientes: MongoDB admite índices secundarios y consultas complejas, lo que permite realizar consultas rápidas y eficientes incluso en grandes volúmenes de datos.

- Escalabilidad y rendimiento: MongoDB está diseñado para funcionar en entornos distribuidos y puede aprovechar al máximo los recursos de hardware disponibles, lo que permite obtener un rendimiento óptimo incluso en aplicaciones con alta carga de trabajo.

Desventajas

- Consistencia eventual: MongoDB utiliza un modelo de consistencia eventual por defecto en entornos distribuidos, lo que significa que los datos pueden tardar algún tiempo en propagarse a todos los nodos en el clúster. Esto puede resultar en una consistencia débil en algunas situaciones.
- Consumo de memoria y almacenamiento: MongoDB puede consumir una cantidad significativa de memoria y almacenamiento, especialmente en entornos con grandes volúmenes de datos o con un alto número de operaciones de escritura.
- Curva de aprendizaje: Para aquellos que están acostumbrados a trabajar con bases de datos relacionales, puede haber una curva de aprendizaje al adoptar MongoDB y aprender a diseñar y consultar bases de datos NoSQL de manera eficiente.
- Falta de soporte para transacciones ACID a nivel de documentos: Aunque MongoDB ofrece transacciones ACID a nivel de colección desde la versión 4.0, no soporta transacciones ACID a nivel de documentos, lo que puede ser una limitación en algunos casos de uso.
- Herramientas y ecosistema menos maduros que los de las bases de datos relacionales: Aunque MongoDB cuenta con un sólido ecosistema de herramientas y bibliotecas, puede ser menos maduro que el de las bases de datos relacionales más establecidas como MySQL o PostgreSQL.

6. Principios SOLID

Arquitectura cliente-servidor

- En la arquitectura cliente-servidor, cada componente (cliente y servidor) debe tener una única responsabilidad y motivo para cambiar. Por ejemplo, el servidor debe ser responsable de manejar la lógica de negocios y el acceso a datos, mientras que el cliente debe ser responsable de la presentación de la interfaz de usuario y la interacción con el usuario final.
- El principio Abierto/Cerrado establece que las entidades de software deben ser abiertas para su extensión, pero cerradas para su modificación. En el contexto cliente-servidor, esto se puede traducir a que tanto el cliente como el servidor deben estar diseñados de manera que puedan extenderse para agregar nuevas funcionalidades sin modificar el código existente. También se podrían tener más clientes o servidores según se necesite sin afectar los ya existentes.

Next.js

- En Next.js, cada componente o módulo debería tener una única responsabilidad y motivo para cambiar. Por ejemplo, los componentes de React en Next.js deberían estar enfocados en una sola tarea, como la presentación de datos o la gestión de la lógica de negocios, siguiendo así el principio de SRP.
- Los módulos en una aplicación Next.js deberían estar abiertos para la extensión, pero cerrados para la modificación directa. Esto se logra mediante el uso de patrones como la composición y la herencia, lo que permite extender el comportamiento de los componentes sin modificar su código fuente original.

TypeScript

- En TypeScript, cada clase o función debería tener una única responsabilidad. Esto significa que una clase o función debería hacer una sola cosa y hacerla bien. Al seguir este principio, se puede mejorar la cohesión y la claridad del código.
- TypeScript permite la extensión de clases y la implementación de interfaces, lo que facilita la aplicación del principio de OCP. Al diseñar clases y módulos, se deben hacer de manera que puedan ser extendidos para añadir nuevas funcionalidades sin necesidad de modificar el código existente.
- TypeScript es un lenguaje orientado a objetos, por lo que se puede aplicar el principio de LSP al diseñar jerarquías de clases. Esto implica que las subclases deberían poder ser sustituidas por sus superclases sin afectar el comportamiento esperado del programa.

Java

En java, debido a que es un lenguaje orientado a objetos, se espera poder cumplir con todos los principios SOLID, sin embargo, la utilización de estos depende también del contexto o necesidad de la solución que se esté creando

Spring Boot

- En Spring Boot, se puede aplicar el principio de SRP a los componentes de la aplicación, como los controladores, servicios y repositorios. Cada componente debe tener una única responsabilidad, como manejar la lógica de negocio, la gestión de datos o la presentación, y no debería tener más de una razón para cambiar.
- En Spring Boot, se puede aplicar el principio de OCP mediante el uso de la inyección de dependencias y la configuración basada en anotaciones. Los componentes de la aplicación pueden ser extendidos sin necesidad de modificar su código fuente, lo que facilita la adición de nuevas funcionalidades o cambios de comportamiento.
- Spring Boot permite la implementación de interfaces y la programación basada en contratos, lo que facilita la sustitución de implementaciones concretas por otras compatibles. Por ejemplo, al definir interfaces para los servicios y repositorios, se puede intercambiar fácilmente entre diferentes implementaciones sin modificar el código que las utiliza.
- En Spring Boot, se pueden definir interfaces específicas y cohesivas que reflejen las responsabilidades de los componentes. Por ejemplo, al definir interfaces de servicios con métodos específicos para cada función, se evita que los clientes dependan de funcionalidades que no necesitan.

- Spring Boot promueve la inversión de dependencias a través del uso de inyección de dependencias y la inversión de control. Los componentes de la aplicación dependen de abstracciones en lugar de implementaciones concretas, lo que facilita la sustitución de componentes y la modularidad del sistema.

MongoDB

Al ser MongoBD una base de datos no SQL no se basa en principios SOLID, teniendo en cuenta además que al ser NoSQL no se necesita seguir parámetros o normas para una organización estándar como normalización

7. Atributos de calidad

Arquitectura cliente-servidor

Los atributos de calidad relevantes son:

- Rendimiento: La arquitectura cliente-servidor debe ser capaz de manejar eficientemente las solicitudes de múltiples clientes, garantizando tiempos de respuesta rápidos.
- Escalabilidad: Debe poder escalar para manejar un mayor número de clientes y transacciones sin degradación del rendimiento.
- Seguridad: Debe garantizar la seguridad de la comunicación entre el cliente y el servidor, protegiendo los datos transmitidos.
- Fiabilidad: Debe ser fiable, minimizando las interrupciones del servicio y los fallos del sistema.

Los atributos de calidad de los que tiene problemáticas al cumplir son:

- Escalabilidad: Puede ser difícil escalar verticalmente debido a limitaciones en el hardware del servidor central, lo que puede resultar en cuellos de botella y un rendimiento degradado en momentos de alta demanda.

Next.js

Los atributos de calidad relevantes son:

- Rendimiento: Next.js debe ser capaz de renderizar páginas web de forma rápida y eficiente, mejorando la experiencia del usuario.
- Mantenibilidad: Debe permitir un desarrollo y mantenimiento ágil de aplicaciones web, facilitando la incorporación de nuevas características y correcciones de errores.

Los atributos de calidad de los que tiene problemáticas al cumplir son:

- Escalabilidad: Si no se implementa correctamente, Next.js puede tener dificultades para escalar horizontalmente, lo que podría afectar su capacidad para manejar grandes volúmenes de tráfico.

TypeScript

Los atributos de calidad relevantes son:

- Mantenibilidad: TypeScript mejora la mantenibilidad del código al agregar tipos estáticos, lo que facilita la detección temprana de errores y la comprensión del código.

SOAP

Los atributos de calidad relevantes son:

- Interoperabilidad: SOAP se utiliza para facilitar la comunicación entre sistemas heterogéneos, permitiendo la interoperabilidad entre diferentes plataformas y lenguajes de programación.

Los atributos de calidad de los que tiene problemáticas al cumplir son:

- Sobrecarga: SOAP puede tener una mayor sobrecarga en comparación con otros protocolos de comunicación más ligeros como REST, lo que podría afectar el rendimiento en sistemas con recursos limitados.

Java

Los atributos de calidad relevantes son:

- Portabilidad: Java es conocido por su portabilidad, lo que significa que las aplicaciones desarrolladas en Java pueden ejecutarse en diferentes sistemas operativos sin cambios significativos en el código fuente.
- Seguridad: Java ofrece características de seguridad robustas, como la verificación de código y la gestión de permisos, que ayudan a proteger las aplicaciones contra amenazas.

Los atributos de calidad de los que tiene problemáticas al cumplir son:

- Consumo de recursos: Las aplicaciones Java pueden requerir una cantidad significativa de recursos del sistema, lo que podría ser un desafío en entornos con recursos limitados.

Spring Boot

Los atributos de calidad relevantes son:

- Rendimiento: Spring Boot proporciona un rendimiento eficiente al utilizar un modelo de programación basado en convenciones que reduce la complejidad y mejora el tiempo de desarrollo.
- Mantenibilidad: Spring Boot favorece la mantenibilidad al promover buenas prácticas de desarrollo y proporcionar herramientas que facilitan la gestión y actualización del código.

Los atributos de calidad de los que tiene problemáticas al cumplir son:

- Complejidad: A medida que las aplicaciones Spring Boot crecen en tamaño y complejidad, pueden volverse difíciles de mantener y de entender, especialmente para desarrolladores nuevos en la plataforma.

MongoDB

Los atributos de calidad relevantes son:

- Escalabilidad: MongoDB es conocido por su capacidad de escalar horizontalmente, lo que permite gestionar grandes volúmenes de datos distribuidos en múltiples servidores.
- Fiabilidad: MongoDB ofrece características como la replicación y el balanceo de carga automático, que mejoran la fiabilidad y la disponibilidad del sistema.

Los atributos de calidad de los que tiene problemáticas al cumplir son:

- Consistencia: MongoDB prioriza la disponibilidad y la partición de datos sobre la consistencia, lo que puede llevar a problemas de consistencia en sistemas altamente distribuidos si no se configura correctamente.

8. Ejemplo práctico

Descripción

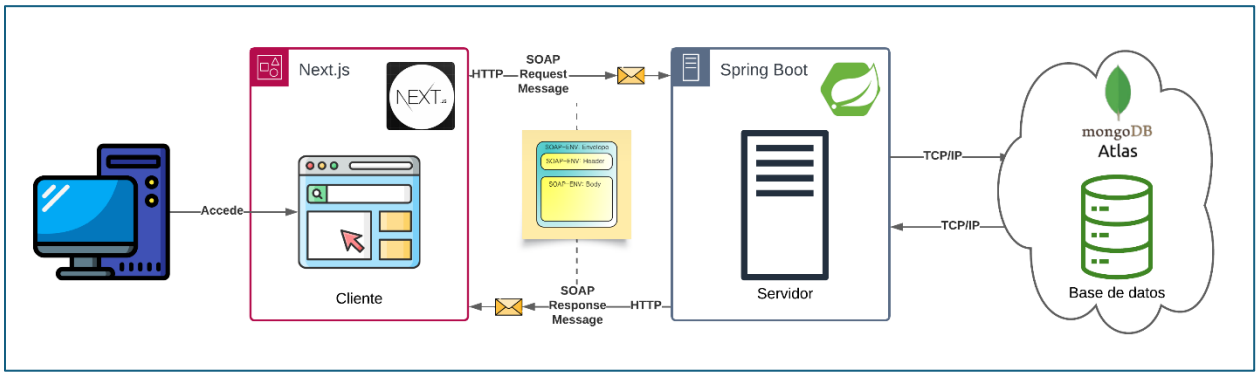
Se desarrolló una página web utilizando la arquitectura cliente-servidor. En términos generales, se implementó un sistema CRUD (Crear, Leer, Actualizar, Eliminar) destinado a gestionar información sobre empleados. Cada entidad de empleado incluye campos como el ID, nombre, correo electrónico, departamento y rol.

En el lado del cliente, se utiliza Next.js con Typescript, una combinación que ofrece un desarrollo rápido y robusto de interfaces de usuario, aprovechando las ventajas de la tipificación estática. Mientras tanto, en el lado del servidor, se empleó Spring Boot junto con Java, una combinación ampliamente utilizada en el desarrollo de aplicaciones empresariales, que proporciona un marco sólido para la construcción de servicios RESTful y aplicaciones web escalables.

Para la comunicación entre el cliente y el servidor, se eligió el protocolo SOAP (Simple Object Access Protocol), conocido por su capacidad para facilitar la interoperabilidad entre diferentes sistemas a través de la definición de interfaces basadas en estándares XML. Esto permite una comunicación estructurada y segura entre los componentes cliente y servidor.

En cuanto a la gestión de datos, se seleccionó MongoDB Atlas como la solución de base de datos. MongoDB Atlas es una base de datos NoSQL en la nube, que ofrece escalabilidad, flexibilidad y alta disponibilidad para almacenar y gestionar datos de forma eficiente.

Diagrama de alto nivel



Referencias

Next.js – Vercel. (s. f.). Vercel. https://vercel.com/blog/next

Santis, N. (2020, 27 octubre). *Lanzamiento de Next.js 10 | Nikolas Santis Software Developer*. Nikolas Santis | Desarrollador de Software. https://nikosantis.dev/blog/nextjs-10

Sánchez, D. (2021, 16 junio). *¿Qué hay de nuevo en Next.js?* DEV Community. https://dev.to/d4vsanchez/que-hay-de-nuevo-en-next-js-1a14

Rosenwasser, D. (2022, 1 octubre). *Ten Years of TypeScript - TypeScript*. TypeScript. https://devblogs.microsoft.com/typescript/ten-years-of-typescript/

2023 State of the API Report | API Technologies. (s. f.). Postman API Platform. https://www.postman.com/state-of-api/api-technologies/

Box, D. (2001, 4 abril). *A Brief History of SOAP*. https://www.xml.com/pub/a/ws/2001/04/04/soap.html

History of Java - Javatpoint. (s. f.). www.javatpoint.com. https://www.javatpoint.com/history-of-java

IBM documentation. (s. f.). https://www.ibm.com/docs/es/rsas/7.5.0?topic=standards-soap

JavaScript with syntax for types. (s. f.). https://www.typescriptlang.org/es/

Kinsta. (2022, 19 diciembre). *¿Qué es Next.js? Un Vistazo al Popular Framework de JavaScript*. Kinsta®. https://kinsta.com/es/base-de-conocimiento/next-js/

NextJS: ¿el futuro de la web?¿por qué y cuándo usarlo? - Aplyca. (2022, 14 septiembre). Aplyca Tecnología SAS. https://www.aplyca.com/blog/nextjs-el-futuro-web-que-es-nextjs

Simple Object Access Protocol overview. (s. f.). https://docs.oracle.com/cd/A97335_02/integrate.102/a90297/overview.htm

Stack Overflow Developer Survey 2020. (s. f.-a). Stack Overflow. https://insights.stackoverflow.com/survey/2020#most-loved-dreaded-and-wanted

State of JS 2020: JavaScript Flavors. (s. f.). https://2020.stateofjs.com/en-US/technologies/javascript-flavors/

A Short History of MongoDB. (s. f.). https://petedejoy.com/writing/mongodb

Admin. (2019, 9 diciembre). *Un poco de historia sobre Spring - Java desde 0*. Java Desde 0. https://javadesde0.com/un-poco-de-historia-sobre-spring/

Akshit Madan. (2023, 30 agosto). *Client-Server Model | Software Development Patterns | System Design Basics* [Vídeo]. YouTube. https://www.youtube.com/watch?v=JenSBaNpcXM

Bandara, D. (2022, 6 enero). Evolution of Client-Server Architecture and Web-servers. *Medium*. https://damsak.medium.com/evolution-of-client-server-architecture-and-web-servers-cd58f60436f7

Caules, C. Á. (2023, 26 diciembre). *¿Qué es Spring Boot?* Arquitectura Java. https://www.arquitecturajava.com/que-es-spring-boot/

Client-Server Architecture | EN.601.421: Object-Oriented Software Engineering (OOSE). (s. f.). https://darvishdarab.github.io/cs421_f20/docs/readings/client_server/

En pocas palabras, la historia del marco Spring y Spring Boot - programador clic. (s. f.). https://programmerclick.com/article/1902668891/

Gautam, S. (s. f.). *Client Server Architecture*. https://www.enjoyalgorithms.com/blog/client-server-architecture

GfG. (2022, 2 diciembre). *Client-Server model*. GeeksforGeeks. https://www.geeksforgeeks.org/client-server-model/

H, E., & H, E. (2023, 29 octubre). *La evolución de Spring Boot: características y novedades*. Codigo Vanguardia. https://codigovanguardia.com/desarrollo-web/la-evolucion-de-spring-boot-caracteristicas-y-novedades/

History of MongoDB. (s. f.). https://www.w3schools.in/mongodb/history

Japan, I. P. S. O. (s. f.). *Brief History-Computer Museum*. Information Processing Society Of Japan. https://museum.ipsj.or.jp/en/computer/unix/history.html#:~:text=Client%2Dserver%20systems%20began%20to,multiple%20workstations%20or%20personal%20computers.

Klimov, P. (2022, 14 agosto). A brief history of the origin of the Spring Boot with a short example of spring boot starters. *Medium*. https://medium.com/@klimovpm/the-short-story-of-spring-boot-and-what-its-starters-750f27d715d

Mesa, N. V. N. (2021, 8 septiembre). El mundo del Spring Boot. *Portal de Noticias de Tecnología, Realidad Virtual, Aumentada y Mixta, Videojuegos*. https://niixer.com/index.php/2020/11/20/el-mundo-del-spring-boot/

MongoDB. (s. f.-a). *Acerca de nosotros - nuestra historia | MongoDB*. https://www.mongodb.com/es/company

MongoDB. (s. f.-b). *MongoDB Evolved – Version History*. https://www.mongodb.com/evolved

Srikanth_V. (2023, 26 julio). History of Cloud Computing - Srikanth_V - Medium. *Medium*. https://medium.com/@vaddenenisrikanth/history-of-cloud-computing-9b5d865c0021

Thevathas, D. A. (2022, 10 junio). MonGoDB History - Dilakshan Antony Thevathas - Medium. *Medium*. https://atdilakshan.medium.com/mongodb-history-11649c6b3d7c

Константин. (2021, 28 febrero). *Lo que necesita saber sobre Spring: historia, módulos clave, comparación con Java EE*. JavaRush. https://javarush.com/es/groups/posts/es.3546.lo-que-necesita-saber-sobre-spring-historia-mdulos-clave-comparacin-con-java-ee

Varnas, M. (2023, 2 octubre). *Monthly database popularity rankings determined through extensive data analysis*. Red9. https://red9.com/database-popularity-ranking/

Delgado, D. O. (2023, 4 mayo). Empresas que usan MongoDB. *OpenWebinars.net*. https://openwebinars.net/blog/empresas-que-usan-mongodb/

Lista de empresas que usan Spring Boot (30,248) | TheirStack.com. (s. f.). TheirStack.com. https://theirstack.com/es/technology/spring-boot

Ashraf, S. (2023, 26 diciembre). *Top 10 Startups using Spring Boot that you should know about*. Back4App Blog. https://blog.back4app.com/startups-using-spring-boot/