

Nombre		Fecha	dia	mes	año
Profesor		Materia			
Institución		Curso			Nota

1. Comparación del rendimiento entre arquitecturas monolíticas y de microservicios en sistemas web.

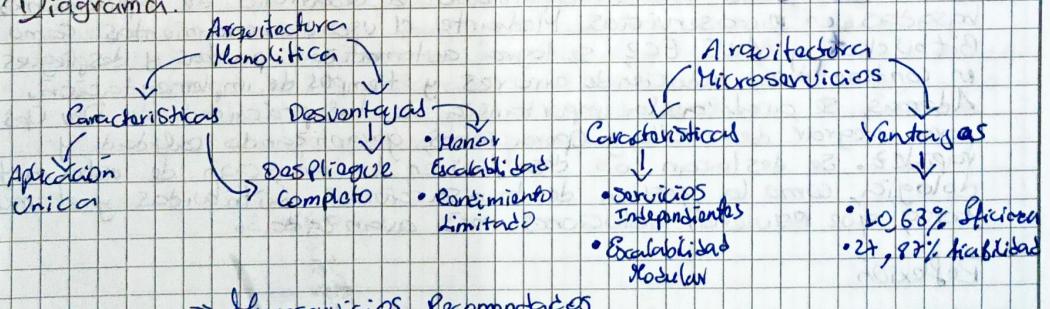
Resumen.

El artículo compara las arquitecturas monolíticas y de microservicios en términos de eficiencia y fiabilidad en sistemas web. A través de un estudio experimental se evalúa cómo las arquitecturas impactan el consumo de recursos, tiempo de respuesta y conexiones simultáneas. Los resultados muestran que la arquitectura de microservicios mejora la eficiencia en un 10,6% y la fiabilidad en un 27,8% frente a la monolítica. Se observa que la adopción de microservicios ofrece ventajas significativas para entornos exigentes, como el desarrollo de software em presarial. Se recomienda evaluar los costos de implementación y estudiar herramientas específicas para maximizar su efectividad.

Pensamiento:

Se resalta la importancia de elegir la arquitectura adecuada para los sistemas web, especialmente en proyectos complejos. Lleva a reflexionar sobre cómo una arquitectura bien diseñada puede marcar la diferencia no solo en el rendimiento técnico, sino también en la experiencia del usuario. Las ventajas de los microservicios, como la modularidad y escalabilidad, son especialmente relevantes en un mundo donde la tecnología avanza rápidamente y las necesidades de los usuarios cambian constantemente. Este tema me ayuda a valorar la importancia de considerar la viabilidad técnica y económica antes de implementar cambios arquitectónicos.

Diagrama.



⇒ Microservicios Recomendados para proyectos complejos.

2. Arquitectura de microservicios para aplicaciones desplegadas en Contenedores.

Resumen

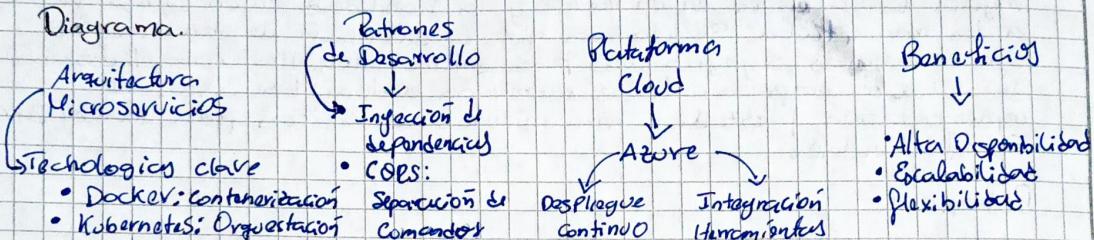
Este artículo explora cómo las arquitecturas de microservicios han transformado el desarrollo de aplicaciones al combinarlas con tecnologías como contenedores y orquestadores. Utilizando Docker para la containerización y Kubernetes para la gestión, se asegura alta disponibilidad, escalabilidad y despliegues rápidos en entornos clara. Además, se aplican patrones de desarrollo como

Inyecciones de dependencias y CQRS, lo que permite diseñar aplicaciones modulares y eficientes. La propuesta se enfoca en aprovechar plataformas como Azure, que integran herramientas de desarrollo y despliegue continuo, asegurando una infraestructura adaptable y robusta.

Reflexión

Se muestra cómo los microservicios representan un cambio fundamental en el diseño de aplicaciones modernas. Me hizo pensar en cómo la containerización y los orquestadores no solo optimizan los recursos, sino que también democratizan el acceso a tecnologías avanzadas. Con un enfoque tipo modelo lo cual es valioso en un mundo donde las demandas de los usuarios evolucionan rápidamente. Me parece importante valorar las metodologías ágiles y el desarrollo en la nube como caminos para garantizar flexibilidad y competitividad.

Diagrama.



3. Método de automatización del despliegue continuo en la nube para microservicios.

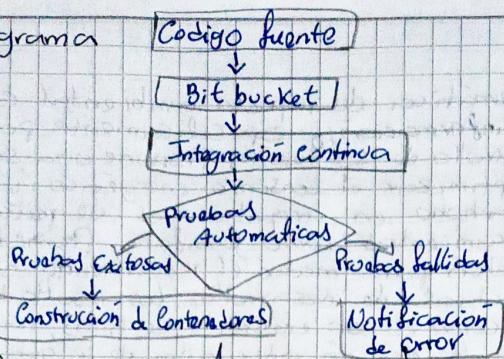
Resumen

El artículo detalla cómo la automatización del despliegue continuo en entornos Cloud han transformado el desarrollo de aplicaciones basadas en microservicios. Mediante el uso de herramientas como Bitbucket y AWS EC2, se logra automatizar pruebas y despliegues en contenedores, reduciendo errores y tiempos de implementación. Además, se analiza la importancia de las prácticas de DevOps para integrar desarrollo y operaciones, garantizando calidad y rapidez. Se destacan los desafíos en la adopción de estos métodos, como la gestión de transacciones distribuidas y el monitoreo, que requieren soluciones más avanzadas.

Reflexión

Se refuerza la importancia de adoptar procesos automatizados en el desarrollo de software moderno. Me hizo pensar en cómo la integración de pruebas y despliegues automáticos no solo mejora la calidad, sino que también reduce significativamente los riesgos asociados al error humano. Además, resalta la necesidad de seguir innovando en áreas como la seguridad y la gestión de microservicios. Esta metodología parece ideal para organizaciones que buscan adaptarse rápidamente a cambios tecnológicos y de mercado.

Diagrama



4. Diseño de una Arquitectura basada en Contenedores para la Integración y el despliegue continuo (CI/CD)

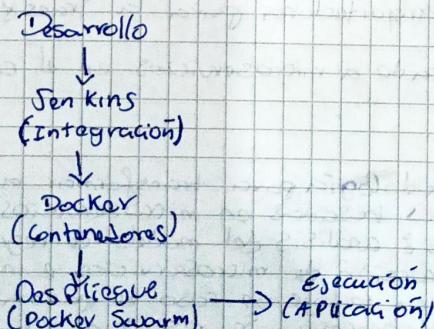
Resumen

En el artículo se presenta el diseño e implementación de una arquitectura basada en contenedores diseñada para soportar la integración y el despliegue continuo (CI/CD). Utilizando herramientas como Docker, Jenkins y Docker Swarm, el trabajo se detalla cómo se automatizan tareas con scripts de Shell. Se gestionan aplicaciones en entornos laboratorios locales y se asegura la escalabilidad en servidores remotos. Además destaca los beneficios del uso de contenedores sobre máquinas virtuales tradicionales, como mayor eficiencia y menor consumo de recursos. La propuesta incluye técnicas como versionado de imágenes y redes de alta disponibilidad, optimizando la ejecución de aplicaciones modernas para pequeñas y medianas empresas.

Reflexión

Este artículo muestra cómo las tecnologías basadas en contenedores y la automatización CI/CD están transformando la manera de desarrollar y gestionar aplicaciones. Personalmente, me hace valorar la importancia de herramientas como Docker y Jenkins para simplificar tareas repetitivas y mejorar la eficiencia. También me ayuda a reflexionar sobre cómo este enfoque modular y escalable es clave para proyectos actuales, especialmente en entornos empresariales donde la flexibilidad y la rapidez de implementación son esenciales. La automatización no solo ahorra tiempo, sino que también reduce errores, lo que la convierte en una herramienta indispensable para los equipos de desarrollo.

Diagrama



5. Recomendaciones para el desarrollo de software sostenible en arquitecturas de microservicios.

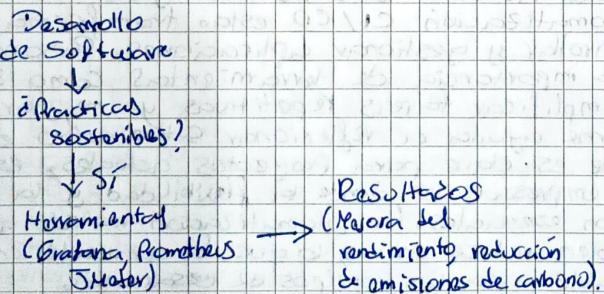
Resumen

El artículo aborda la problemática del impacto ambiental causado por las tecnologías de la información, específicamente por las malas prácticas en el desarrollo de software. Propone una serie de recomendaciones para optimizar el consumo energético y minimizar las emisiones de carbono en arquitecturas de micro-servicios. A través de herramientas como Grafana, Prometheus y JMeter se realizaron pruebas para medir el rendimiento del hardware con diferentes prácticas de desarrollo. Los resultados evidencian que prácticas como el uso eficiente de hilos, la implementación de cachés y la elección adecuada de lenguajes y versiones de programación tienen un impacto significativo en la sostenibilidad energética. El estudio busca crear conciencia en la comunidad tecnológica sobre la importancia de adoptar prácticas verdes en el diseño y desarrollo de aplicaciones modernas.

Reflexión

En este artículo es como una invitación a analizar nuestra responsabilidad como desarrolladores en la reducción del impacto ambiental de las tecnologías de la información. Me hizo pensar en cómo nuestras decisiones diarias, como el diseño de arquitectura y la elección de herramientas, pueden contribuir al cambio climático. La idea de un "Software Verde" no solo tiene un impacto positivo en el ambiente, sino que también mejora el rendimiento de las aplicaciones, lo que es benéfico para usuarios y empresas. Es motivador saber que pequeños cambios en nuestras prácticas pueden generar grandes resultados y promover un futuro más sostenible.

Diagrama



6. Definición de una arquitectura para la transformación de Software centralizado a microservicios en el ámbito web.

Resumen

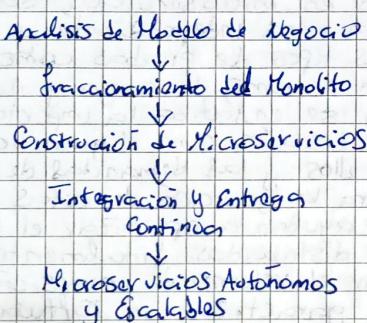
Se presenta una metodología para transformar aplicaciones monolíticas en arquitecturas basadas en microservicios. Se organiza en tres bloques principales: análisis del modelo de negocio, fragmentación del monolito y construcción de microservicios e integración y entrega continua. La investigación incluye un caso de estudio práctico que detalla cada etapa del proceso. Se resaltan las ventajas de los microservicios, como escalabilidad, mantenimiento más sencillo y despliegues más seguros. Además, se proponen buenas prácticas para gestionar la fragmentación del código y su despliegue, garantizando una transición eficiente y confiable.

Nombre	Fecha	dia	mes	año
Profesor	Materia			
Institución	Curso			Nota

Reflexión

Este artículo muestra la importancia de adoptar una metodología adecuada para la transformación de software monolítico a micro servicios. El enfoque gradual permite minimizar los riesgos asociados a la migración asegurando que cada módulo sea funcional y autónomo. Además, resalta la necesidad de contar con herramientas y equipos capacitados para garantizar el éxito del proceso. Considero que esta transición es vital para las empresas que buscan adaptarse a entornos tecnológicos cambiantes y mejorar su capacidad de respuesta ante las necesidades del mercado. Implementar microservicios no solo reduce costos a largo plazo, sino que también fomenta la innovación y la flexibilidad.

Diagrama



7. Arquitectura de software para el desarrollo de aplicaciones

Web Orientadas a micro servicios en TecNM Campus Escárcega

Resumen

En el artículo se describe la implementación de una arquitectura de software para el desarrollo de aplicaciones web basadas en microservicios, diseñada para remplazar al enfoque monolítico en el TecNM Campus Escárcega. La propuesta integra tecnologías como Angular para el cliente y Spring Boot para los servicios, enfocándose en reducir tiempos de desarrollo y aumentar la productividad obteniendo resultados favorables en comparación con el paradigma monolítico con mejoras en escalabilidad, modularidad y experiencia del usuario. Además, se implementaron módulos reutilizables que simplifican la creación de nuevos proyectos y manteniendo a los estudiantes y docentes actualizados con tecnologías modernas.

Reflexión

Se muestran una solución práctica para adoptar para algunos modernos en el desarrollo de software. Me hizo reflexionar sobre cómo los microservicios ofrecen una ventaja clave en términos de escalabilidad y modularidad, características esenciales en un entorno académico donde los recursos y el tiempo son limitados. La transición del paradigma monolítico al de microservicios no solo fomentaba la innovación, sino que también aseguraba que los futuros profesionales estén alineados con las tendencias tecnológicas actuales. Creo que esta arquitectura podría replicarse en otros

Instituciones, generando un impacto positivo en la formación de estudiantes y en la eficiencia de proyectos académicos y empresariales.

Diagrama.

Requerimientos de Aplicación

↓
¿Necesita Microservicios?

↓ Sí

Desarrollo de Microservicios

↓
Implementación (Angular y Spring Boot)

↓
Despliegue (Escalable y Seguro) → Monitoreo y Mantenimiento

8. Pruebas de software para microservicios: Retos y Estrategias

Resumen

En el artículo se abordan la complejidad de realizar pruebas en aplicaciones basadas en microservicios, donde cada componente debe ser evaluado de manera independiente y en conjunto. Se destacan diferentes tipos de pruebas como unitarias, de integración, de extremo a extremo de carga y de seguridad. De la misma manera se enfatiza la importancia de la automatización para enfrentar desafíos como el gran volumen de servicios, las dependencias entre ellos y las demandas de rendimiento. Entre las herramientas clave para la ejecución de estas pruebas se mencionan Parasoft y JMeter que ayudan a optimizar el proceso y garantizar la calidad en sistemas distribuidos. Finalmente se proponen estrategias para maximizar el retorno de inversión (ROI) en la ejecución de pruebas y garantizar un software robusto y escalable.

Reflexión

En este artículo se habla sobre la relevancia de las pruebas en sistemas complejos como los basados en microservicios. A menudo, el enfoque está en el desarrollo, pero garantizar que todo funcione correctamente es igual de importante. Me pareció interesante cómo herramientas automatizadas pueden simplificar este proceso, permitiendo detectar errores y mejorar el rendimiento antes de que lleguen a los usuarios. Además, las pruebas de carga y seguridad son un recorrido de la responsabilidad que tenemos como desarrolladores para proteger los datos y ofrecer experiencias confiables. Creo que estas prácticas deberían integrarse desde el principio en cualquier proyecto de Software.

Diagrama

Desarrollo de Microservicios

↓
Tipos de Proyectos:

Unitarias, Integración, Extremo a Extremo
Carga, Seguridad

↓
Automatización (Parasoft, JMeter)

↓
Ejecución de pruebas
(Independientes y Conjuntas)

↓
Análisis de Resultados
(Errores Persistentes)

→
Mejora Continua
(Iteraciones, Refactorización).

9. Modelo de arquitectura de software para el procesamiento de datos en arquitecturas actuales.

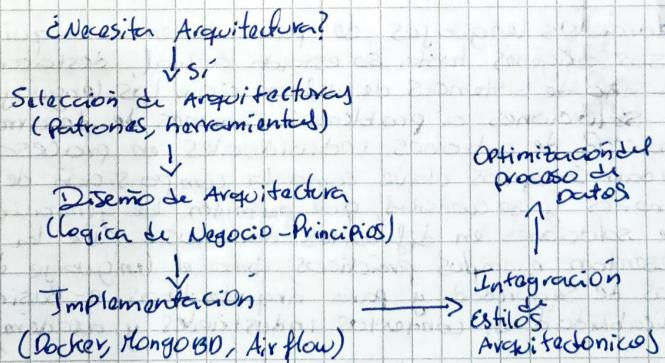
Resumen

Se propone un modelo de arquitectura de software diseñado para abordar los desafíos en el procesamiento de datos masivo. Basándose en patrones arquitecturales y herramientas modernas como Docker, MongoDB y Airflow, se ofrece una guía comparativa para seleccionar y evaluar arquitecturas adecuadas en función de las necesidades empresariales. El modelo combina técnicas de diseños basadas en lógica de negocio y principios de arquitectura para mejorar la eficiencia y escalabilidad en el manejo de datos. Además, se analiza cómo diferentes estilos arquitectónicos pueden integrarse para superar problemas comunes como la latencia y la corrupción de datos, ofreciendo soluciones optimizadas para entornos dinámicos.

Pensamiento

En este artículo ayuda a comprender la importancia de elegir cuidadosamente una arquitectura de software adecuada al contexto y las necesidades del negocio. La propuesta de usar patrones arquitecturales no solo mejora la organización de los datos sino que también optimiza los recursos, lo que es crucial en un entorno donde el volumen de datos crece exponencialmente. Me parece interesante cómo tecnologías como Docker y Airflow simplifican el manejo de datos masivos, al tiempo que promueven la flexibilidad. Esto resalta la necesidad de que las empresas inviertan en capacitación y herramientas que respalden una toma de decisiones informada y efectiva. Implementar un modelo como este no solo impacta en la productividad, sino también en la capacidad de adaptación tecnológica.

Diagrama.



10. Arquitectura para la implementación de servicios de video en Redes Móviles: Enfoque en SDN y Segmentación de Red.

Resumen

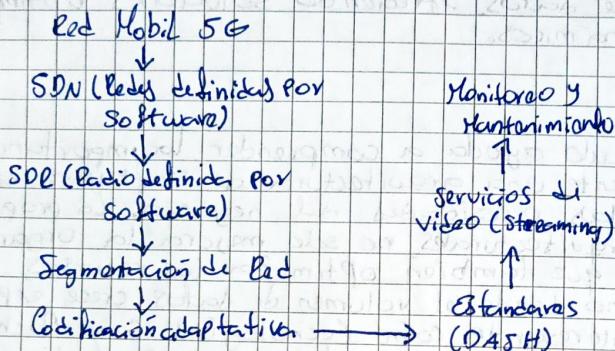
En este artículo se propone una arquitectura innovadora para ofrecer servicios de video streaming en redes móviles, utilizando conceptos como redes definidas por software (SDN), radio definida por software (SDR) y segmentación de red. La propuesta destaca por su flexibilidad y escalabilidad, presentando dos perspectivas: funcional e implementativa. Se describen módulos clave como la codificación adaptativa y el uso de estándares como DASH para garantizar una transmisión eficiente. Además, se analiza la viabilidad de la arquitectura en redes 5G mediante pruebas de concepto, mejoras significativas en la calidad del servicio y la reducción de latencia. Esta solución es aplicable tanto en

entornos académicos como empresariales, sirviendo como un referente en la modernización de infraestructuras de red.

Reflexión

Se invita a reflexionar sobre cómo las tecnologías emergentes como SDN y SDR pueden transformar la forma en que consumimos y distribuimos contenido multimedia en redes móviles. Personalmente, me sorprendió cómo la segmentación de red puede garantizar la calidad del servicio incluso en condiciones de alta demanda. Este enfoque no solo moderniza las infraestructuras existentes, sino que también ofrece una solución más sostenible y flexible para satisfacer las necesidades de la era digital. Creo que adoptar este tipo de arquitecturas es esencial para garantizar que las redes estén preparadas para las demandas futuras.

Diagrama



11. Lenguajes de patrones de arquitectura de software: Una aproximación al Estado del Arte.

Resumen.

Se exploran los lenguajes de patrones en la arquitectura de software, desde sus orígenes hasta su estado actual, destacando su importancia en el diseño de sistemas de información. Los lenguajes de patrones ofrecen soluciones a problemas comunes de desarrollo mediante la combinación de patrones individuales en procesos estructurales. Se abordan conceptos clave como la composición de patrones, historias de patrones y secuencias, que permiten la generalización y reutilización de soluciones en diferentes dominios. De la misma manera se presentan ejemplos prácticos, como el lenguaje de patrones para sistemas de seguridad y para arquitecturas e-Business, destacando su aplicabilidad en contextos industriales y académicos.

Reflexión

El artículo lleva a reflexionar sobre cómo los lenguajes de patrones son herramientas fundamentales para optimizar el diseño de software. Me parece interesante cómo estos lenguajes facilitan la resolución de problemas específicos y permiten generar soluciones reutilizables y adaptables. En un mundo donde los sistemas se vuelven más complejos, esta metodología no solo ahorra tiempo, sino que también garantiza la calidad de software. Además, considero que el uso de lenguajes de patrones fomenta la innovación, ya que combina la experiencia pasada con las necesidades actuales. Este enfoque debería ser adoptado ampliamente por equipos de desarrollo que buscan eficiencia y sostenibilidad.

Nombre

Fecha día mes año

Profesor

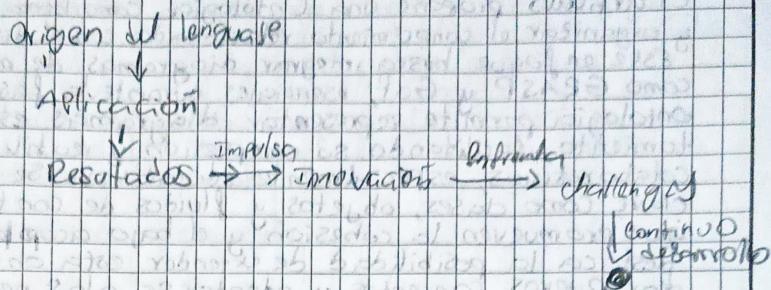
Materia

Institución

Curso

Nota

Diagrama



12. Patrones de diseño, Refactorización y Antipatrones: Ventajas y Desventajas en el Software Orientado a Objetos.

Resumen

Se explica los conceptos de patrones de diseño, refactorización y antipatrones, enfocándose en sus ventajas y desventajas al aplicarlos en software orientado a objetos. Los patrones de diseño son soluciones recurrentes que mejoran la flexibilidad y mantenibilidad del software. Por otro lado, los antipatrones identifican prácticas negativas que generan problemas recurrentes, mientras que la refactorización permite transformar estas malas prácticas en código más comprensible y eficiente. En el texto destaca la importancia de estas técnicas para evitar errores comunes y optimizar el desarrollo, aunque advierte sobre el riesgo de usarlas de manera inapropiada, lo cual podría aumentar la complejidad innecesariamente.

Reflexión

Este artículo ayuda a apreciar el impacto de las buenas y malas prácticas en el desarrollo del software. Los patrones de diseño son una guía útil para resolver problemas comunes, pero también requieren un uso adecuado para evitar que añadan complejidad. Es interesante cómo los patrones y antipatrones no solo identifican fallas, sino que también ofrecen una solución refactorizada para corregirlas. Este enfoque práctico es esencial en un entorno donde el código mal estructurado puede generar retrasos y costos adicionales. Estas herramientas no solo mejoran la calidad del software, sino también fomentan una cultura de aprendizaje continuo entre los desarrolladores.

Diagrama

Problemas de Desarrollo

Patrones de Diseño

Ventajas

- Flexibilidad
- Mantenibilidad

Desventajas

- Complejidad
- Inmesesantes

Mayoria de los
casos de software

↑
Transformación de
Código

Antipatrones

Identificación de Prácticas Negativas

→ Refactorizacional

13. Una Ontología para la representación de conceptos de Diseño de Software.

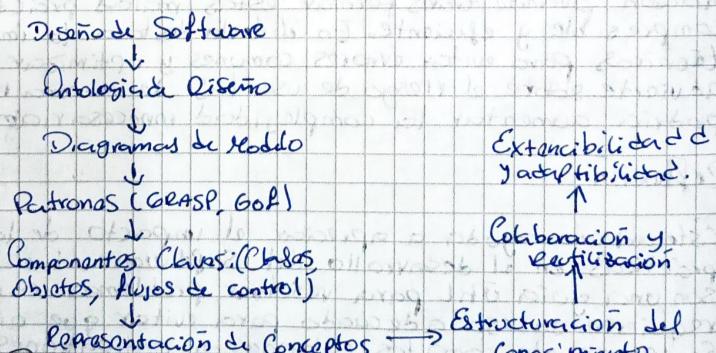
Resumen

El artículo propone una Ontología como herramienta para estructurar y organizar el conocimiento relacionado con el diseño de software. Este enfoque busca integrar diagramas de modelado y patrones como GRASP y GoF, esenciales durante la fase de diseño. La ontología permite representar diagramas estructurales y de comportamiento, facilitando su comprensión y reutilización en entornos colaborativos, especialmente educativos. Se identifican componentes clave como clases, objetos y flujos de control, así como patrones que promueven la cohesión y el bajo acoplamiento. Además, se destaca la posibilidad de extender esta ontología para incorporar nuevos conceptos y adaptarse a las necesidades de la industria.

Reflexión

Este artículo me hizo reflexionar sobre la importancia de estructurar el conocimiento en el diseño de software, un aspecto que suele ser complejo y disperso. Las ontologías, al organizar conceptos como diagramas y patrones no solo facilitan la enseñanza, sino también la implementación de buenas prácticas en el desarrollo de software. Considero que integrar herramientas como esta en entornos académicos y empresariales podría mejorar la calidad de los proyectos y fomentar la colaboración entre equipos. Además, la capacidad de extender la ontología la convierte en una solución adaptable para enfrentar los desafíos tecnológicos actuales y futuros.

Diagrama



14. Revisión de los Patrones de Diseño de Software Aplicados a las Aplicaciones Web.

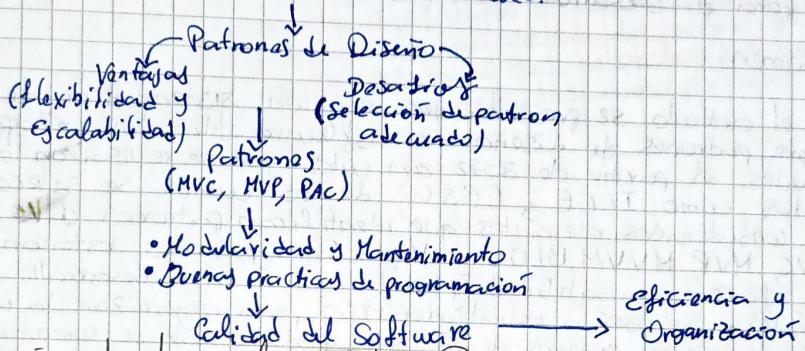
Resumen

Se examina cómo los patrones de diseño se han aplicado en el desarrollo de aplicaciones web para optimizar su rendimiento y reducir errores. Los patrones como MVC, MVPy PAC destacan por su capacidad para modularizar componentes y mejorar la mantenibilidad del software. A través de una revisión bibliográfica, se identificaron las ventajas de estos patrones en términos de flexibilidad y escalabilidad, aunque también se señaló la dificultad de seleccionar el patrón adecuado debido a la amplia variedad existente. Se concluye que la aplicación efectiva de patrones de diseño puede incrementar significativamente la calidad de las aplicaciones web, haciendo el desarrollo más eficiente y organizado.

Reflexión

Podemos hablar de la importancia de los patrones de diseño en el desarrollo de software, especialmente en un entorno tan cambiante como las aplicaciones web. Los patrones como MVC o MVP no solo facilitan la modularidad y mantenimiento, sino que también promueven buenas prácticas de programación. Sin embargo, también en función del reto de seleccionar el patrón adecuado para cada problema, un proceso que requiere experiencia y un entendimiento profundo del contexto. Considero que es útil tanto para estudiantes como para profesionales, ya que subraya la relevancia de los patrones en la construcción de software más eficiente y escalable.

Diagrama Desarrollo de Aplicaciones Web.



15. Incidencia de los patrones de Diseño en la Seguridad de Aplicaciones Web.

Resumen.

Se explora como los patrones de diseño de software pueden mitigar vulnerabilidades comunes en aplicaciones web, basándose en los riesgos identificados por OWASP. Se analizan patrones como "Chain of Responsibility", "State" y "Proxy" aplicados a problemas como el control de acceso roto, ataques de fuerza bruta y Cross-Site Scripting. Utilizando herramientas como SonarQube y NDepend, se evaluaron aplicaciones con y sin patrones de diseño, mostrando mejoras significativas en seguridad y mantenimiento. Este enfoque demuestra cómo los patrones estructurales y comportamentales no solo mejoran la arquitectura del software, sino también su capacidad para prevenir ataques.

Reflexión

Con este artículo me hizo reflexionar sobre la relación directa entre el diseño de software y la seguridad en aplicaciones web. Es interesante como los patrones de diseño no solo resuelven problemas técnicos, sino que también ofrecen soluciones prácticas para vulnerabilidades críticas. Me pareció valiosa la integración de herramientas como SonarQube, que facilita la identificación de riesgos antes de que afecten a los usuarios finales. En un entorno tecnológico donde las amenazas son constantes, se reafirma la importancia de planificar y diseñar con seguridad en mente desde las etapas iniciales del desarrollo. Adoptar estas prácticas no solo mejora la calidad del software, sino que también protege la información sensible de los usuarios.

Diagramas

Desarrollo de Aplicaciones Web.

Riesgos de seguridad (OWASP)

Patrones de Diseño

Patrones Chain of Responsibility, State, Proxy

Mitigación de Vulnerabilidades

Protección de la información

Mayores Seguridad

Evaluación de seguridad (SonarQube, NO depend)

16. Análisis comparativo de patrones de Diseño de Software para el Desarrollo de Aplicaciones Móviles de Calidad.

Resumen

En el artículo se presenta una revisión sistemática de la literatura sobre patrones de diseño de software utilizados en aplicaciones móviles. A partir de 3072 artículos que se revisaron en bases de datos como IEEE y EBSICO de los cuales se seleccionaron 16 considerados relevantes que identifican patrones tales como MVC, MVP, MVVM, BLOC y Viper. Estos patrones destacan por mejorar la eficiencia, usabilidad y reutilización en el desarrollo de aplicaciones móviles. Se uso la estrategia PICO para organizar la investigación y se definieron criterios de calidad que ayudan a comparar y seleccionar patrones adecuados según las necesidades del proyecto. El estudio concluye que el uso correcto de patrones puede optimizar el tiempo y los recursos, garantizando aplicaciones móviles más robustas y de alta calidad.

Reflexión

Con este artículo se comprende cómo los patrones de diseño impactan directamente en la calidad del desarrollo de aplicaciones móviles. Patrones como MVC o Viper no solo simplifican el diseño, sino que también fomentan la reutilización y escalabilidad, aspectos esenciales en un entorno tecnológico tan dinámico. La metodología PICO utilizada en la investigación resalta la importancia de un enfoque estructurado para seleccionar las mejores soluciones. Considero que aplicar estos patrones desde las primeras fases del desarrollo puede ahorrar tiempo y costos, al tiempo que mejora la experiencia del usuario final. Este estudio refuerza la idea de que el diseño de software no solo es técnico, sino también estratégico.

Diagramas

Patrón/Criterio	Reutilización	Extensibilidad	Mutabilidad	Complejidad
iOS - fabrica	Alta	Media	Alta	Media
iOS - MVC	Media	Media	Media	Baja
Android - MVP	Alta	Alta	Alta	Media
Android - MVVM	Alta	Alta	Alta	Alta

Nombre

Fecha dia mes año

Profesor

Materia

Institución

Curso Nota

17. Arquitectura de Software para el Sistema de Visualización Médico Vismedic

Médico: Vismedic

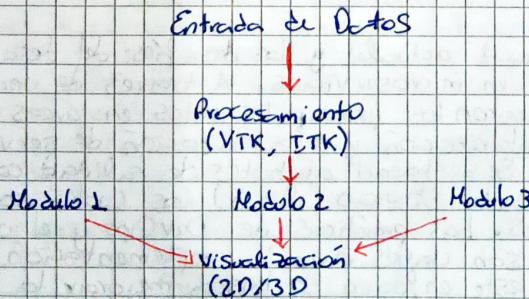
Resumen

El artículo describe la propuesta de una nueva arquitectura de software para el sistema Vismedic, diseñada para visualizar imágenes médicas. Basándose en una combinación de estilos arquitectónicos como componentes, capas y tuberías y filtros, la arquitectura busca solucionar problemas de extensibilidad, reusabilidad y dependencias de la versión previa. La propuesta se centra en integrar plugins que permitan una mayor flexibilidad y personalización de funcionalidades, facilitando actualizaciones y reduciendo costes de mantenimiento. La validación del modelo se realizó mediante el método ATAM y pruebas basadas en prototipos, confirmando mejoras significativas en atributos como portabilidad, mantenibilidad y escalabilidad.

Reflexión:

En este artículo se habla sobre la importancia de una arquitectura bien diseñada en sistemas tan críticos como los de visualización médica. Es interesante cómo la combinación de estilos arquitectónicos puede abordar problemas específicos y mejorar la calidad del software. La inclusión de plugins no sólo hace más flexible el sistema, sino que también fomentaba la innovación y la colaboración entre equipos. Considero que esta investigación es un ejemplo claro de cómo una buena arquitectura puede ahorrar tiempo y recursos, al mismo tiempo que mejora la experiencia del usuario final. Este enfoque debería ser adoptado ampliamente en proyectos tecnológicos con alto impacto social.

Diagrama



18. Desarrollo y Certificación de un Sistema Web Basado en Microservicios

Resumen

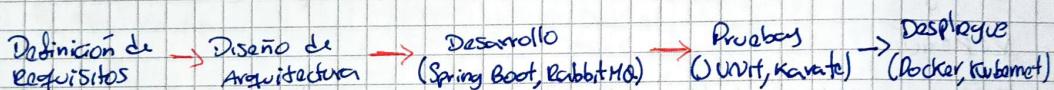
El artículo describe el desarrollo de un sistema web basado en microservicios utilizando un broker de mensajería para garantizar la comunicación entre ellos. Se utilizaron Spring Boot y WebFlux para construir servicios reactivos y una arquitectura distribuida que asegura la escalabilidad y el despliegue.

La implementación se organizó en fases, incluyendo diseño, desarrollo, pruebas y despliegue. Las pruebas se realizaron siguiendo la pirámide de Cohn, con herramientas como JUnit, Karate y JMeter para validar el rendimiento, la funcionalidad y la usabilidad. Finalmente, se implementaron pipelines de CI/CD para automatizar los despliegues en entornos como desarrollo, certificación y producción, aprovechando la infraestructura de Kubernetes y AWS.

Reflexión

En el artículo observamos la importancia de adoptar arquitecturas distribuidas y prácticas de DevOps en proyectos modernos. La integración de herramientas como Spring Boot y Webflux demuestra cómo las tecnologías actuales pueden facilitar el desarrollo de sistemas escalables y de alta calidad. Las pruebas automatizadas, además de garantizar la funcionalidad, evidencian cómo la planificación y el trabajo en equipos guiados por metodologías ágiles, pueden reducir los tiempos de desarrollo y mejorar la experiencia del usuario final. Esta investigación me inspira a valorar el papel del diseño reactivo y la automatización como pilares del desarrollo de software eficiente y sostenible.

Diagrama:



19. Desarrollo de Aplicaciones Basadas en Microservicios: Tendencias y Desafíos de Investigación

Resumen

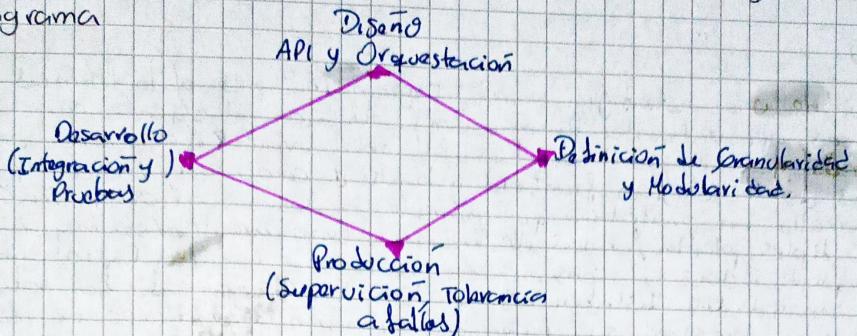
Se analiza las tendencias actuales y los desafíos del desarrollo de aplicaciones basadas en microservicios. A través de una revisión de literatura se identificaron los principales retos en áreas como la granularidad, modularización y refactoring de servicios, orquestación y monitoreo. Se destacan atributos de calidad como la escalabilidad y la calidad del servicio (QoS), los cuales son esenciales en sistemas distribuidos. Las prácticas de DevOps y el uso de herramientas específicas son claves para la implementación exitosa de estos sistemas. Con este enfoque se busca mejorar la eficiencia, garantizar actualizaciones rápidas y promover el uso de arquitecturas flexibles en entornos dinámicos como la nube y el internet de las cosas (IoT).

Reflexión

Se comprende cómo los microservicios están transformando el desarrollo de software. La posibilidad de desacoplar aplicaciones monolíticas en servicios más pequeños y gestionables no solo facilita el mantenimiento sino que también promueve la innovación tecnológica. Sin embargo, este granulado y garantizar la seguridad en sistemas distribuidos. Me pareció valiosa la aplicación de prácticas como DevOps,

que automatizan procesos y reducen los tiempos de entrega. Este trabajo resalta la importancia de la planificación y la elección de herramientas adecuadas para lograr sistemas escalables y de alta calidad.

Diagrama



20. Arquitectura Basada en Microservicios y DevOps para una Ingeniería de Software Continua

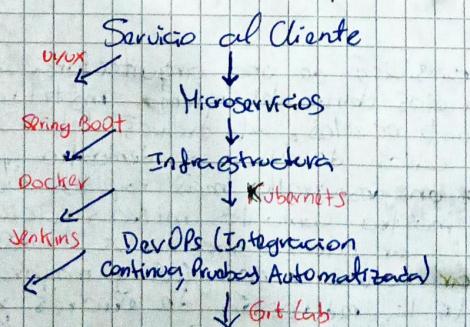
Resumen

En el artículo se presenta una arquitectura que combina microservicios y DevOps para mejorar la entrega e integración continua de software. Este enfoque propone un modelo escalable y colaborativo aplicable a proyectos complejos como SIGAP, un sistema integrado de gestión académica. La arquitectura incluye capas diferenciadas para datos, microservicios y presentación, con DevOps como una capa transversal que conecta desarrollo y operaciones. Además, se emplearon herramientas como GitHub, Heroku y Trello para la gestión y despliegue. Los resultados evidencian mejoras significativas en la calidad y productividad del software, destacando el papel de metodologías ágiles y tecnologías disruptivas en la formación de profesionales destacados para afrontar los retos del mercado actual.

Reflexión

Se analiza cómo las tecnologías como DevOps y los microservicios no solo optimizan el desarrollo de software sino que también fomentan una cultura de colaboración y mejora continua. Es interesante cómo un enfoque ágil y bien estructurado puede reducir significativamente la brecha entre desarrollo y operaciones, mejorando la calidad y la escalabilidad del software. Además, la integración de herramientas como Trello y Heroku demuestra la importancia de utilizar plataformas modernas que transporten la automatización y el trabajo en equipo. Creo que este modelo debería ser adoptado ampliamente en proyectos académicos y empresariales por su enfoque práctico y eficiente.

Diagrama



21. Estrategia de Integración de Diferentes Plataformas Web

Dinámicas Mediante un framework del Lado del Cliente.

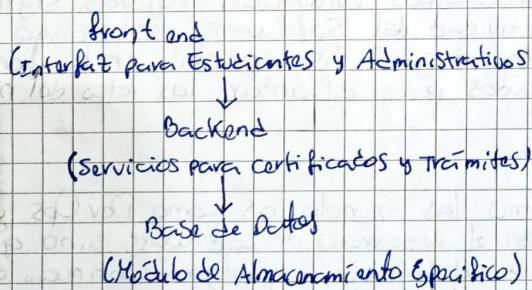
Resumen

Se propone una estrategia para integrar múltiples plataformas web dinámicas aprovechando frameworks del lado del cliente como Angular.js y la arquitectura RESTful en el servicio. A través del análisis de tecnologías frontend y backend como PHP, Java y Python, se desarrolla un prototipo que permite a los usuarios interactuar con múltiples plataformas de manera transparente. Esto mejora la escalabilidad, reduce el consumo de recursos y facilita la integración de funcionalidades distribuidas en distintas máquinas. La validación del prototipo demostró su eficiencia en términos de tiempo de respuesta y estabilidad, proporcionando una solución innovadora para superar las limitaciones de las aplicaciones monolíticas.

Reflexión

Podemos observar la importancia de diseñar aplicaciones web que se adapten a las necesidades actuales de escalabilidad y eficiencia. La separación de funcionalidades entre frontend y backend no solo mejora el rendimiento, sino que también permite manejar grandes volúmenes de usuarios sin problemas de congestión. Es particularmente interesante cómo el uso de frameworks del lado del cliente facilita la experiencia del usuario al tiempo que los microservicios en el backend aseguran un sistema robusto y adaptable. Esta estrategia resalta la importancia de adoptar tecnologías modernas y arquitecturas matadoras en el desarrollo de software.

Diagramas



22. Desarrollo Ágil del nuevo Sistema Institucional Basado en una Arquitectura Orientada a Microservicios.

Resumen

Se presenta una transición de una arquitectura monolítica a una basada en microservicios dentro del sistema institucional de la Universidad de Monte Morelos. A través del uso de métodos logísticos ágiles y puntos de casos de uso, se demostró que la nueva arquitectura facilita un desarrollo más rápido, modular y escalable. El módulo de caja mostró como los microservicios permiten implementar actualizaciones sin afectar otros componentes. Además, se destacó la capacidad de integrar nuevas tecnologías y mejorar la experiencia del usuario mediante la reducción de errores y tiempos de mantenimiento. Los resultados confirmaron que la arquitectura de microservicios es más eficiente en comparación con su predecesora.

Nombre

Fecha

día

mes

año

Profesor

Materia

Institución

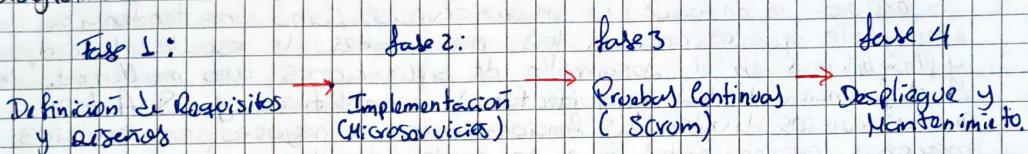
Curso

Nota

Reflexión

Permite descubrir como las arquitecturas de software pueden impactar significativamente la eficiencia y la escalabilidad de los sistemas. La transición de una arquitectura monolítica a una de microservicios no solo mejora los tiempos de desarrollo, sino que también fomenta una mayor flexibilidad en la integración de tecnologías. Es imprescindible el uso de metodologías ágiles y puntos de casos de uso para evaluar y validar el impacto de esta transformación. Se resalta la importancia de planificar arquitecturas que sean adaptables a largo plazo, especialmente en instituciones académicas, donde las necesidades tecnológicas cambian constantemente.

Diagrama.



2.3. Implementación de Microservicios para la Evaluación de Programas

Basado en Casos de Prueba

Resumen.

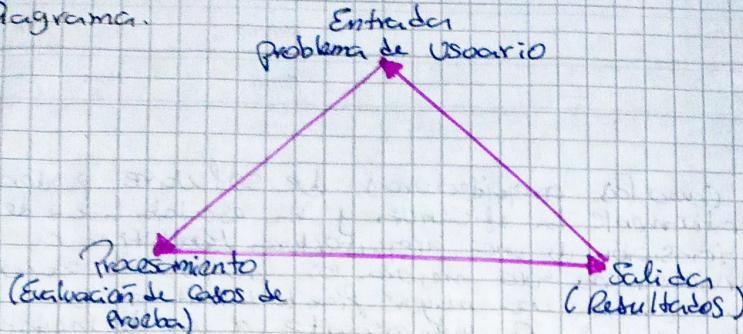
Se presenta un modelo virtual JUZ para evaluar programas basados en casos de prueba. Implementado mediante arquitectura de microservicios. Con este enfoque se busca superar las limitaciones de los sistemas monolíticos optimizando el rendimiento y escalabilidad. La metodología utilizada fue Scrum, destacando la segmentación del desarrollo en Sprints, con tareas específicas para cada microservicio. Los resultados indican que el sistema responde eficientemente hasta 50 peticiones concurrentes, mejorando significativamente el tiempo de respuesta y la capacidad de gestión de capacidades y solicitudes. Incluye componentes como un servicio principal, un servicio evaluador, y un servicio de interfaz de usuario, cada uno especializado en sus funciones. Demostran la flexibilidad de esta arquitectura.

Reflexión

Comprendemos la importancia de adoptar arquitecturas modernas como los JUZ, que son sistemas críticos. Los microservicios no solo ofrecen una solución escalable sino que también permiten mantener los componentes actualizados de forma independiente. Me pareció interesante como la metodología Scrum estructura el desarrollo, garantizando entregas incrementales y funcionales. Además, la capacidad del sistema para soportar múltiples solicitudes concurrentes resalta la eficiencia y relevancia de este enfoque en aplicaciones.

académicos y de evaluación.

Diagramas.



24. Un Acercamiento a los Microservicios: Beneficios, Retos y Consideraciones para su Implementación

Resumen.

Se explora el enfoque de microservicios como una tendencia emergente que responde a las necesidades de escalabilidad y flexibilidad en el desarrollo de aplicaciones web modernas. Al diferenciar dentro arquitecturas monolíticas y SOA, los microservicios dividen las funcionalidades de negocio en servicios pequeños independientes que se pueden desplegar de manera autónoma. El enfoque trae beneficios como la resiliencia, el alineamiento organizacional y la heterogeneidad tecnológica. Pero también enfrenta desafíos en comunicación, desempeño y seguridad. Se destaca el uso de prácticas como Domain-Driven Design (DDD) para guiar el diseño, así como el uso de herramientas como APIs RESTful y colas de mensajes para la integración.

Reflexión

Hace Comprender como los microservicios de software han revolucionado su desarrollo al permitir soluciones más modulares y escalables, es interesante como esta arquitectura facilita la innovación tecnológica y reduce los riesgos asociados al mantenimiento de sistemas grandes y complejos. Es importante destacar que implementar microservicios requiere preparación técnica, coordinación entre equipos y herramientas adecuadas para mejorar la comunicación y el monitoreo. Es factible considerar como los conceptos de resiliencia y desacoplamiento pueden aplicarse más allá del software, en la gestión de proyectos y sistemas organizados.

Diagramas

Arquitectura Monolítica
Una sola unidad funcional
Dependencia alta
Dificultad para escalar
Menor flexibilidad

Comparación de Arquitecturas Monolítica vs Microservicios.

Arquitectura de Microservicios

- Servicios pequeños e independientes
- Alta Escalabilidad
- Despliegue flexible
- Resiliencia Mejorada

29. Aplicación Web para el servicio de Tramites Académicos de la UNACH usando una Arquitectura Basada en Microservicios

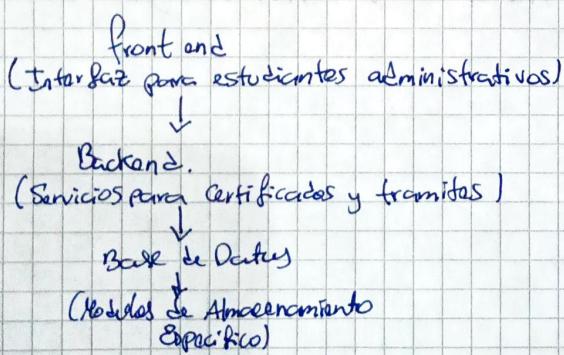
Resumen.

Se describe el desarrollo de una aplicación web para gestionar trámites académicos en la Universidad Nacional de Chimborazo (UNACH) utilizando arquitecturas de microservicios y la metodología Scrum. Incluye módulos como la generación de certificados académicos y recibo esto digitales, logrando una gestión descentralizada y eficiente durante las pruebas realizadas con JMeter, se obtuvo una eficacia del 100% tiempo de respuesta promedio 74.557 ms y un uso de recurso del 25%, superando los estándares de modelo TPS. El sistema permite una escalabilidad y modularidad que facilita el mantenimiento y la integración con tecnologías modernas optimizando los procesos administrativos y académicos.

Reflexión

Observamos la importancia de aplicar tecnologías modernas como los microservicios en el ámbito académico. La capacidad de gestionar procesos de forma eficiente y descentralizada representa un gran avance frente a los sistemas monolíticos. Es sorprendente como la metodología Scrum estructura el desarrollo, permitiendo cumplir con los requerimientos funcionales de manera ágil. Además, el uso de herramientas como JMeter garantiza que las aplicaciones no solo cumplan con estándares de calidad, sino que también optimicen recursos tecnológicos. Este enfoque se puede replicar en instituciones educativas para mejorar la experiencia de profesores y alumnos.

Diagrama.



26. Estrategia para la Implementación de Modelos de Aprendizaje Automático usando Microservicios en Ciudades Inteligentes

Resumen.

Se propone una estrategia para implementar modelos de aprendizaje automático (ML) en el contexto de la seguridad informática en ciudades inteligentes utilizando arquitecturas basadas en microservicios. Incluye dos microservicios clave: preprocesamiento y predicción, diseñados para ejecutar, almacenar y evaluar modelos ML. Se utilizan tecnologías como Kafka para la comunicación asíncrona y herramientas como Python, Java y MongoDB para garantizar escalabilidad y eficiencia. El sistema se probó con base de datos como Bolt-Lot y UNIS W-NBIS, logrando procesar altas demandas de tráfico en tiempo real y detectando posibles ataques con precisión. Las arquitecturas modernas pueden optimizar la seguridad en entornos urbanos.

27.- Estrategia para la implementación de modelos de aprendizaje automático usando microservicio en ciudades inteligentes. Resumen.

Propone una estrategia para implementar modelos de aprendizaje automático (ML) en el contexto de la segunda oleada informática en ciudades inteligentes, utilizando arquitecturas basadas en microservicio. La propuesta incluye dos microservicios clave: procedimientos y predicción, diseñados para ejecutar, almacenar y evaluar modelos ML. Además se utilizan tecnologías como Kafka para la comunicación asíncrona y健scimienter como