Resumenes

Hecho por:

Erick Daniel Peña cerleño

Tecnología

Analisis y desarrollo de software

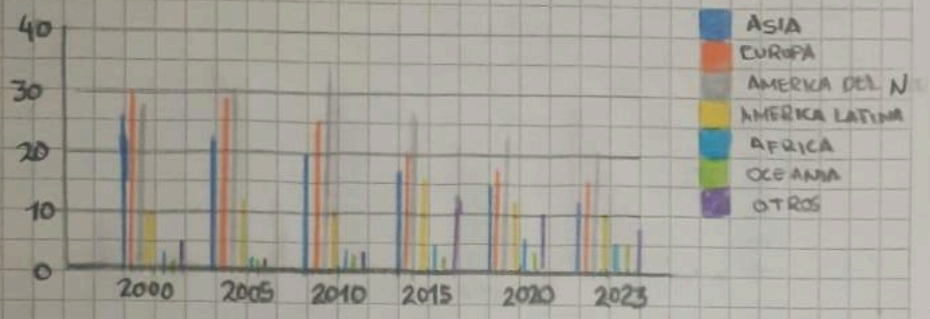
Ficha

2694679

Instructor

Jesús Ariel González Bonilla

Arquitectura de Software



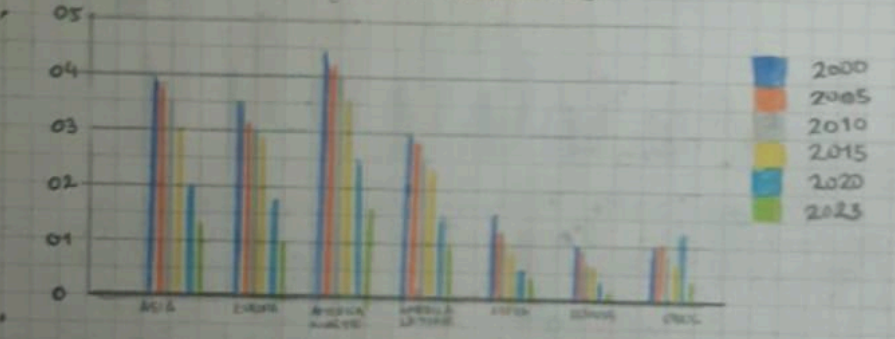
El documento sobre la arquitectura de software profundiza en la importancia de las arquitecturas de referencia de software (ARS) en el desarrollo de sistemas complejos. Las ARS establecen un marco común que facilita la colaboración entre desarrolladores al definir un conjunto de patrones, principios y lineamientos que guían la construcción del software. Estos patrones no solo manejan la coherencia y eficiencia en el desarrollo, si no que tambien aseguran que el software sea escalable y sostenible a largo plazo.

Uno de los puntos clave es la utilización de lenguajes de descripción arquitectónica que ayudan a los ingenieros a documentar y comunicar el diseño del sistema. Entre los lenguajes mencionados están Wright, Darwin y C2, cada uno con características que les permiten representar diferentes aspectos de la arquitectura. Estos lenguajes son esenciales para documentar cómo interactúan los componentes y cómo se manejan las dependencias y conexiones entre ellos.

El texto tambien aborda la relación entre la arquitectura empresarial y los sistemas distribuidos, señalando la necesidad de marcos de referencia solidos para garantizar que estos sistemas complejos puedan integrarse de manera eficiente y soporten las demandas de escabilidad y rendimiento.

comunica con otros servicios atraves de interfaces bien definidas, como APIs. Esta modularidad facilita el desarrollo agil, ya que permite a los equipos de trabajo encargarse en mejorar y actualizar partes del sistema sin afectar el resto de la aplicación. Ademas, los microservicios pueden usar tecnologias, lenguajes de programacion y bases de datos diferentes segun las necesidades de cada componente, lo que proporciona una mayor flexibilidad tecnologica.

Arquitectura Monolitica



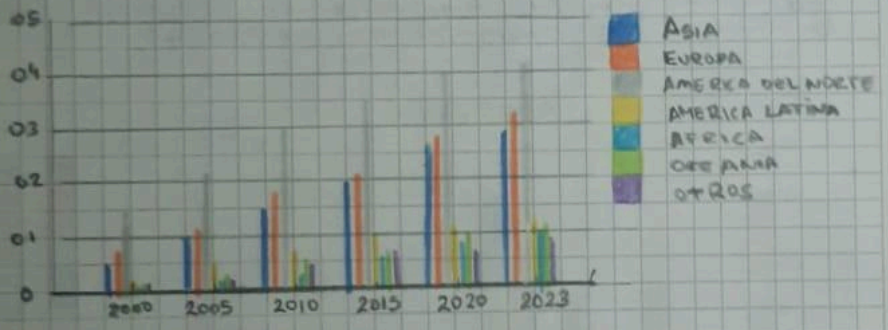
El articulo sobre la arquitectura monolatica analiza los beneficios y riesgos asociados a la migracion de un sistema monolitico a una arquitectura de microservicios. La arquitectura monolitica tradicional consiste en un solo código base que abarca todas las funcionalidades de una aplicación. Si bien es mas sencilla de gestionar en etapas iniciales de desarrollo, a medida que la aplicacion crece, surgen desafios relacionados con la escalabilidad, mantenimiento y velocidad de desarrollo.

Uno de los principales problemas con una arquitectura monolitica es que cualquier cambio o mejora, incluso en una parte pequeña del sistema, requiere que se despliegue y pruebe todo el sistema completo. Este enfoque puede llevar a tiempos de inactividad y errores operativos, ya que las actualizaciones menores pueden impactar a toda la aplicación.

de forma aislada sin afectar el resto de la aplicación.

Esta arquitectura también está muy alineada con los principios del diseño dirigido por el dominio (DDD), que se enfoca en modelar el software en torno al negocio y las necesidades del dominio en lugar de las limitaciones tecnológicas. Al combinar la arquitectura Hexagonal con DDD, Hexa Game logra una estructura de software robusta y flexible que facilita tanto el desarrollo como el mantenimiento a largo plazo.

Event-Sourcing

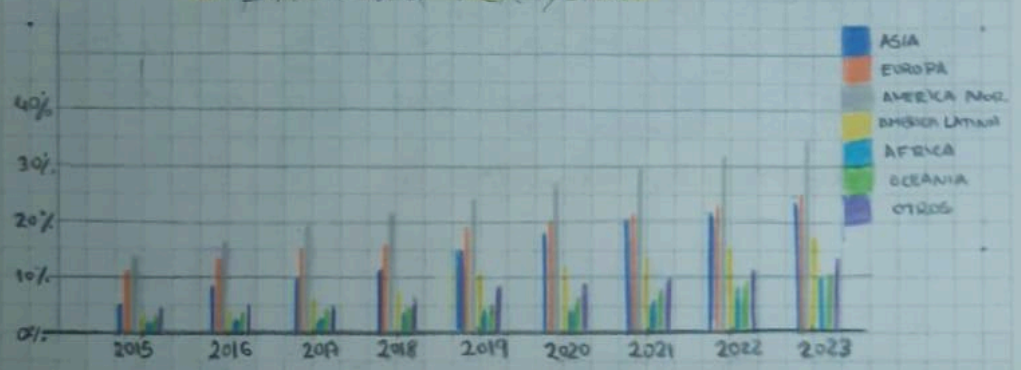


El trabajo analiza el desarrollo de una aplicación que utiliza los patrones de diseño Event-Sourcing y CQRS, destacando como estos patrones permiten una gestión eficientes de eventos en sistemas complejos. El objetivo principal es implementar una estructura desacoplada que se separe claramente el tratamiento, almacenamiento y gestión de los eventos, lo que aporta flexibilidad y escalabilidad a la aplicación.

En el subsistema de escritura, los eventos son capturados por componentes llamados Event-Sourcing Handlers y son almacenados en una base de datos especializada, conocida como Event Store. Este almacenamiento basado en eventos permite que cada cambio en el

A medida que la aplicación se vuelva más compleja, el monolítico tiende hacer menos flexible, y su mantenimiento puede volverse cada vez más costoso y propenso a errores.

Arquitectura Hexagonal

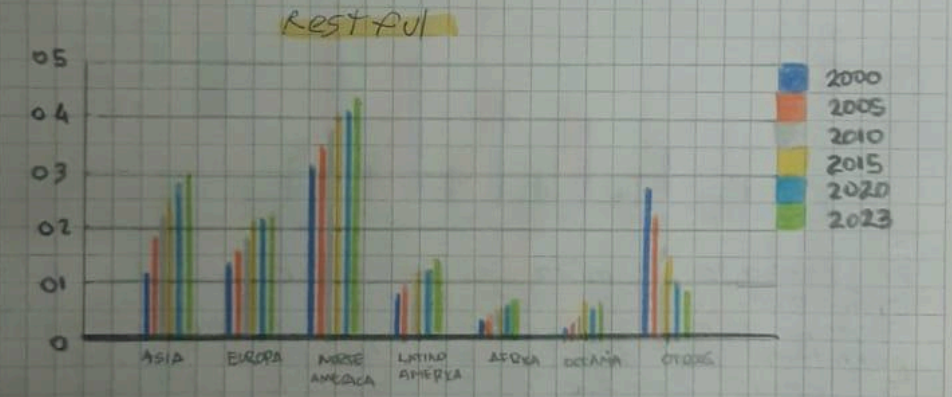


El artículo sobre Hexa Game detalla una plataforma en línea diseñada para coleccionistas de videojuegos, desarrollada utilizando AngularJS en el frontend y SpringBoot en el backend. La aplicación tiene como objetivo facilitar la gestión de colecciones de videojuegos, permitiendo a los usuarios llevar un control detallado de sus juegos, comparar precios y crear listas de deseos. Además, hay planes para expandir la plataforma a dispositivos móviles utilizando Swiftui para iOS y Kotlin para Android, lo que amplíara su alcance y accesibilidad.

Un aspecto clave del desarrollo de Hexageme es su fundamento a la arquitectura Hexagonal, también conocida como arquitectura de puertos y adaptadores. Este enfoque arquitectónico es muy valorado por su capacidad para separar las preocupaciones dentro de una aplicación, promoviendo una independencia entre el núcleo de la lógica de negocio y las tecnologías externas, como bases de datos, interfaces de usuario o API externas. Este principio de separación permite que el sistema sea más flexible y modular, ya que los componentes pueden desarrollarse y modificarse

estado del sistema se guarde como un evento separado, lo que facilita la reconstrucción del estado completo del sistema en cualquier momento. Cada vez que ocurre una acción, como la creación o la modificación de un anuncio inmobiliario, se genera un evento que se persiste en el Eventstore.

El proyecto está contextualizado en un portal inmobiliario, donde se gestionan eventos relacionados con la creación, modificación y eliminación de anuncios de propiedades. Este entorno es ideal para aplicar Event-Sourcing, ya que los sistemas inmobiliarios suelen tener un volumen elevado de cambios y actualizaciones de tiempo real, y es crucial mantener un historial detallado de todas las operaciones realizadas.



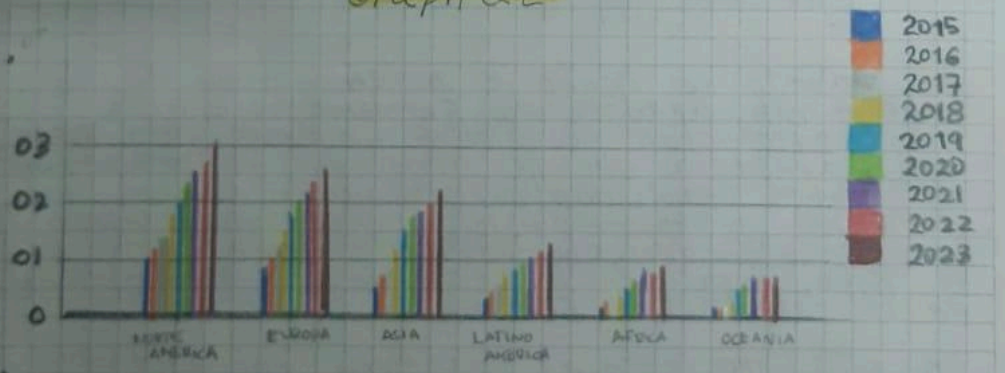
El proyecto House Finder es una innovadora aplicación web diseñada para facilitar la búsqueda de inmuebles en las principales ciudades de colombia.

Una de las tecnologías clave detras de House Finder, es la arquitectura RESTful. Restful es un estilo arquitectónico ampliamente utilizado para diseñar interfaces de programación de aplicaciones que facilitan la interacción entre sistemas atraves de la web. Esta arquitectura aprovecha el protocolo HTTP para la

la comunicación y utiliza formatos de intercambio de datos como JSON y XML. Gracias a Restful es posible desarrollar Api que sean escalables, eficientes y seguras, permitiendo una interacción continua y confiable entre distintas aplicaciones.

En colombia, Restful es un desarrollo de sistemas que requiere compartir datos de manera eficiente y segura. House Finder utiliza esta arquitectura para integrar información proveniente de las principales plataformas inmobiliarias del país, como metro cuadrado, Finca Raiz y ciencuadras. Mediante el uso de estas Apis Restful, la aplicación extrae, procesa y presenta las mejores ofertas de inmuebles de manera automatizada. Esto no solo agiliza el proceso de busqueda para los usuarios, si no que tambien, garantiza que obtengan la información más actualizada y revelante en tiempo real.

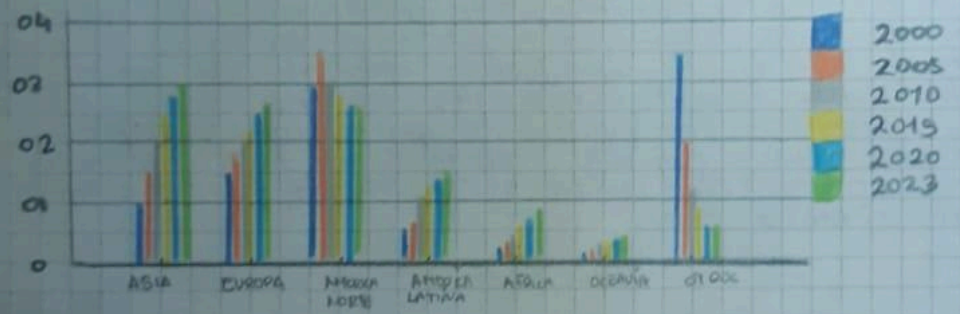
Graph QL



El artículo se enfoca en el desarrollo de una aplicacion backend para la gestion de transferencias utilizando tecnologías modernas como microservicios, Graphql, Rest y contenedores Docker, integrado en una arquitectura DevOps. Este trabajo refleja la importancia creciente de los enfoques modernos en la creaciones de aplicaciones que optimicen tanto la flexibilidad como la escalabilidad en entornos de producción.

El proyecto también aborda la implementación de microservicios, un enfoque arquitectónico en el que una aplicación se divide en pequeños servicios independientes que se comunican entre sí. Al utilizar contenedores Docker, estos microservicios pueden ser desplegados y escalados de manera más eficiente, fomentando una arquitectura ágil y compatible con las prácticas DevOps, que integran el desarrollo y la operación en un ciclo continuo de mejora.

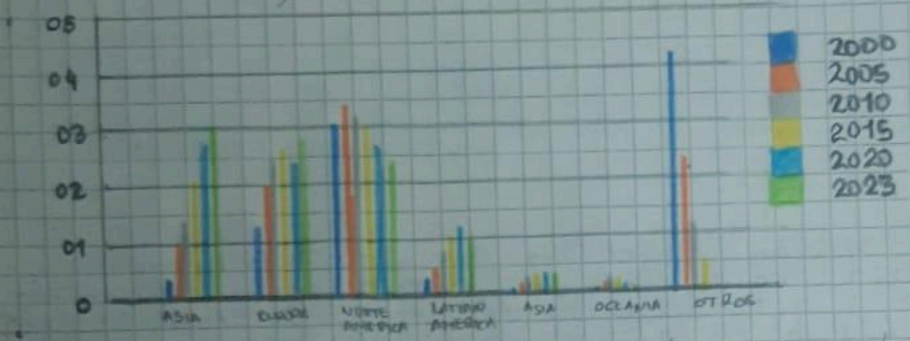
Arquitectura Serverless



El trabajo de diseño e implementación de un chatbot con arquitectura Serverless explora cómo aprovechar tecnologías modernas para crear una aplicación sin la necesidad de administrar servidores tradicionales. Utilizando Amazon Web Services y Dialogflow, el proyecto implementa una solución de chatbot que se ejecuta en la nube, destacando la eficiencia y escalabilidad de la arquitectura Serverless.

En este proyecto, el backend del chatbot está alojado en AWS, lo que significa que la lógica de la aplicación y las funciones de procesamiento se ejecutan en la nube. AWS Lambda, el servicio Serverless de Amazon, ejecuta funciones en respuesta a eventos, como las solicitudes de los usuarios que interactúan con el chatbot. Este enfoque elimina la necesidad de administrar servidores constantemente, reduciendo costos y complejidad operativa.

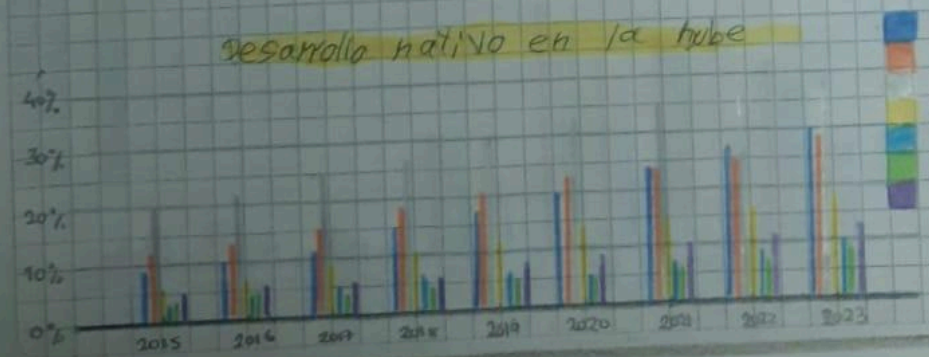
Seguridad de Software



El artículo se centra en el análisis de metodologías de desarrollo de software seguro con enfoque en propiedades agiles, haciendo especial hincapié en Microsoft Security Development Lifecycle y Building Security In Maturity Model. A través de una investigación hermenéutica, descriptiva y heurística, se examina como estas metodologías pueden aplicarse en el ciclo de vida del desarrollo de software y se evalúa su capacidad para integrarse en entornos ágiles.

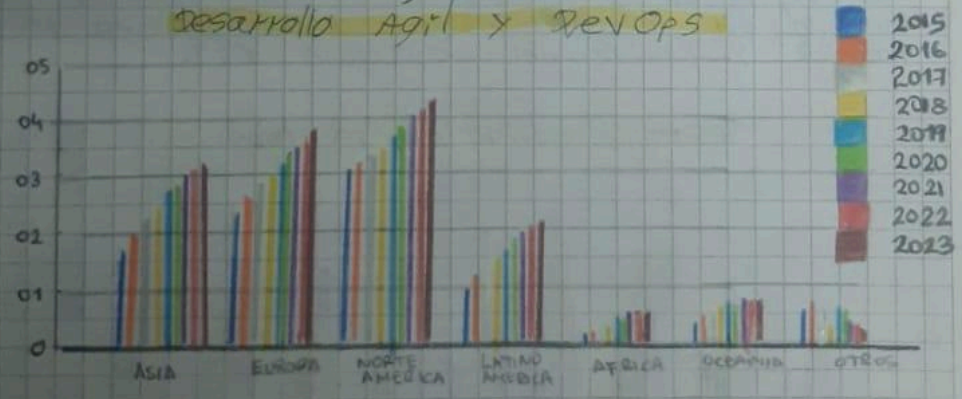
La investigación hermenéutica, descriptiva y heurística realizada en el artículo permitió una comparación de seguridad dentro del desarrollo ágil. El análisis concluye que ambas metodologías son compatibles con entornos agiles, ya que permiten la integración de prácticas de seguridad sin intervenir con la naturaleza iterativa y rapida del desarrollo ágil.

Desarrollo nativo en la nube



El informe presentado por José Alexander Suaza Montes, donde se enfoca en la modernización y automatización de procesos empresariales, utilizando tecnologías de SharePoint y herramientas de Cloud computing. El documento proporciona una visión detallada de las actividades realizadas durante su experiencia, así como de los aprendizajes obtenidos en torno al desarrollo nativo en la nube.

Desarrollo nativo en la nube se refiere al diseño y construcción de aplicaciones que están optimizadas para ejecutarse en plataformas de nube, aprovechando todas las ventajas que estas ofrecen, como la escalabilidad, la flexibilidad, y la disponibilidad global. Este enfoque implica utilizar servicios y tecnologías en la nube desde la etapa de planificación hasta el despliegue y mantenimiento de las aplicaciones.



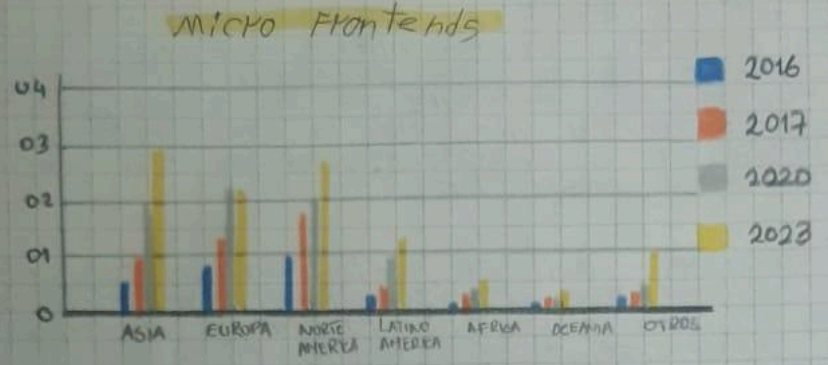
El artículo aborda el desarrollo ágil y DevOps, dos enfoques complementarios que han revolucionado la manera en que se gestionan o entregan proyectos de software en la actualidad. Ambos se enfocan en mejorar la eficiencia y la calidad en el desarrollo, pero lo hacen desde diferentes perspectivas.

El desarrollo ágil se refiere a un conjunto de metodologías que priorizan la colaboración entre los equipos, la iteración rapida, y la capacidad de adaptarse a los cambios durante el proceso de desarrollo. Entre las metodologías ágiles mas conocidas se encuentran Scrum y Kanban.

El desarrollo Ágil para proyectos donde los requisitos, van cambiando o no están completamente definidos desde el inicio ya que permite un ajuste continuo en función de la evolución de las necesidades del cliente o del mercado.

Por otro lado DevOps es una filosofía que busca integrar el desarrollo y las operaciones para mejorar la colaboración entre estos dos equipos, que tradicionalmente han trabajado de manera independiente. DevOps se centra en la automatización y la entrega continua a lo largo de todo el ciclo de vida del software, desde el desarrollo hasta el despliegue y mantenimiento.

DevOps es una extención natural de los principios ágiles, pero su enfoque se amplia para abarcar todo el ciclo de vida del software, incluyendo la operación y el mantenimiento. La automatización es una piedra angular de DevOps ya que permite implementar cambios en el software de manera mas rapida y segura.

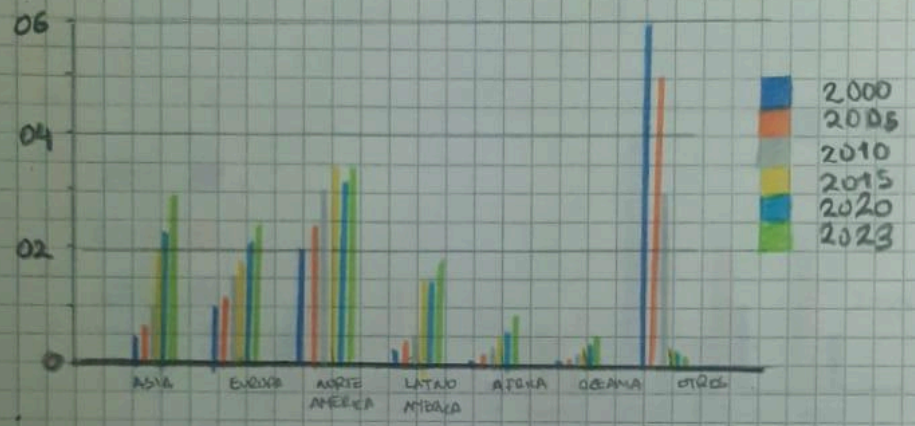


El articulo examina la implementación de micro-frontends en el banco de Bogota como una estrategia para mejorar la mantenibilidad y escalabilidad de los productos digitales, especialmente en el contexto de una arquitectura basada en microservicios para la parte del frontend. A lo largo del documento se destaca como esta tecnologia busca resolver las limitaciones asociados con arquitecturas monoliticos tradicionales.

La implementación de micro-frontends en el banco de bogota fue propuesta como una solución para enfrentar estos desafios. Esta arquitectura permite que cada componente del frontend se modifique y despliegue de manera independiente, sin necesidad de afectar a toda la aplicación. Esto es particularmente útil para fomentar pequeños despliegues continuos y mejorar la responsabilidad individual en cada unidad funcional del software.

El artículo tambien analiza la segregación de componentes en micro-frontends, evaluando cómo esta estrategia puede impactar tanto a la velocidad del desarrollo como en la calidad del software. Aunque la separación de micro-frontends ofrece ventajas claras en terminos de mantenibilidad y escalabilidad, tambien plantea preguntas sobre la eficiencia y la complejidad adicional que puede introdir al tener multiples equipos trabajando en partes diferentes de la aplicación.

Estudio de arquitecturas software para servicios de Internet de las cosas



El artículo compara las EDA con otros patrones arquitectónicos comunes, como las arquitecturas monolíticas y las basadas en microservicios. A diferencia de las arquitecturas tradicionales, donde las iteraciones suelen estar basadas en solicitudes y respuestas síncronicas, una EDA procesa los eventos de manera asíncronica. Esto permite que el sistema responda a los eventos en tiempo real, sin tener que esperar a que se completen otras tareas, lo que mejora la escalabilidad y la eficiencia.

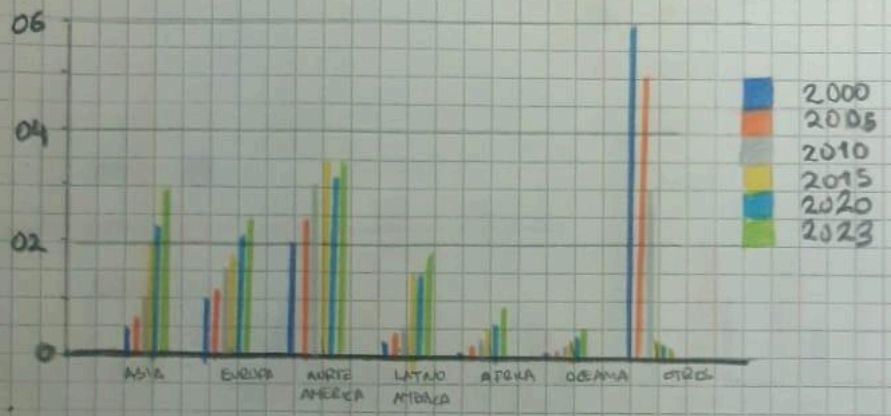
El artículo también menciona la relación de las EDA con el Internet de las cosas. Las arquitecturas orientadas a eventos son ideales para manejar dispositivos IoT, ya que estos dispositivos generan constantemente eventos de manera asíncronica y en tiempo real, o generan eventos en función de sus interacciones con el entorno. Las EDA pueden procesar estos eventos de manera asíncrona y en tiempo real, lo que permite a los sistemas IoT monitorear y reaccionar ante cambios de estado de manera eficiente.

Diseño de una arquitectura basada en contenedores para la integración y despliegue continuo (CI/CD)

El artículo se centra en la Arquitectura Basada en los contenedores, describiendo su diseño, y implementación para el desarrollo y despliegue de aplicaciones modernas. Esta arquitectura utiliza la tecnología de contenedores para encapsular aplicaciones y sus dependencias, facilitando la portabilidad, escalabilidad y administración eficiente de los entornos de desarrollo y producción.

El artículo también explora la automatización de tareas mediante shell scripts y servicios externos. La automatización es crucial para mejorar la eficiencia del desarrollo y despliegue de aplicaciones, reduciendo el tiempo de configuración manual y minimizando errores humanos. Estos scripts permiten la automatización de procesos repetitivos, como la creación de contenedores, la configuración del entorno, y el despliegue de la aplicación.

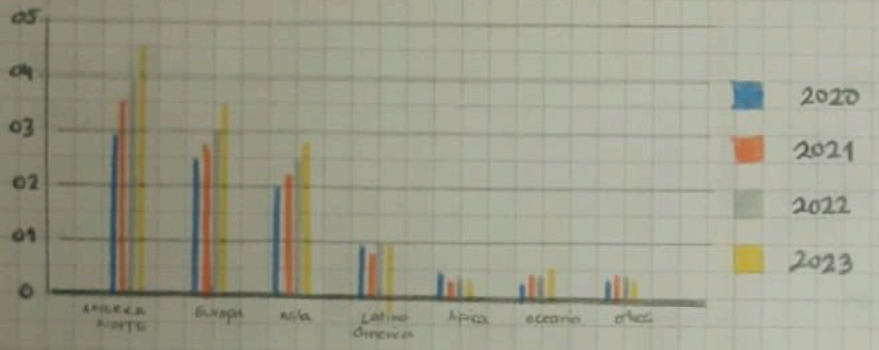
Estudio de arquitecturas software para servicios de Internet de las Cosas



El artículo compara las EDA con otros patrones arquitectónicos comunes, como las arquitecturas monolíticas y las basadas en microservicios. A diferencia de las arquitecturas tradicionales, donde las iteraciones suelen estar basadas en solicitudes y respuestas síncronicas, una EDA procesa los eventos de manera asíncronica. Esto permite que el sistema responda a los eventos en tiempo real, sin tener que esperar a que se completen otras tareas, lo que mejora la escalabilidad y la eficiencia.

El artículo también menciona la relación de las EDA con el internet de las cosas. Las arquitecturas orientadas a eventos son ideales para manejar dispositivos IoT, ya que estos dispositivos generan constantemente eventos de manera asíncronica y en tiempo real, o generan eventos en reacción de sus interacciones con el entorno. Las EDA pueden procesar estos eventos de manera asíncrona y en tiempo real, lo que permite a los sistemas IoT monitorear y reaccionar ante cambios de estado de manera eficiente.

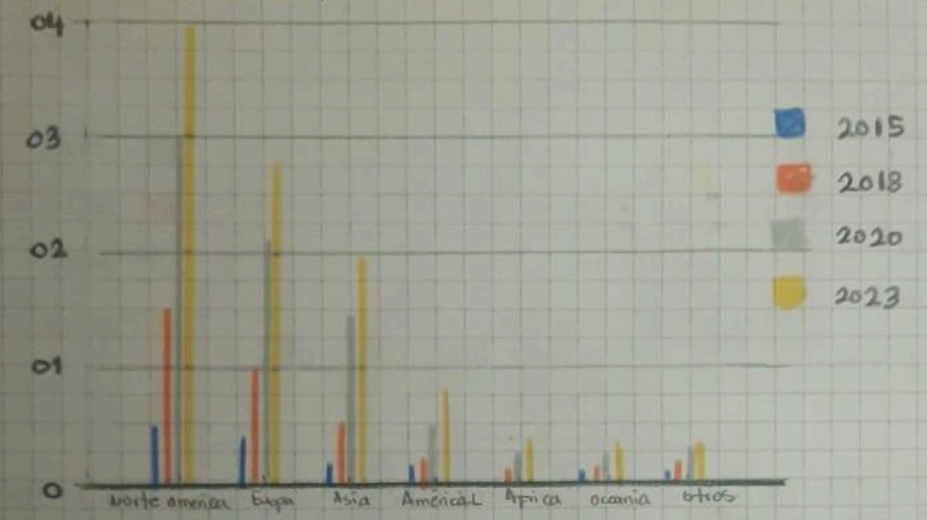
Análisis de la ciberseguridad a la infraestructura tecnológica de la empresa Señal X



La tesis aborda la arquitectura de seguridad zero trust y su implementación en la infraestructura tecnológica de la empresa señal X. Este modelo de seguridad tiene como objetivo proteger la información empresarial en un contexto donde las amenazas cibernéticas son cada vez más satisficadas y las redes tradicionales no pueden garantizar la seguridad completa de los sistemas.

El documento también aborda los desafíos enfrentados durante la implementación de la arquitectura zero trust. Uno de los principales obstáculos fue la adaptación de la infraestructura tecnológica existente para cumplir con los requisitos de zero trust. Esto implicó una inversión significativa en nuevas herramientas de seguridad, así como en la formación del personal para que comprendieran y aplicaran los principios del modelo.

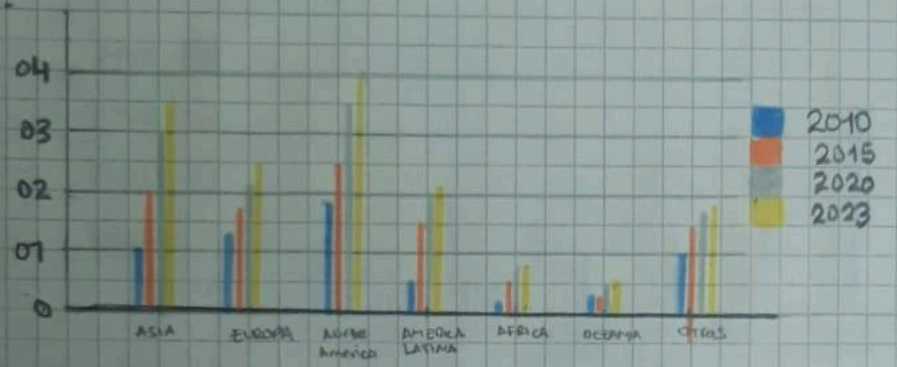
Sistema de delibery cross-cloud basado en Data Fabric



El artículo aborda el concepto de Data Mesh, un enfoque innovador para la gestión y compartición de datos en organizaciones complejas. A diferencia de los enfoques tradicionales centralizados, Data Mesh distribuye la responsabilidad de los datos a través de diferentes equipos, tratándolos como productos independientes.

El artículo concluye que el enfoque Data Mesh representa una evolución significativa en la gestión de datos, particularmente en organizaciones grandes y complejas donde los enfoques tradicionales pueden resultar ineficaces. Al tratar los datos como productos y descentralizar su gestión, Data Mesh ofrece una solución escalable y flexible que puede mejorar la calidad y la accesibilidad de los datos en toda la organización

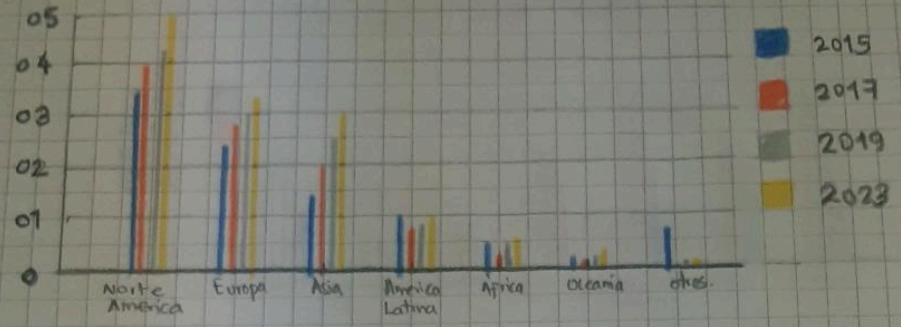
Diseño y despliegue de soluciones auto-adaptativas basadas en microservicios utilizando infraestructura elástica en la nube. Una aplicación sobre la IoT en la Industria 4.0



El artículo trata sobre el diseño y despliegue de soluciones auto-adaptivas basadas en microservicios utilizando infraestructura elástica en la nube, aplicado en el contexto de la internet de las cosas. La IoT permite que los objetos físicos adquieran inteligencia, comunicándose entre ellos y generando datos que pueden ser usados para mejorar procesos industriales. El desafío de implementar estas tecnologías no es trivial, ya que requiere una gestión eficiente de los datos y la capacidad de adaptarse a entornos cambiantes. Para abordar este reto, el proyecto propone una arquitectura basada en microservicios que aprovecha los recursos de la nube, con una infraestructura flexible, escalable y con alta disponibilidad.

El trabajo desarrolla un prototipo que demuestra cómo estos sistemas pueden adaptarse automáticamente a cambios en el entorno sin intervención humana, respondiendo a cualquier necesidad que surja.

Desarrollo de aplicación web basada en FaaS con .NET core Evolución desde aplicación Monolítica.

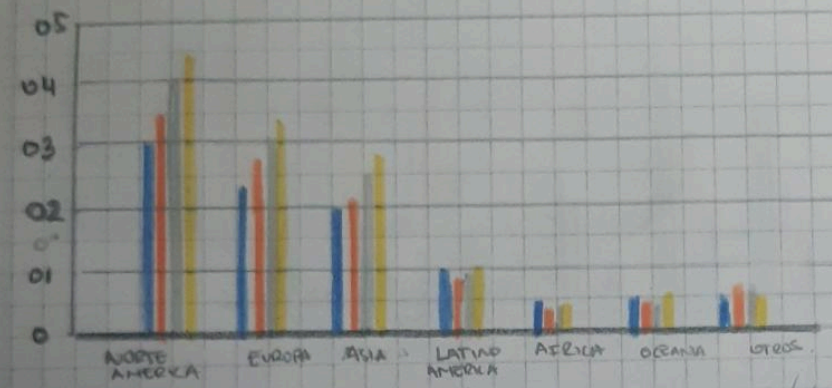


El artículo explora el concepto de Funciones como servicio (FaaS), un modelo de computación en la nube también conocido como "serverless" o sin servidor. Este enfoque permite ejecutar funciones individuales en respuesta a eventos específicos, sin que los desarrolladores necesiten gestionar los servidores subyacentes.

El artículo destaca la implementación de FaaS utilizando varias tecnologías, entre ellas Azure Functions. Esta es una plataforma popular para FaaS que se integran con otras herramientas y servicios de Microsoft, facilitando la creación y gestión de funciones en la nube.

El artículo concluye que el modelo FaaS proporciona una solución eficaz para ejecutar funciones en la nube sin la necesidad de gestionar la infraestructura subyacente. Al centrarse en el código y permitir la ejecución basada en eventos, FaaS mejora la flexibilidad y la escalabilidad de las aplicaciones. La comparación entre plataformas como Azure Functions y OpenFaaS subraya la importancia de seleccionar la herramienta adecuada en función de los requisitos específicos del proyecto y el entorno tecnológico.

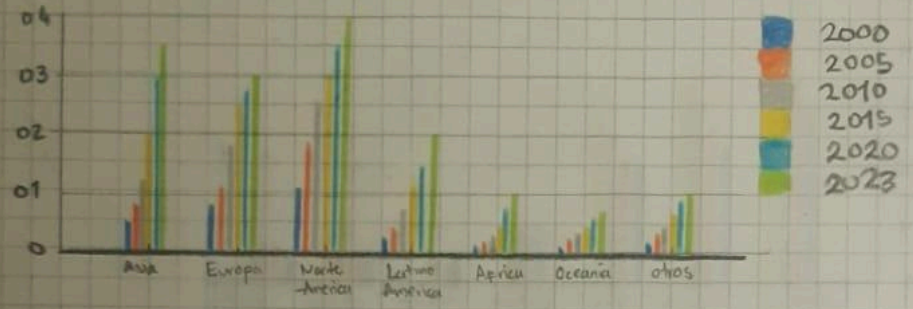
Arquitecturas multicloud desde una perspectiva de seguridad



El artículo explora el enfoque Multicloud, que se refiere al uso de múltiples servicios de computación en la nube proporcionados por diferentes proveedores. Este enfoque está ganando popularidad entre las empresas debido a su capacidad de mejorar la gestión de sistemas y ofrecer una serie de beneficios importantes que serían difíciles de lograr utilizando solo infraestructura propia o confiando en un único proveedor de nube.

El artículo concluye que la adopción de un enfoque Multicloud ofrece a las empresas una estrategia sólida para mejorar la eficiencia, la seguridad, y la flexibilidad de sus operaciones, optimizando el uso de recursos en múltiples plataformas y reduciendo los riesgos asociados con la dependencia de un solo proveedor de servicios en la nube.

Adopción de enfoque Api-First: Caso de estudio



El artículo analiza el enfoque Api-First como un metodo de desarrollo emergente en el que las Apis se sitúan en el centro del diseño de software. Esto significa que, en lugar de desarrollar primero las funcionalidades internas y luego crear API para exponerlas, los equipos de desarrollo empiezan definiendo y diseñando las API antes de trabajar en el resto de aplicaciones. Esta metodologia facilita que todos los componentes del sistema trabajen entorno a una API bien definida desde el principio.

El Artículo tambien resalta que, aunque el enfoque Api-First está cada vez más adoptado en la industria, aún no ha sido suficientemente estudiado en la literatura academica, lo que podria indicar la necesidad de más investigación en este campo. Sin embargo, a través de un caso de estudio presentado en el artículo, se observa que la implementación de este enfoque en la practica tiene desafios importantes, especialmente en relación con la dependencia de herramientas y tecnologias. Las Apis diseñadas inicialmente a menudo requieren ajustes durante el proceso de desarrollo, lo que puede introducir dificultades para mantener las especificaciones actualizadas y garantizar la coherencia entre la definición y la implementación del API.