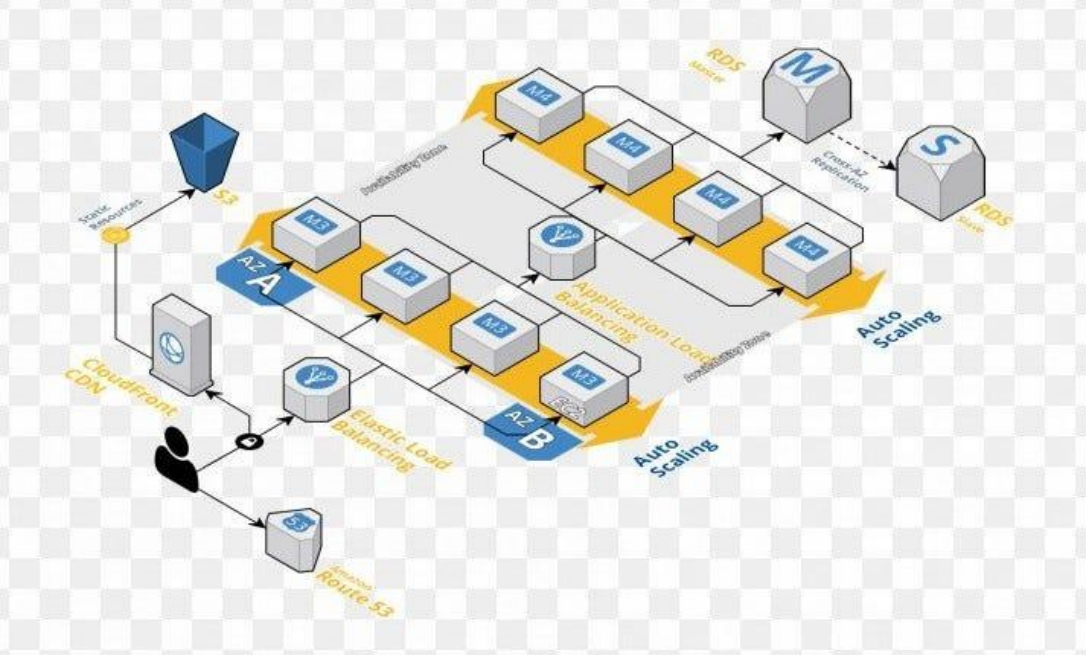
Jhon Alexander Corredor Medina

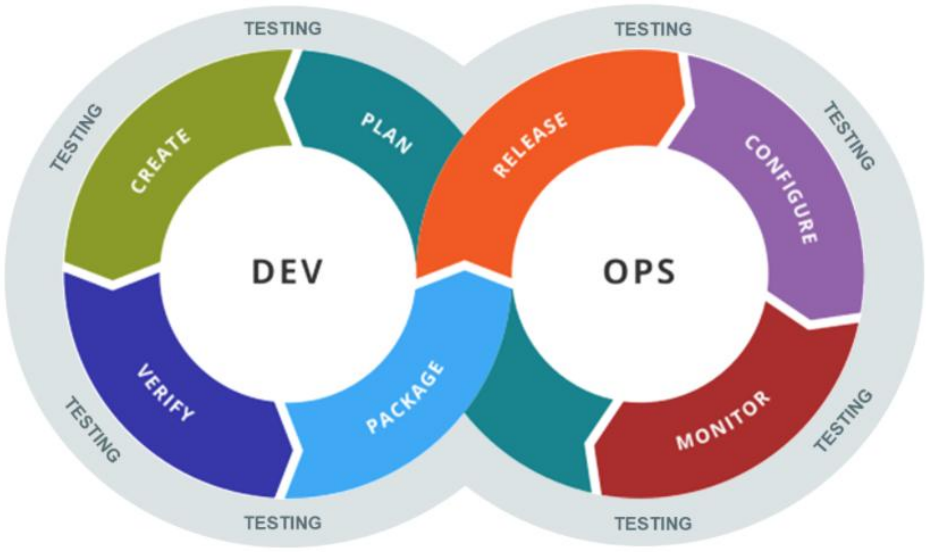
Soy un joven de 18 años que actualmente estudia en el Servicio Nacional de Aprendizaje (SENA), donde curso un tecnólogo en análisis y desarrollo de software. La tecnología siempre ha sido una de mis mayores pasiones, y me he enfocado especialmente en el desarrollo web, lo que me ha permitido convertirme en un desarrollador fullstack con experiencia tanto en frontend como en backend. En el ámbito backend, tengo sólidos conocimientos en Java y C#, y he trabajado en la gestión y diseño de bases de datos, asegurando que sean eficientes y bien estructuradas. Por otro lado, en el desarrollo frontend, manejo frameworks modernos como Vue y Angular, con los que he creado interfaces dinámicas, atractivas y responsivas. Mi dominio de tecnologías como HTML, CSS y JavaScript me permite diseñar interfaces de usuario que no solo son visualmente agradables, sino también funcionales y centradas en la experiencia del usuario. Estoy comprometido con seguir desarrollando mis habilidades y contribuyendo al desarrollo de soluciones tecnológicas innovadoras, siempre buscando mejorar y adaptarme a las nuevas tendencias del mercado.

Revisión de elementos conceptuales para la representación de las arquitecturas de referencias de software



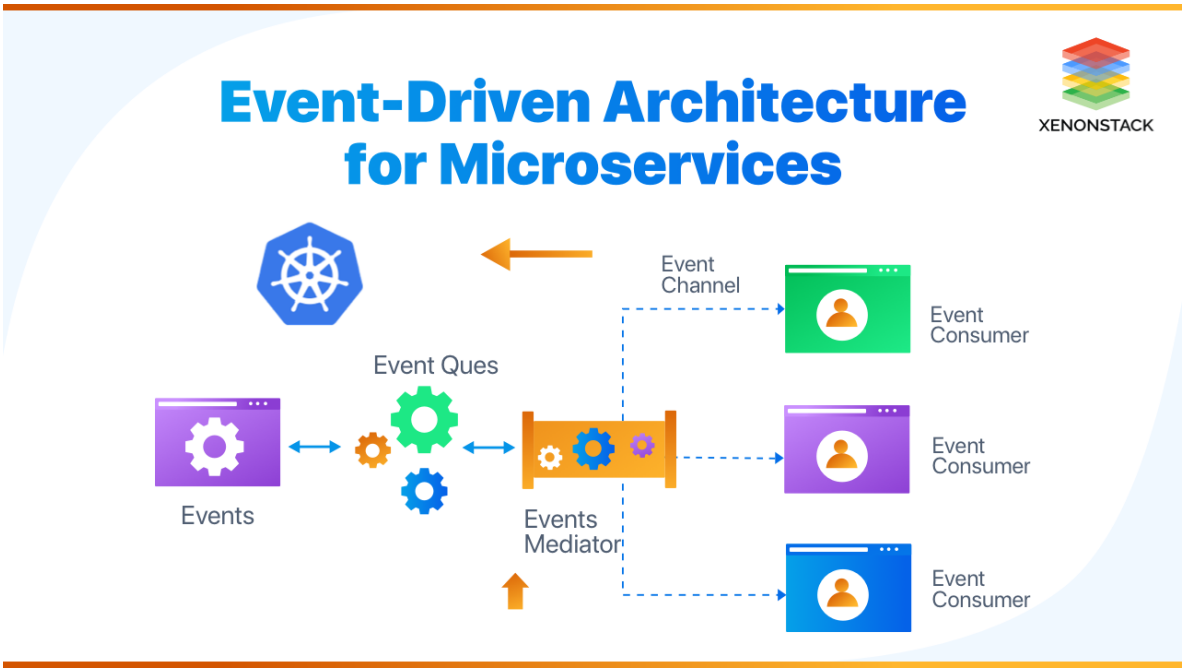
El artículo examina en profundidad las diferentes formas de representar arquitecturas de software, destacando la importancia de comprender cómo se descomponen y definen sus elementos esenciales para lograr una representación clara y coherente. En particular, se exploran y analizan los métodos más utilizados para describir dichas arquitecturas, poniendo énfasis en el uso del lenguaje natural como una herramienta común y accesible en este ámbito. El lenguaje natural permite una comunicación más sencilla y comprensible entre los involucrados en el proceso de desarrollo de software, lo que lo convierte en una opción popular. No obstante, se señala que, a pesar de sus ventajas en términos de accesibilidad, este enfoque presenta limitaciones significativas cuando se trata de realizar un análisis más riguroso y detallado de las arquitecturas de software. En especial, el lenguaje natural puede carecer de la precisión y la estandarización necesarias para representar con exactitud todos los aspectos técnicos y complejos que involucra una arquitectura. Esto puede generar ambigüedades o interpretaciones incorrectas, lo que dificulta la implementación de soluciones eficientes. Por ello, el artículo sugiere que, aunque el lenguaje natural es útil en las primeras fases del diseño, es fundamental complementarlo con otras representaciones más formales o técnicas que permitan un análisis más profundo y riguroso de los sistemas. De este modo, se puede lograr una mejor estandarización en la descripción de arquitecturas de software, facilitando tanto su desarrollo como su posterior mantenimiento.

Arquitectura basada en Microservicios y DevOps para una



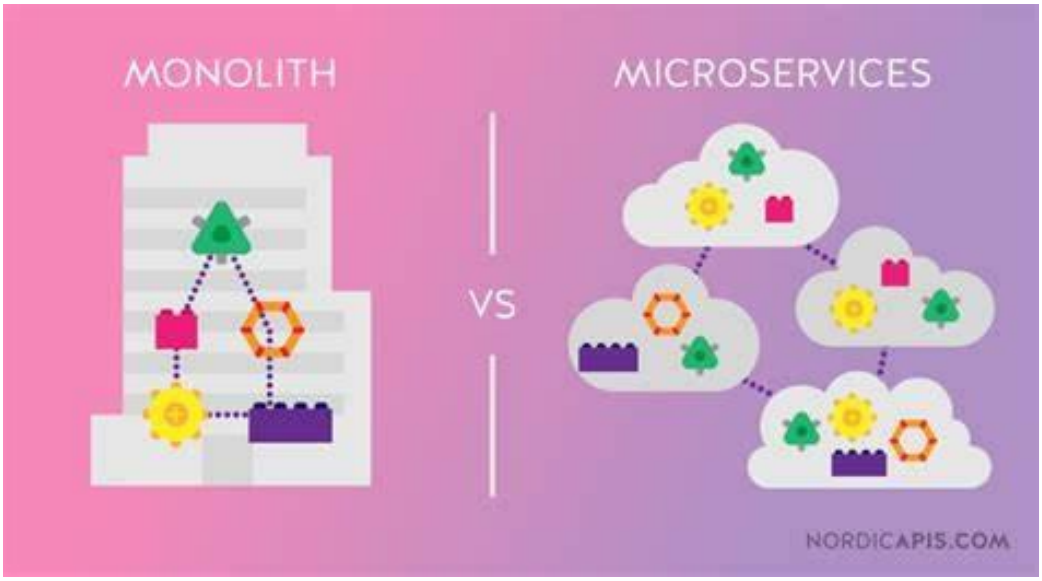
El desarrollo del proyecto SIGAP tuvo como objetivo principal implementar una arquitectura basada en microservicios y principios DevOps, con la finalidad de optimizar los procesos de ingeniería de software y, al mismo tiempo, aumentar significativamente la productividad del equipo de desarrollo. La elección de esta arquitectura estuvo motivada por la necesidad de realizar entregas de software de manera más ágil, progresiva y controlada, permitiendo a los equipos de trabajo adaptar sus flujos de trabajo a un entorno de desarrollo más eficiente. Una de las características más destacadas de la arquitectura de microservicios es su capacidad de descomponer una aplicación en servicios independientes y especializados, lo que facilita su mantenimiento y escalabilidad. Por su parte, la integración de DevOps busca cerrar la brecha entre el desarrollo y las operaciones, promoviendo una cultura de colaboración, automatización y mejora continua, lo que resulta en ciclos de entrega más rápidos y confiables. A lo largo del proyecto, se evidenció que esta combinación no solo mejora la calidad del software, sino que también permite una mayor flexibilidad para adaptarse a los cambios y requerimientos del cliente. Además, la arquitectura propuesta demostró ser altamente adaptable a otros dominios fuera del ámbito de SIGAP, permitiendo su aplicación en sectores que requieren una gestión continua de software, como los sistemas empresariales o aplicaciones orientadas al servicio. La experiencia obtenida a través de SIGAP reafirma que este enfoque arquitectónico no solo es viable, sino también escalable para proyectos futuros, proporcionando una base sólida para la evolución del software en diversas áreas.

Diseño e implementación de una arquitectura de microservicios orientada a trabajar con transacciones distribuidas



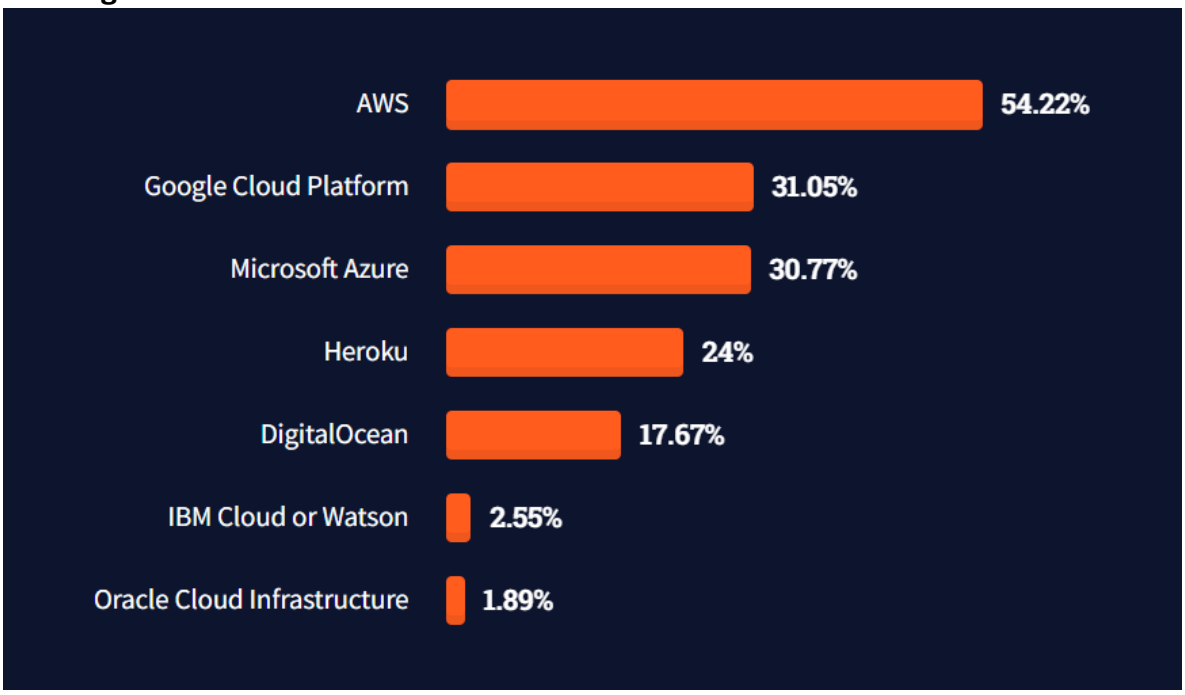
En la investigación se analizó el problema de transacciones fallidas durante la implementación de una arquitectura de microservicios, un desafío común debido a la naturaleza distribuida de estos sistemas. En arquitecturas tradicionales, las transacciones suelen gestionarse de manera centralizada, pero en un entorno de microservicios, donde cada servicio opera de manera independiente, lograr una transacción distribuida puede ser complejo. Las fallas en estas transacciones pueden generar inconsistencias en los datos y afectar la estabilidad del sistema. La investigación aborda esta problemática proponiendo una solución basada en un patrón de eventos, que permite coordinar las operaciones entre distintos microservicios de manera más eficiente. Además, se sugiere el uso de tecnologías en la nube, que proporcionan herramientas especializadas para manejar transacciones distribuidas de forma robusta y escalable. Estas tecnologías permiten que cada microservicio maneje sus propias operaciones mientras se sincronizan los eventos a través de un sistema de mensajería, lo que garantiza la consistencia de los datos sin sacrificar la independencia de los servicios. La solución propuesta no solo resuelve el inconveniente de las transacciones fallidas, sino que también mejora la escalabilidad y resiliencia del sistema, ofreciendo un enfoque moderno para la implementación de microservicios.

Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web



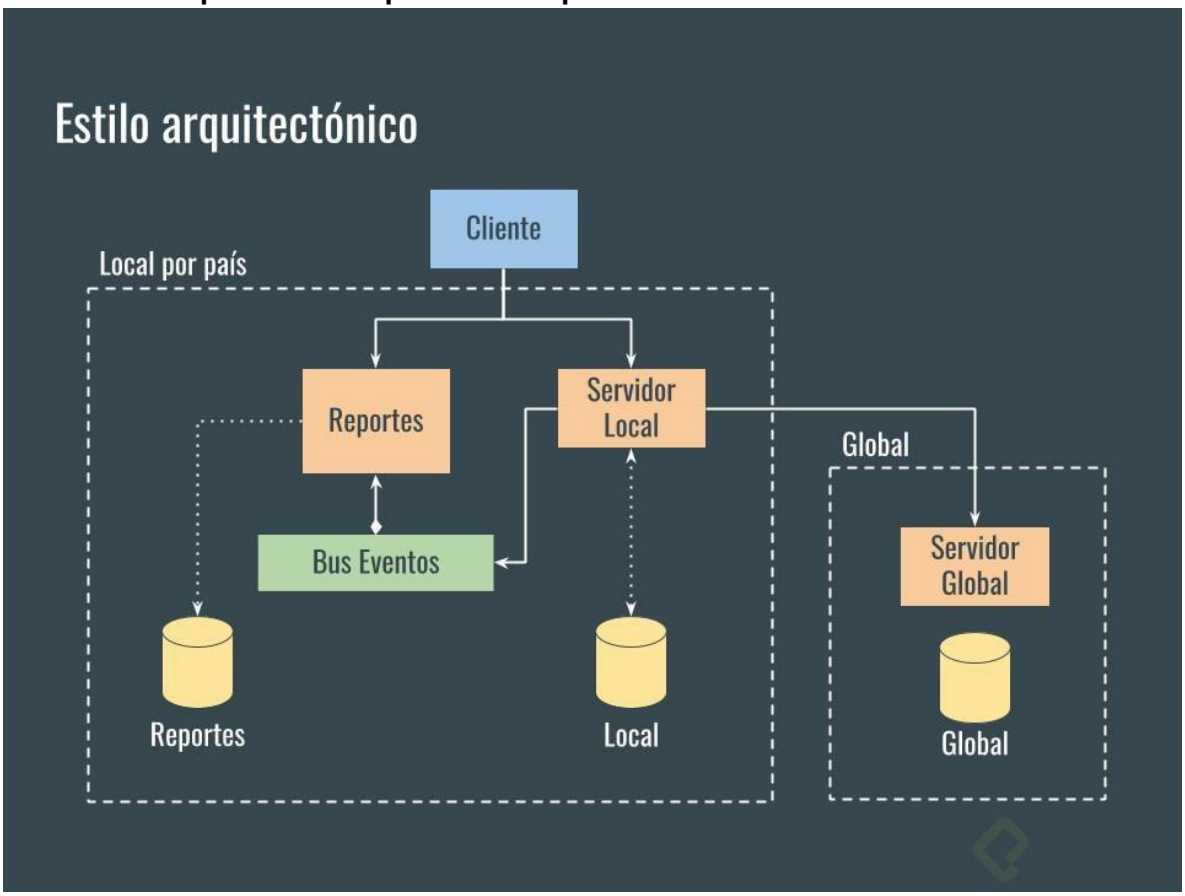
El estudio de la Coordinación General de Tecnologías de la Información y Comunicación de la Asamblea Nacional del Ecuador analiza las tecnologías, metodologías y arquitecturas empleadas en el desarrollo de aplicaciones web, evaluando la viabilidad de implementar la arquitectura de microservicios en contraste con las aplicaciones monolíticas. Las aplicaciones monolíticas, conocidas por su estructura unificada, son más simples de desplegar en sus fases iniciales, pero tienden a enfrentar problemas de escalabilidad y flexibilidad a medida que crecen. Por otro lado, la arquitectura de microservicios ofrece una solución más flexible, dividiendo las aplicaciones en componentes independientes que permiten un desarrollo más ágil y escalable. Cada microservicio puede gestionarse, desplegarse y actualizarse de forma autónoma, lo que facilita la adopción de nuevas tecnologías y mejora la resiliencia del sistema. El estudio resalta la importancia de metodologías como DevOps e integración continua para maximizar los beneficios de esta arquitectura. Aunque se reconocen desafíos como la reestructuración de sistemas y el cambio cultural necesario, la adopción de microservicios en la Asamblea Nacional podría mejorar significativamente la eficiencia y agilidad en la gestión de aplicaciones complejas.

Desarrollo de aplicaciones basadas en microservicios: tendencias y desafíos de investigación



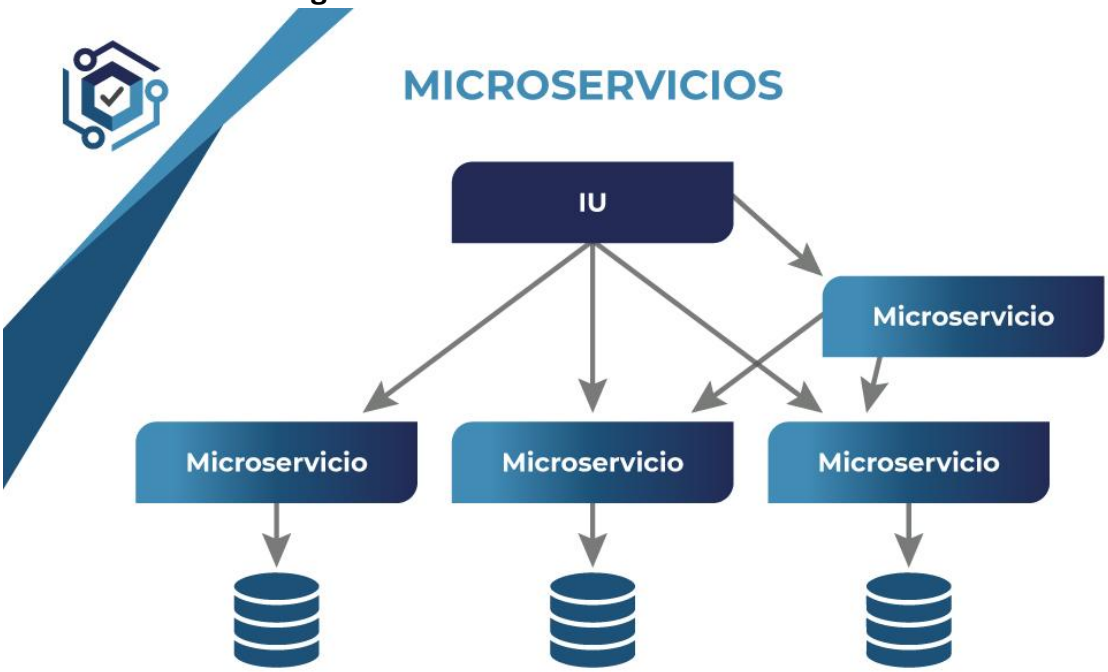
El estudio aborda los desafíos en la fase de desarrollo de aplicaciones basadas en microservicios, destacando las tendencias identificadas en cuanto a las características clave de esta arquitectura, como la modularidad, la seguridad y otros atributos esenciales. La investigación revela que la modularidad de los microservicios permite un desarrollo más ágil y flexible, facilitando la actualización y escalabilidad de los componentes de manera independiente. En términos de seguridad, se observa que, aunque cada microservicio puede tener controles de seguridad propios, la gestión de la seguridad a nivel de sistema completo requiere una planificación más compleja para asegurar la comunicación entre los servicios y proteger los datos sensibles. Además, el estudio enfatiza que la adopción de microservicios implica retos en la coordinación y orquestación de los diferentes módulos, así como en la implementación de buenas prácticas como DevOps y automatización de pruebas para asegurar la integridad del sistema. En general, se concluye que, a pesar de estos desafíos, la arquitectura de microservicios ofrece ventajas significativas en términos de escalabilidad y mantenimiento, siempre que se implementen con las estrategias adecuadas.

Marco de trabajo para la selección de la arquitectura de un proyecto de software mediante la aplicación de patrones arquitectónicos



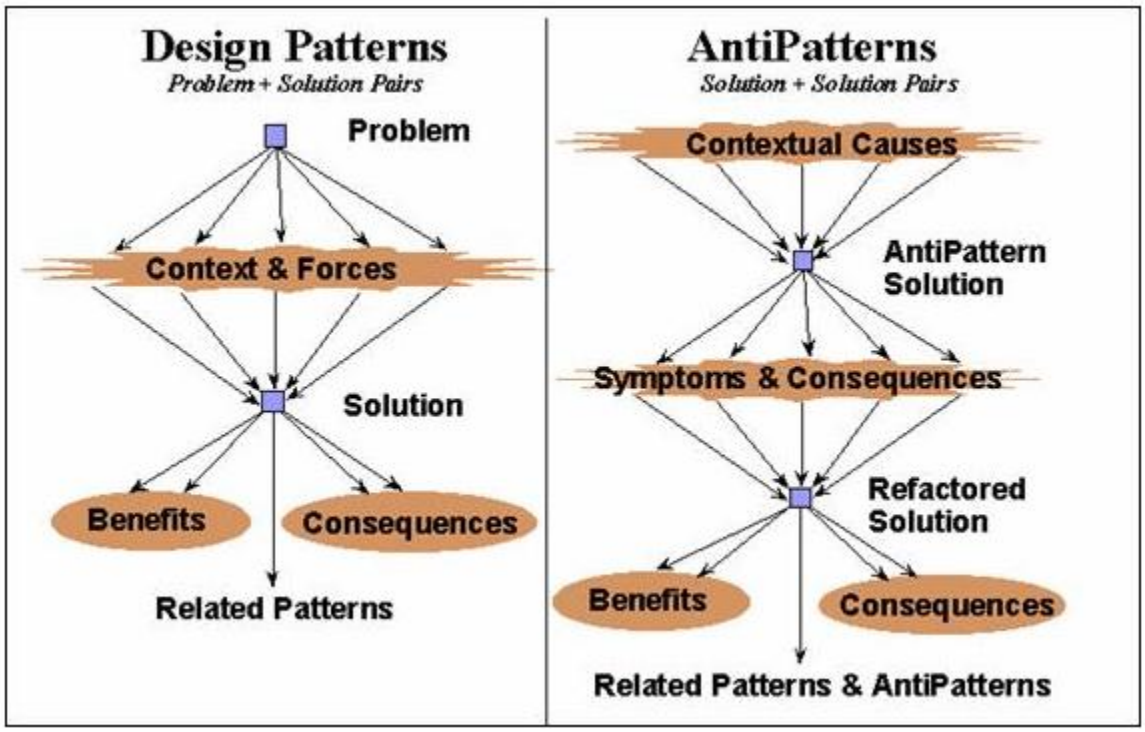
La investigación sobre los patrones de software abarca desde las arquitecturas de información hasta los patrones de navegación, interacción y visualización, con el objetivo de facilitar la selección y aplicación de un patrón arquitectónico adecuado según el contexto y las necesidades del proyecto. Esta exploración permite a los desarrolladores identificar la mejor estructura para organizar y presentar la información, optimizando la experiencia del usuario a través de patrones de navegación eficientes y una interacción fluida. Además, se analizan patrones de visualización que aseguran una presentación clara y coherente de los datos. Al tener una visión integral de estos patrones, se puede tomar decisiones fundamentadas que contribuyan al éxito del desarrollo de la aplicación, alineando la arquitectura con los objetivos funcionales y técnicos del proyecto.

Implementación de una arquitectura de software orientada a microservicios en la Dirección de Tecnología de una Institución Universitaria



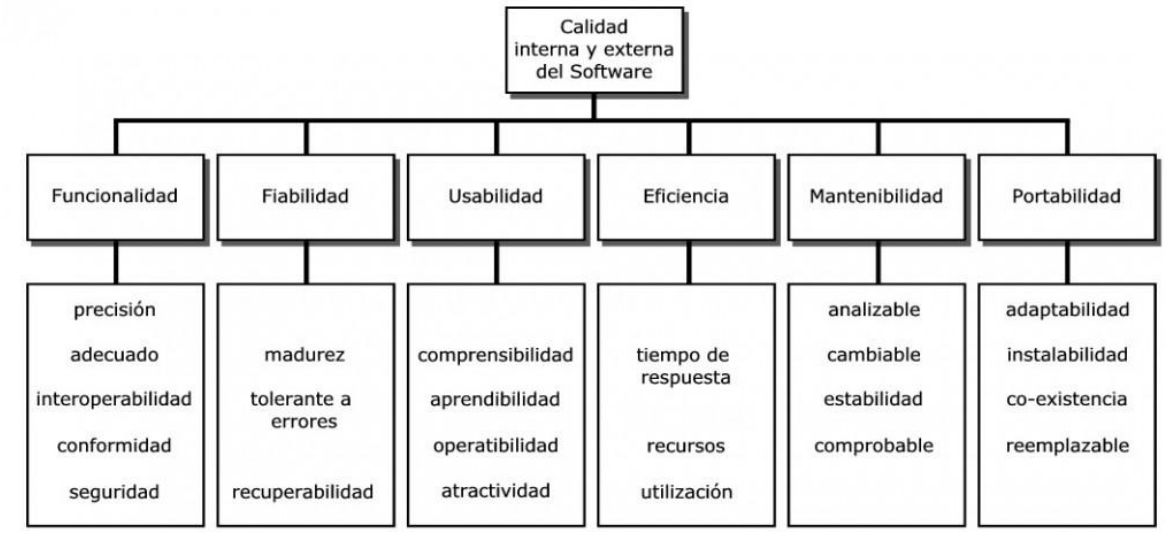
El estudio de propuestas de arquitecturas de software y tecnologías para una institución universitaria se centra en definir e implementar una arquitectura de microservicios, destacando sus ventajas en términos de escalabilidad, flexibilidad y mantenimiento. La investigación analiza cómo esta arquitectura permite descomponer el sistema en servicios independientes que facilitan el desarrollo y la implementación continua. Además, se exploran tecnologías como los micro-frontends, los cuales permiten una integración modular de interfaces, mejorando la experiencia del usuario y la capacidad de actualización sin afectar al sistema completo. El estudio también subraya la importancia del testing automático, que garantiza la calidad del software mediante la automatización de pruebas, reduciendo errores y acelerando el ciclo de desarrollo. En conjunto, estas soluciones tecnológicas apuntan a optimizar la eficiencia y la capacidad de respuesta del sistema, adaptándose a las necesidades dinámicas de la institución universitaria.

Modelo Teórico para la Identificación del Anti-patrón “Stovepipe System” en la Etapa de la Implementación de una Arquitectura de Software



El análisis se enfoca en identificar anti-patrones como el "Stovepipe System", que afectan negativamente el desarrollo de software al generar múltiples adversidades, como sistemas fragmentados y difíciles de integrar. Este tipo de anti-patrón suele resultar en una falta de cohesión entre los módulos del sistema, complicando el mantenimiento y la escalabilidad. La propuesta presentada para abordar este problema sugiere el uso de redes neuronales, ya que estas permiten analizar grandes volúmenes de datos y detectar patrones que puedan indicar la presencia de estos anti-patrones. Mediante el aprendizaje automático, las redes neuronales pueden identificar comportamientos repetitivos y problemáticos en el desarrollo, facilitando la toma de decisiones correctivas a tiempo. Esta solución mejora la calidad del software y ayuda a evitar problemas futuros asociados a la arquitectura del sistema.

Atributos de Calidad y Arquitectura del Software



Este artículo ofrece un análisis detallado de la arquitectura de software, destacando sus conceptos clave y su importancia en el desarrollo de sistemas robustos. Se examinan los atributos de calidad, como escalabilidad, mantenibilidad, flexibilidad y seguridad, y su relevancia en la elección de una arquitectura adecuada según las necesidades específicas del software. Además, se subraya la importancia de modelar la arquitectura desde diferentes perspectivas para garantizar un diseño equilibrado que cumpla con los requisitos de calidad sin sacrificar otros atributos de manera desproporcionada. Con el objetivo de proporcionar una guía clara para seleccionar e implementar la arquitectura más efectiva para cada proyecto.

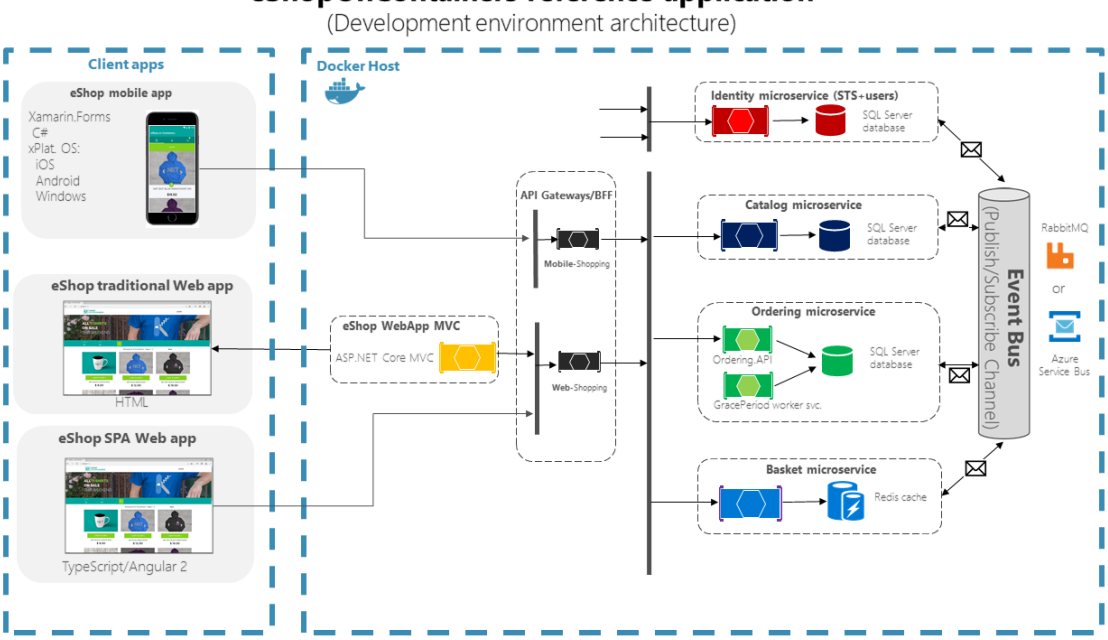
Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles

La investigación explora cómo las arquitecturas de software y las metodologías ágiles se interrelacionan y se integran en el desarrollo de software, analizando cómo ambas pueden colaborar para optimizar el proceso de creación. Se enfoca en el alcance de esta integración, destacando la importancia de considerar los "requisitos significativos para la arquitectura". Estos requisitos son cruciales para garantizar que la arquitectura sea flexible y adaptable a los cambios frecuentes que caracterizan a los entornos ágiles. La investigación busca demostrar cómo una arquitectura bien definida puede facilitar la implementación de metodologías ágiles, permitiendo un desarrollo más eficiente y adaptado a las necesidades cambiantes del proyecto, al mismo tiempo que asegura la calidad y coherencia del software.

Introducción a la Arquitectura de Software

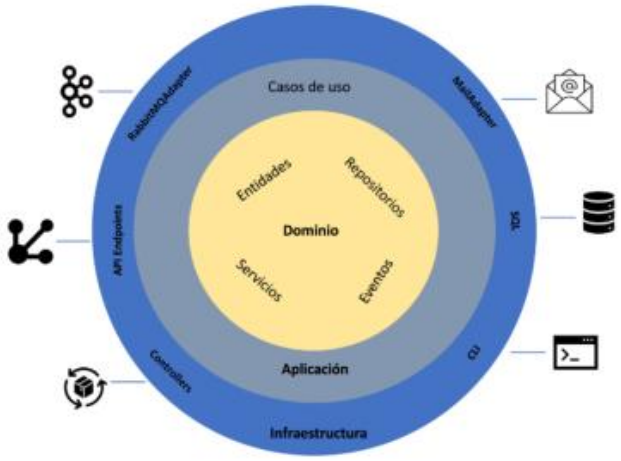
El artículo aborda la arquitectura de software y destaca su importancia en el ciclo de vida del sistema. En él se explora la fase de la arquitectura del sistema, subrayando cómo esta etapa es crucial para definir la estructura y los componentes del software de manera que se alineen con los objetivos del proyecto. Se analiza el ciclo de la arquitectura, que incluye la planificación, el diseño, la implementación y la evaluación continua, destacando cómo cada fase contribuye a la evolución y adaptación del sistema. Además, el artículo enfatiza cómo el entorno en el que opera el sistema y los requerimientos no funcionales, como el rendimiento y la escalabilidad, influyen significativamente en la definición de la arquitectura. Estos factores externos y las necesidades específicas del sistema deben ser considerados cuidadosamente para asegurar que la arquitectura propuesta sea robusta, eficiente y capaz de satisfacer las expectativas y necesidades a lo largo de todo el ciclo de vida del software

Arquitectura basada en microservicios para aplicaciones web



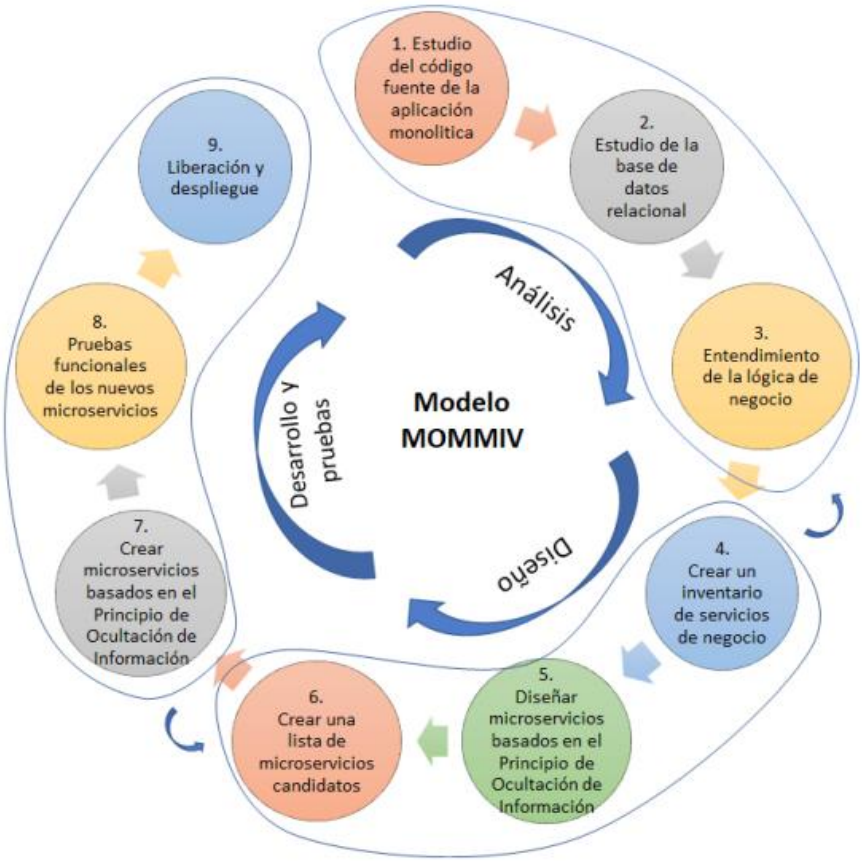
La arquitectura basada en microservicios es un enfoque de diseño de software que descompone aplicaciones complejas en servicios independientes y autónomos, cada uno responsable de una función específica y comunicándose a través de interfaces bien definidas. Entre sus ventajas se encuentran la escalabilidad, la flexibilidad en el despliegue, y la resiliencia, permitiendo que fallos en un servicio no afecten el sistema completo. A diferencia de las aplicaciones monolíticas, que se construyen como una única unidad indivisible, las arquitecturas de microservicios permiten una mayor agilidad en el desarrollo y mantenimiento al permitir a los equipos trabajar en diferentes servicios de manera independiente. Esta separación facilita la adopción de tecnologías variadas y la implementación continua, optimizando así la respuesta a cambios en los requisitos del negocio. En comparación, las aplicaciones monolíticas pueden ser más fáciles de desarrollar inicialmente, pero suelen enfrentar desafíos significativos en términos de escalabilidad y flexibilidad a medida que crecen, lo que puede complicar el mantenimiento y la actualización del sistema.

Herramienta para el modelado y generación de código de Arquitecturas de Software basadas en Microservicios y Diseño guiado por el dominio (DDD)



Se propone implementar una herramienta de modelado y generación de código que se base en principios de microservicios y diseño dirigido por el dominio, con el objetivo de acelerar y facilitar el desarrollo de proyectos de software. Esta herramienta busca optimizar el proceso de creación de aplicaciones al permitir un diseño más modular y flexible, alineado con las necesidades específicas del dominio de cada proyecto. Además, facilita la migración de sistemas heredados a arquitecturas modernas, promoviendo una transición más eficiente y menos disruptiva. La herramienta permitirá a los desarrolladores modelar sistemas complejos de manera intuitiva y generar código automáticamente, reduciendo el tiempo de desarrollo y minimizando errores. Esto se logra al proporcionar una interfaz gráfica que traduce el diseño conceptual en componentes de software reutilizables y fácilmente integrables. La integración de microservicios garantizará que cada componente del sistema sea independiente y escalable, adaptándose mejor a las demandas cambiantes y a la evolución del negocio.

MOMMIV: Modelo para descomposición de una arquitectura monolítica hacia una arquitectura de microservicios bajo el Principio de Ocultación de Información

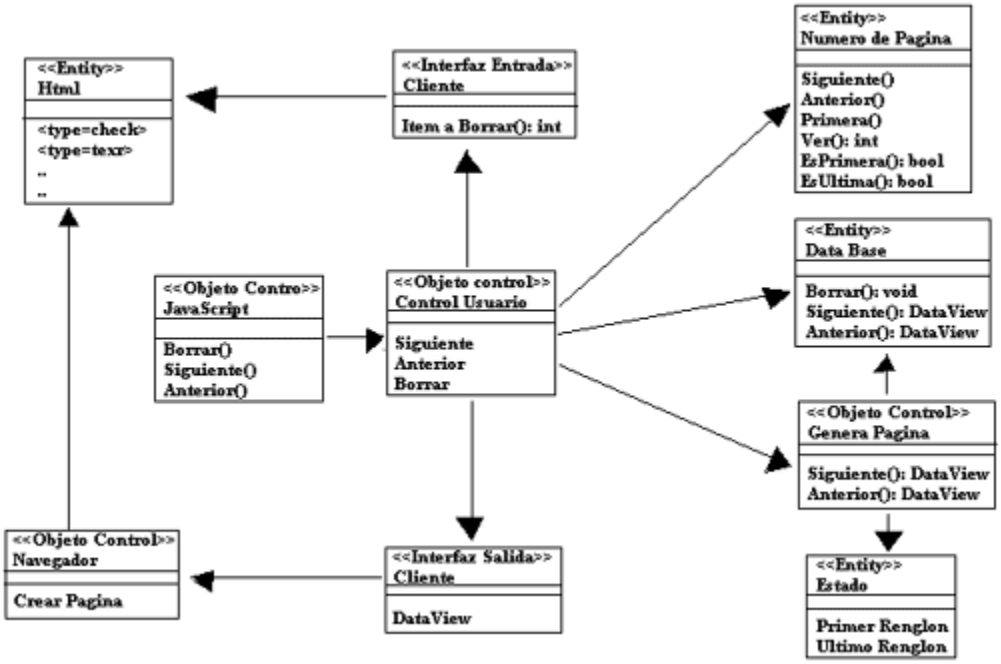


Trata sobre como muchas empresas dependen de aplicaciones legadas, a menudo monolíticas, que se están volviendo obsoletas debido a su falta de flexibilidad y escalabilidad. Estas aplicaciones monolíticas pueden dificultar la adaptación a nuevas tecnologías y demandas del mercado. Una solución efectiva es migrar estas aplicaciones a una arquitectura de microservicios, utilizando el principio de ocultación de información a través del Modelo de Migración o Microservicios Versátil. Este enfoque permite descomponer las aplicaciones monolíticas en servicios independientes que se comunican entre sí mediante interfaces bien definidas, facilitando la modernización y escalabilidad del sistema. El Modelo de Migración asegura que cada microservicio maneje una funcionalidad específica, lo que simplifica el mantenimiento y la evolución del software. Además, al ocultar la complejidad interna y exponer solo las interfaces necesarias, se mejora la modularidad y se facilita la integración con otras aplicaciones y servicios, promoviendo una transición más fluida hacia una arquitectura moderna y eficiente.

Clean architecture para mejorar el desarrollo de aplicaciones móviles en la empresa GMD

En la empresa GMD, debido a la creciente complejidad y expansión de los aplicativos móviles, se llevó a cabo una exhaustiva investigación y análisis de arquitecturas móviles. Como resultado de este análisis, se decidió implementar una Clean Architecture en varios proyectos clave. Esta decisión se basó en la necesidad de mejorar la estructura y organización del código, facilitando su mantenimiento y escalabilidad. La Clean Architecture permite una separación clara entre las diferentes capas del software, promoviendo un diseño más modular y robusto. Al adoptar esta arquitectura, GMD no solo optimizó la calidad y eficiencia del desarrollo, sino que también aseguró que los aplicativos móviles puedan evolucionar y adaptarse con mayor facilidad a los cambios en los requisitos y tecnologías. Esta implementación representa un avance significativo en la forma en que la empresa aborda el desarrollo de aplicaciones móviles, alineándose con las mejores prácticas de la industria.

Arquitectura de software flexible y genérica para métodos del tipo Newton



En la tesis presentada, se realizó una investigación exhaustiva y un análisis detallado de arquitecturas de software con el propósito de desarrollar una propuesta para un software que permita el uso y control de diversos métodos numéricos de Newton. La propuesta de arquitectura se centra en ser flexible y orientada a objetos, para asegurar una implementación robusta y adaptable a diferentes necesidades. Esta arquitectura está diseñada para manejar eficientemente los métodos de Newton, facilitando su integración y aplicación en diferentes contextos. La flexibilidad de la arquitectura permitirá al software adaptarse a futuras ampliaciones y modificaciones, mientras que su enfoque orientado a objetos contribuirá a una estructura más modular y mantenible. La tesis busca no solo optimizar la ejecución de estos métodos numéricos, sino también proporcionar una solución que pueda evolucionar y ajustarse a los avances en el campo del análisis numérico.

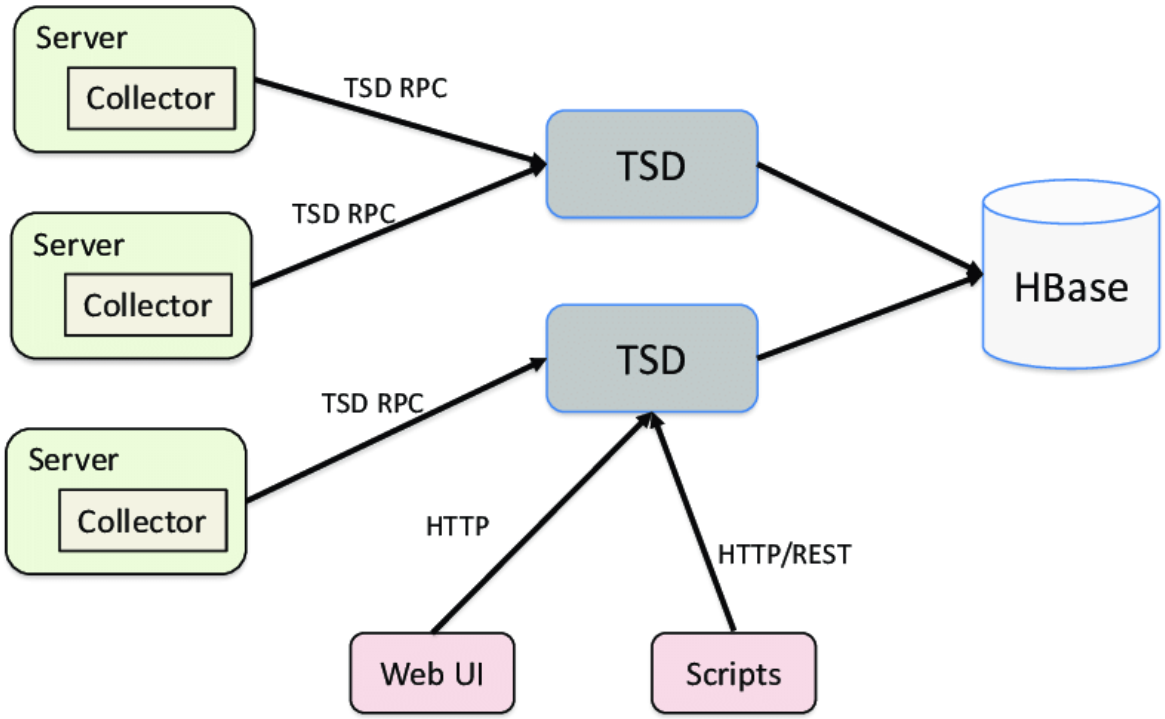
Buenas prácticas en la construcción de software

La guía presentada ofrece un conjunto de consejos clave para lograr un código limpio y de alta calidad durante el desarrollo de software, destacando la importancia de seguir buenas prácticas y adoptar arquitecturas efectivas. A través de recomendaciones sobre metodologías y estilos arquitectónicos, la guía busca proporcionar directrices que mejoren la legibilidad, mantenibilidad y eficiencia del código. Se abordan aspectos fundamentales como la implementación de principios sólidos de diseño, la elección de arquitecturas adecuadas y el empleo de metodologías ágiles que faciliten un desarrollo más organizado y flexible. Además, se enfatiza la importancia de la consistencia en el estilo de codificación y la aplicación de prácticas que promuevan un desarrollo sostenible y colaborativo. La guía pretende ser una herramienta valiosa para desarrolladores que desean optimizar sus procesos de programación y asegurar la calidad de sus proyectos de software.

Adaptación de un patrón de software en seguridad a la arquitectura de un Microservicio

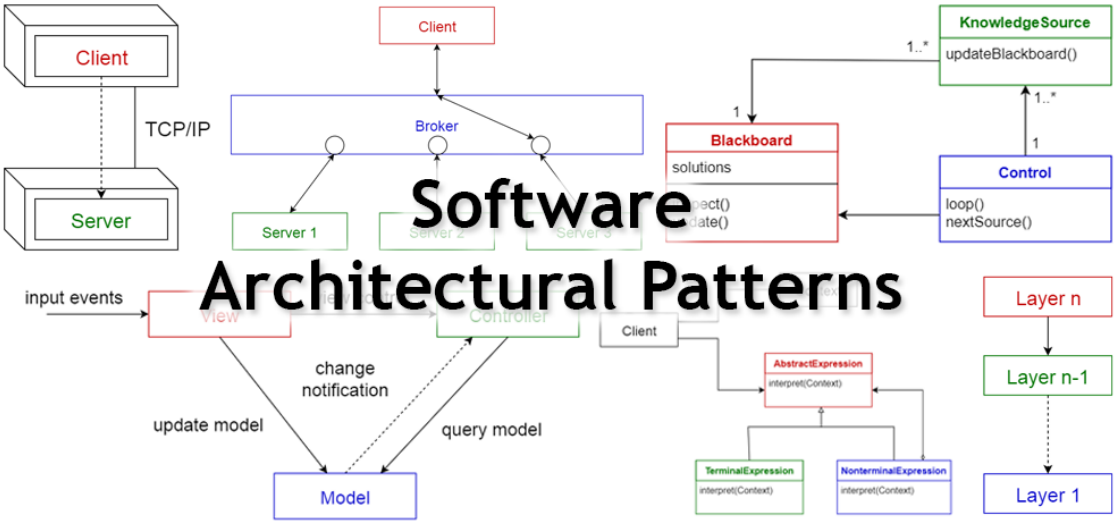
La tesis aborda la problemática de seguridad en el contexto de la arquitectura de microservicios, enfocándose en las vulnerabilidades y desafíos asociados con esta arquitectura distribuida. Para enfrentar estos problemas, se explora el uso e implementación del patrón de seguridad MSPAG (Microservice Security Patterns and Guideline). Este patrón proporciona un conjunto de directrices y estrategias diseñadas para mejorar la seguridad en sistemas basados en microservicios, abordando aspectos críticos como la autenticación, autorización y protección de datos. La investigación se centra en cómo aplicar MSPAG para mitigar riesgos y asegurar una implementación robusta de microservicios. Al seguir estas pautas, se busca establecer prácticas de seguridad efectivas que fortalezcan la protección de las aplicaciones y datos en entornos distribuidos, garantizando una arquitectura de microservicios más segura y confiable.

A scalable architecture for automated monitoring of microservices



El artículo presenta una arquitectura destinada al monitoreo de microservicios, enfocándose en la supervisión de las métricas clave del rendimiento del sistema. La propuesta se basa en el uso de sistemas de monitoreo avanzados como Prometheus, OpenTSDB y Elastic APM para capturar y analizar datos críticos en tiempo real. Prometheus se encarga de la recolección y consulta de métricas, mientras que OpenTSDB se utiliza para almacenar series temporales de datos, y Elastic APM proporciona capacidades de monitoreo y análisis de rendimiento de aplicaciones. La combinación de estas herramientas permite una visibilidad integral sobre el funcionamiento de los microservicios, facilitando la identificación de cuellos de botella y problemas de rendimiento. La arquitectura propuesta tiene como objetivo optimizar el seguimiento de la salud y eficiencia del sistema, mejorando la capacidad de respuesta ante posibles incidencias y asegurando un rendimiento óptimo en entornos distribuidos.

Lenguajes de Patrones de Arquitectura de Software: Una Aproximación Al Estado del Arte



El artículo explora la evolución de la ingeniería de software, enfocándose en los principios fundamentales que han guiado el desarrollo y la maduración del campo. Se examina el estado del arte en los patrones de diseño y lenguajes de patrones, destacando cómo estos conceptos han evolucionado y su aplicación en diversos dominios. La investigación analiza cómo los patrones de diseño han ayudado a estandarizar soluciones para problemas recurrentes en el desarrollo de software, proporcionando marcos y directrices para crear sistemas más eficientes y mantenibles. Además, se discute el impacto de los lenguajes de patrones en la comunicación y documentación de estas soluciones, facilitando su adopción y adaptación en diferentes contextos. El artículo ofrece una visión integral sobre cómo estos principios y herramientas han contribuido a la evolución de la ingeniería de software, mejorando la calidad y la capacidad de adaptación de los sistemas desarrollados.

Reflexión

Al leer diversos artículos, tesis e investigaciones sobre arquitectura de software, sus conceptos, implementaciones en proyectos y buenas prácticas, he comprendido lo importante que es tener en cuenta la arquitectura desde el principio. Antes pensaba que, con tal de que el código fuera funcional, era suficiente para ser un buen desarrollador de software; sin embargo, ahora entiendo que estructurar bien un proyecto marca una gran diferencia. La arquitectura de software ayuda a pensar en cómo interactúan las diferentes partes del sistema y a diseñar estas interacciones de manera que faciliten su mantenimiento y evolución futura. Además, seguir buenas prácticas como la separación de responsabilidades y la creación de código reutilizable no solo ahorra tiempo a largo plazo, sino que también evita que el proyecto se convierta en un caos de código difícil de gestionar. En resumen, tomarse el tiempo para planificar adecuadamente y escribir código de calidad desde el principio permite que el desarrollo sea más fluido y que puedas enfocarte en lo que realmente importa: crear un software funcional, escalable y sostenible.

Bibliografía

Revisión de elementos conceptuales para la representación de las arquitecturas de referencias de software. (2019). *Revista Cubana de Ciencias Informáticas*. http://scielo.sld.cu/scielo.php?pid=S2227-18992019000100143&script=sci_arttext

Arquitectura basada en Microservicios y DevOps para una ingeniería de software continua. (2021). *Revista Iberoamericana de Ciencia, Tecnología y Sociedad*. https://www.redalyc.org/journal/816/81665362014/81665362014.pdf

Diseño e implementación de una arquitectura de microservicios orientada a trabajar con transacciones distribuidas. (2021). *Revista ID Tecnológico*. https://revistas.utp.ac.pa/index.php/id-tecnologico/article/view/3783/4355

Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web. (2017). *Repositorio UDS*. http://138.59.13.30/bitstream/10786/1277/1/93%20Arquitectura%20de%20Software%20basada%20en%20Microservicios%20para%20Desarrollo%20de%20Aplicaciones%20Web.pdf

Introducción a la Arquitectura de Software. (s.f.). https://www.fceia.unr.edu.ar/~mcristia/Introduccion_a_la_Arquitectura_de_Software.pdf

Arquitectura basada en micro-servicios para aplicaciones web. (2020). *Tecnología Investigación y Academia*. https://revistas.udistrital.edu.co/index.php/tia/article/view/13364

Herramienta para el modelado y generación de código de Arquitecturas de Software basadas en Microservicios y Diseño guiado por el dominio (DDD). (2022). *Revista Peruana de Ciencias de la Salud y Sistemas de Información*. https://revistasinvestigacion.unmsm.edu.pe/index.php/rpcsis/article/download/24855/19361/90262

MOMMIV: Modelo para descomposición de una arquitectura monolítica hacia una arquitectura de microservicios bajo el Principio de Ocultación de Información. (2018). *ProQuest Dissertations and Theses*. https://www.proquest.com/docview/2195127731/963E0A5318424FD3PQ/5?accountid=31491&sourcetype=Scholarly%20Journals

Desarrollo de aplicaciones basadas en microservicios: tendencias y desafíos de investigación. (2020). *ProQuest Dissertations and Theses*. https://www.proquest.com/docview/2348878316/963E0A5318424FD3PQ/3?accountid=31491

Marco de trabajo para la selección de la arquitectura de un proyecto de software mediante la aplicación de patrones arquitectónicos. (2021). *Repositorio TDEA*. https://repositorio.tdea.edu.co/bitstream/handle/tdea/923/Sintesis%20Proyecto%20de%20software.pdf?sequence=1&isAllowed=y

Implementación de una arquitectura de software orientada a microservicios en la Dirección de Tecnología de una Institución Universitaria. (2020). *Biblioteca UCASAL*. https://bibliotecas.ucasal.edu.ar/opac_css/index.php?lvl=cmspage&pageid=24&id_notice=73624

Modelo Teórico para la Identificación del Antipatron "Stovepipe System" en la Etapa de la Implementación de una Arquitectura de Software. (2020). *Revista de Postgrado de Ingeniería*. http://revistasbolivianas.umsa.bo/pdf/rpgi/n1/n1_a23.pdf

Clean architecture para mejorar el desarrollo de aplicaciones móviles en la empresa GMD. (2019). *CORE*. https://core.ac.uk/download/pdf/323344533.pdf

Arquitectura de software flexible y genérica para métodos del tipo Newton. (2020). *INAOE Repositorio Institucional*. http://inaoe.repositorioinstitucional.mx/jspui/handle/1009/730

Buenas prácticas en la construcción de software. (2021). *Tecnología Investigación y Academia*. https://revistas.udistrital.edu.co/index.php/tia/article/view/21794

Adaptación de un patrón de software en seguridad a la arquitectura de un Microservicio. (2023). *Repositorio Tecnológico Nacional de México*. http://51.143.95.221/bitstream/TecNM/6574/4/MC_Karen_Monica_Hernandez_Guzman_2023.pdf

A scalable architecture for automated monitoring of microservices. (2019).*ProQuest Dissertations and Theses*. https://www.proquest.com/docview/2172010739/38B8A9878E894A8APQ/1?accountid=31491&sourcetype=Scholarly%20Journals

Lenguajes de Patrones de Arquitectura de Software: Una Aproximación al Estado del Arte. (2008). *Redalyc*.https://www.redalyc.org/pdf/849/84933912003.pdf

Software Architectures - Present and Visions. (2015). *ProQuest Dissertations and Theses*. https://www.proquest.com/docview/1758012314/C6BB69EB2E54D0EPQ/1?accountid=31491&sourcetype=Scholarly%20Journals

Atributos de Calidad y Arquitectura del Software. (2004). *GSISE*. http://www.grise.upm.es/rearviewmirror/conferencias/jiisic04/Tutoriales/tu4.pdf