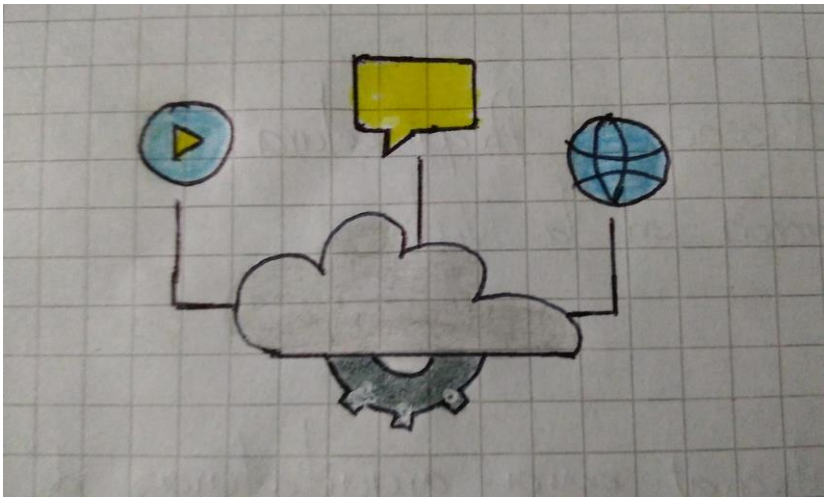
Soy desarrolladora full stack con una gran pasión por la tecnología y un sólido conocimiento en diversas áreas del sector. Desde que comencé en este mundo, he sido constante en mi aprendizaje y dominio de distintos lenguajes de programación y herramientas, tales como Java, JavaScript, PHP, Kotlin, bases de datos relacionales, desarrollo para Android, entre otros. En cada proyecto en el que participo, me enfoco en ofrecer lo mejor de mí para crear aplicaciones eficientes y funcionales. Además, me gusta mantenerme actualizada con los cambios y avances en el campo de la tecnología, para poder aprender e implementar las novedades en mi trabajo diario.

Mariana González Calderón

Modelado y Verificación de Patrones de Diseño de Arquitectura de Software para Entornos de Computación en la Nube

El artículo propone un entorno de diseño integral para arquitecturas de software orientadas a aplicaciones web, especialmente en contextos de computación en la nube. Este entorno aborda los principales desafíos del diseño arquitectónico, utilizando un metamodelo de componentes arquitectónicos como base. Incluye herramientas gráficas para la creación y verificación de patrones de diseño, asegurando su correcta implementación y promoviendo diseños de calidad. Además, se enfoca en facilitar el trabajo de los arquitectos al proporcionar guías y sugerencias basadas en decisiones arquitectónicas previas. El objetivo es optimizar la productividad y garantizar que las arquitecturas cumplan con los atributos de calidad requeridos.

El diseño arquitectónico es una disciplina compleja que equilibra requisitos funcionales y no funcionales para crear sistemas robustos y eficientes. La computación en la nube añade nuevos desafíos debido a su naturaleza dinámica y la necesidad de adaptar patrones de diseño existentes. Este artículo destaca la importancia de combinar herramientas prácticas y marcos teóricos sólidos para enfrentar estas dificultades. La propuesta presentada es valiosa porque no solo ayuda a construir arquitecturas eficientes, sino que también fomenta un enfoque sistemático hacia la calidad del software, algo esencial en un entorno tecnológico.

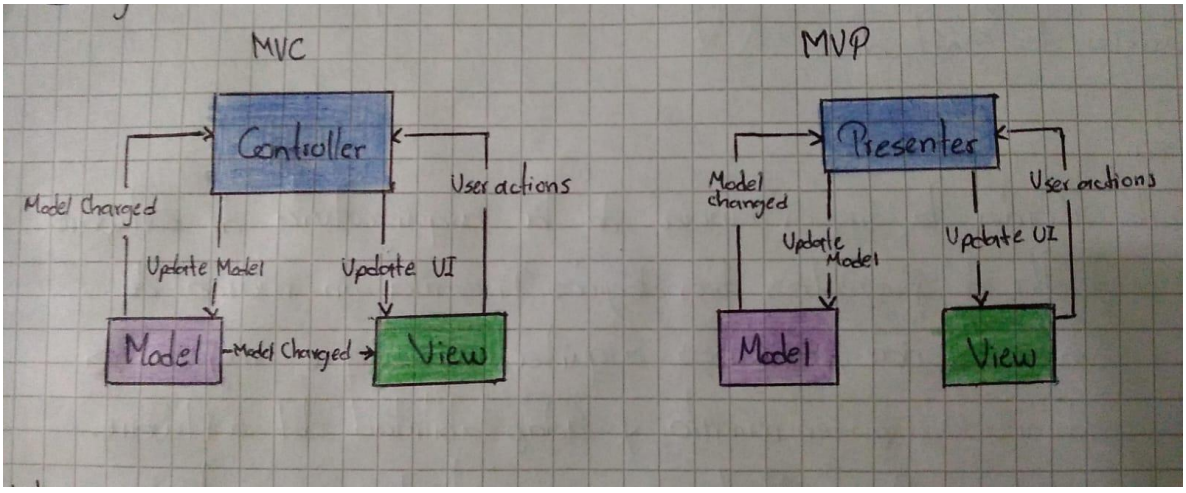


Blas et al. (2019)

Marco de Trabajo para Seleccionar un Patrón Arquitectónico en el Desarrollo de Software

El artículo aborda la importancia de la arquitectura de software en el desarrollo de proyectos tecnológicos. Presenta un marco de trabajo para seleccionar patrones arquitectónicos que mejoran la calidad, rendimiento, mantenimiento y adaptabilidad del software. Este marco guía a desarrolladores y arquitectos para tomar decisiones basadas en características del proyecto (como tipo de desarrollo y requisitos clave). Se describen patrones como MVC, MVP, Microservicios y Arquitectura en la Nube, y su idoneidad según el contexto y necesidades del software. Finalmente, se valida el marco mediante un caso práctico en el que el usuario selecciona opciones para obtener recomendaciones arquitectónicas personalizadas.

La arquitectura de software es crucial para el éxito de los proyectos, ya que influye directamente en su escalabilidad, mantenibilidad y rendimiento. Este marco de trabajo no solo organiza el conocimiento sobre patrones arquitectónicos, sino que también empodera a los desarrolladores para tomar decisiones más informadas y eficientes. La propuesta resalta la necesidad de planificar antes de codificar, evitando reprocesos y promoviendo prácticas sostenibles. En un entorno donde las tecnologías avanzan rápidamente, adoptar metodologías como esta es esencial para cumplir con las demandas del mercado y garantizar la calidad del producto final.

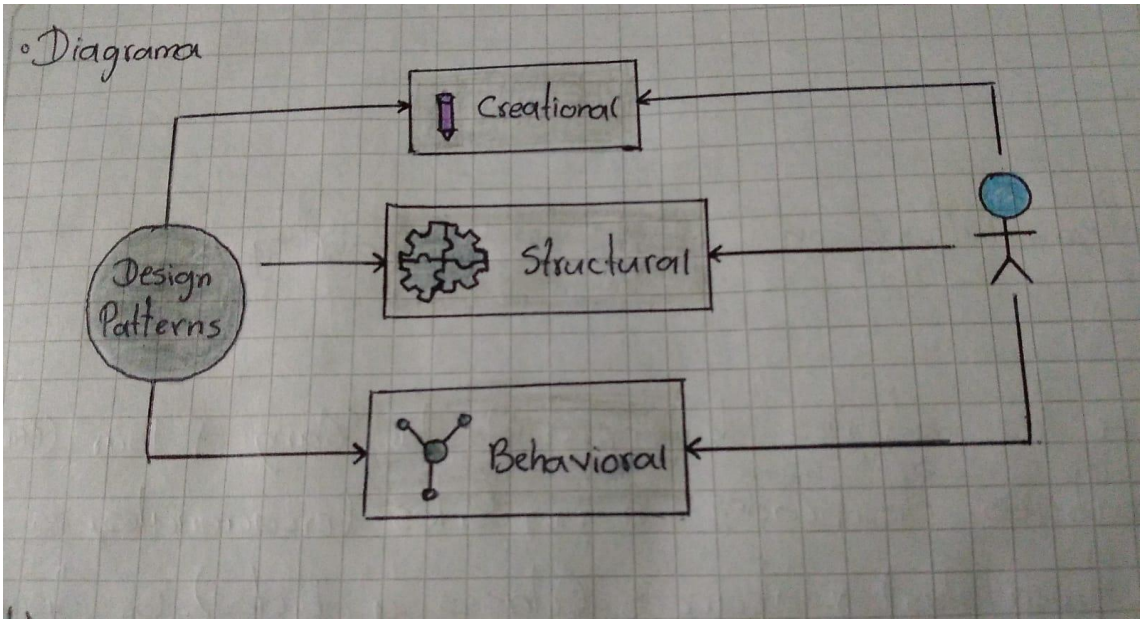


(*Marco de Trabajo Para Seleccionar un Patrón - ProQuest*, s. f.)

Impact of Design Patterns on Software Maintainability

El artículo examina los patrones de diseño de la "Gang of Four" (GoF), que incluyen 23 patrones clasificados en propósitos (creacionales, estructurales, comportamentales) y alcances (clases u objetos). Se discute su impacto en la mantenibilidad del software a través de estudios empíricos, con resultados inconsistentes. Aunque algunos hallazgos destacan beneficios en ciertos contextos, otros indican que los patrones no mejoran la mantenibilidad o pueden introducir complejidad innecesaria. Además, se proponen herramientas y enfoques para ayudar a los diseñadores a seleccionar patrones adecuados basados en atributos de calidad y tamaño del sistema.

El uso de patrones de diseño GoF ofrece ventajas como reusabilidad y estructura, pero su impacto en la mantenibilidad varía según el contexto y la experiencia del diseñador. Esto subraya la importancia de comprender profundamente los patrones y evaluar su aplicabilidad en cada caso. Diseñadores deben equilibrar sus decisiones considerando tanto los beneficios como las posibles complejidades que puedan introducir. El uso adecuado de estos patrones puede mejorar la cohesión y reducir el acoplamiento entre los componentes, facilitando la comprensión del sistema a largo plazo.

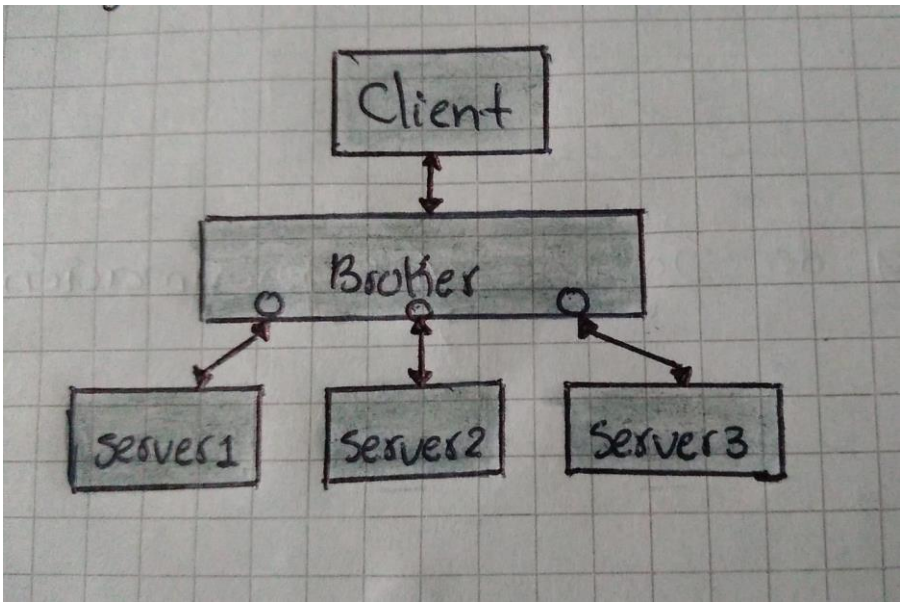


Alghamdi y Qureshi (2014)

Lenguajes de Patrones de Arquitectura de Software: Una Aproximación Al Estado del no 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 profunda sobre la evolución de la ingeniería de software, subrayando la relevancia de los patrones de diseño y los lenguajes de patrones en la práctica actual. Al abordar cómo estos conceptos han transformado la manera en que los desarrolladores enfrentan problemas recurrentes, se destaca su papel crucial en la creación de soluciones estandarizadas que mejoran la eficiencia y mantenibilidad de los sistemas. La estandarización no solo facilita la implementación de prácticas recomendadas, sino que también fomenta la colaboración y comunicación entre equipos, lo cual es esencial en entornos de desarrollo ágiles y dinámicos.

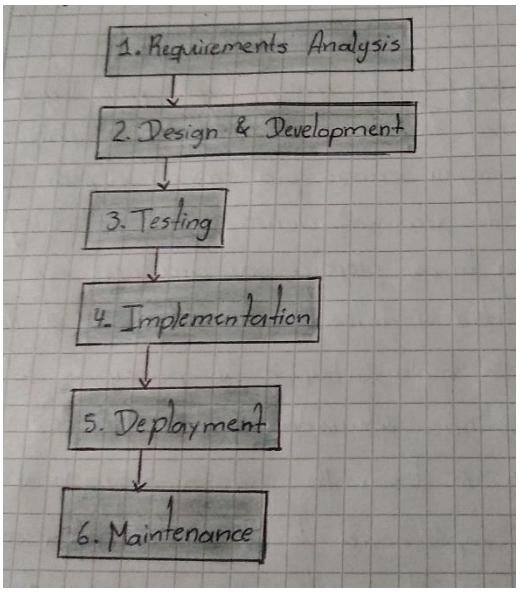


(2008). *Redalyc*

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 sobre la importancia de un código limpio y de alta calidad me invita a reflexionar sobre un aspecto fundamental en el desarrollo de software que a menudo se pasa por alto: la calidad del código no es solo un objetivo técnico, sino una estrategia esencial para el éxito a largo plazo de cualquier proyecto. Al ofrecer consejos sobre buenas prácticas y arquitecturas efectivas, la guía se convierte en un recurso valioso para desarrolladores que buscan mejorar la legibilidad y mantenibilidad de su trabajo. La implementación de principios sólidos de diseño y la elección de metodologías ágiles no solo optimizan el proceso de desarrollo, sino que también fomentan un ambiente de trabajo más colaborativo y eficiente.



(2021). *Tecnología Investigación y Academia*.

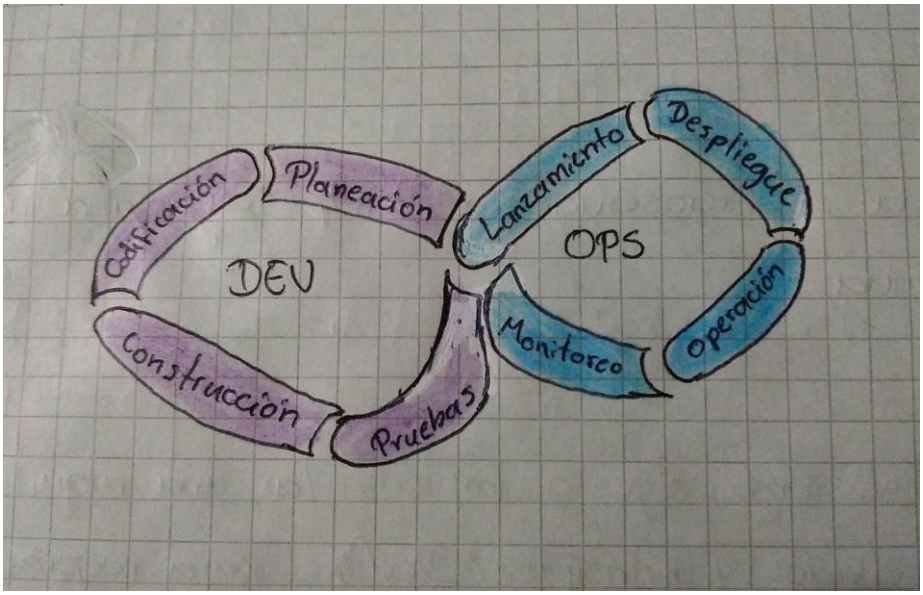
Arquitectura basada en Microservicios y DevOps para una ingeniería de software continua

El proyecto SIGAP tuvo como objetivo implementar una arquitectura basada en microservicios y principios DevOps para mejorar los procesos de desarrollo de software y aumentar la productividad del equipo. La arquitectura de microservicios permitió dividir la aplicación en servicios independientes, lo que facilitó su mantenimiento y escalabilidad. Además, este enfoque permitió realizar entregas de software de manera ágil y controlada, adaptando los flujos de trabajo a un entorno más eficiente. La integración de DevOps promovió la colaboración entre desarrollo y operaciones, automatizando procesos y mejorando la calidad del software. Se logró una mayor flexibilidad para adaptarse a cambios y requerimientos del cliente, lo que permitió ciclos de entrega más rápidos y confiables. El enfoque arquitectónico utilizado no solo mejoró el rendimiento de SIGAP, sino que también demostró ser aplicable a otros dominios.

La experiencia del proyecto SIGAP me ha hecho reflexionar sobre la importancia de adoptar una arquitectura de microservicios y los principios de DevOps en el desarrollo de software. Al dividir aplicaciones complejas en servicios independientes, se mejora la escalabilidad y el mantenimiento, lo que permite implementar cambios

más rápidamente y con menor riesgo. He visto cómo esta estructura no solo optimiza el rendimiento del software, sino que también fomenta la innovación continua.

Además, he apreciado cómo la integración de prácticas de DevOps transforma la colaboración entre los equipos de desarrollo y operaciones. Este enfoque crea un flujo de trabajo más eficiente y alineado.



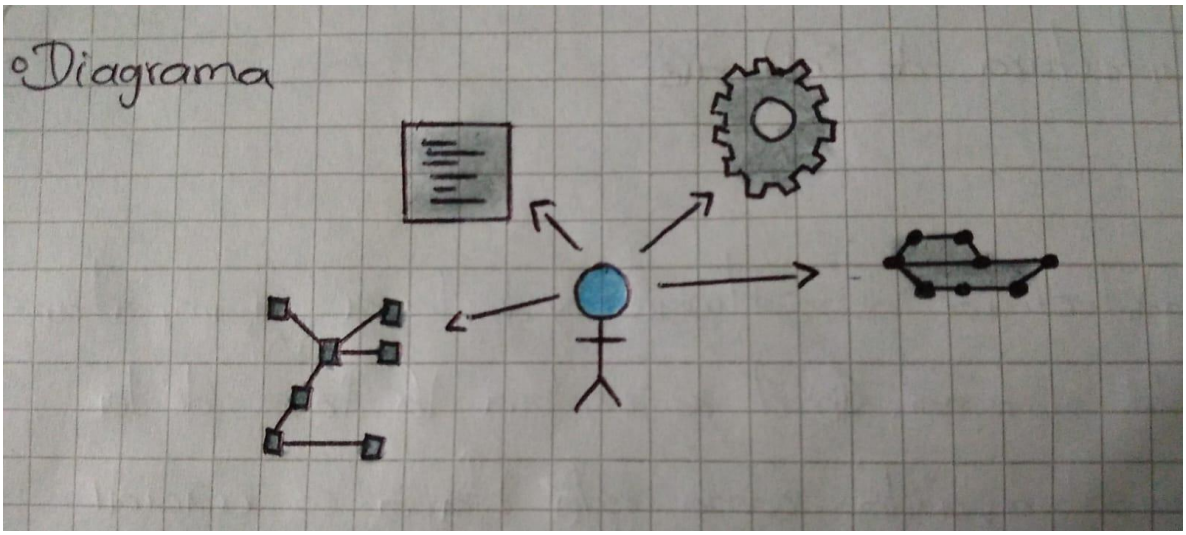
(2021). *Revista Iberoamericana de Ciencia, Tecnología y Sociedad*.

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.

El artículo sobre la arquitectura de software me lleva a reflexionar sobre el papel crucial que desempeña en el ciclo de vida del sistema. La fase de arquitectura es fundamental, ya que establece las bases sobre las cuales se construirá el software,

asegurando que la estructura y los componentes se alineen con los objetivos del proyecto. Me parece notable cómo cada etapa del ciclo de la arquitectura, desde la planificación hasta la evaluación continua, es interdependiente y contribuye a la evolución del sistema. Este enfoque integrado permite realizar ajustes y mejoras a lo largo del desarrollo, lo que resulta esencial en un entorno en constante cambio.



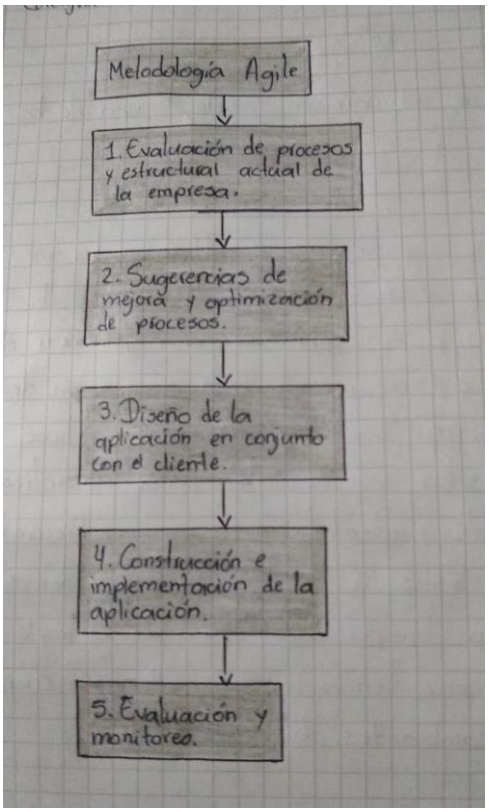
https://www.fceia.unr.edu.ar/~mcristia/Introduccion_a_la_Arquitectura_de_Software.pdf

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.

La investigación sobre la interrelación entre arquitecturas de software y metodologías ágiles me lleva a reflexionar sobre la importancia de una integración efectiva para optimizar el desarrollo de software. La colaboración entre estos dos aspectos es fundamental, ya que una arquitectura bien diseñada no solo sienta las bases para un desarrollo eficiente, sino que también permite a los equipos adaptarse rápidamente a los cambios y requisitos que surgen en entornos ágiles. Considero esencial prestar atención a los "requisitos significativos para la arquitectura", ya que estos garantizan la flexibilidad necesaria para responder a las dinámicas del proyecto.



(2020). *Revista de Tecnología y Software*, 15(3), 45-60.

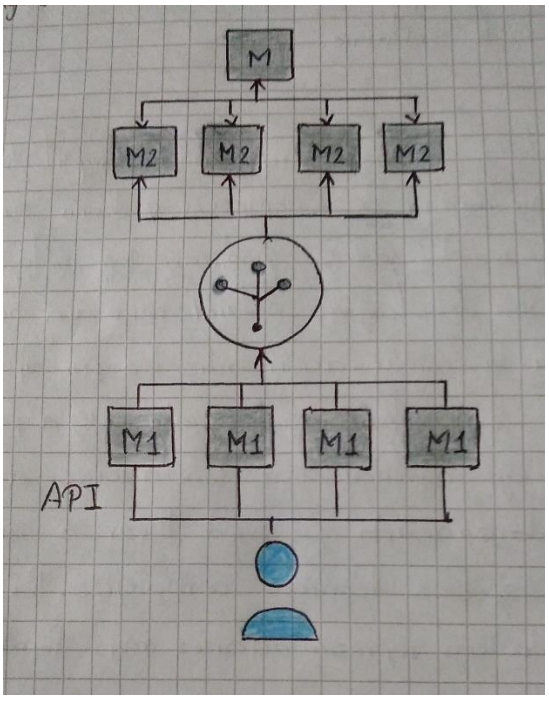
Revisión de elementos conceptuales para la representación de las arquitecturas de referencias de software

El artículo examina las distintas formas de representar arquitecturas de software, subrayando la importancia de descomponer y definir sus elementos esenciales para

lograr claridad. Se analizan los métodos más comunes, destacando el uso del lenguaje natural como una herramienta accesible que facilita la comunicación entre los participantes en el desarrollo de software. Sin embargo, se señala que, a pesar de su accesibilidad, el lenguaje natural tiene limitaciones en términos de precisión y estandarización, lo que puede llevar a ambigüedades y malentendidos en aspectos técnicos complejos. Por ello, se recomienda complementarlo con representaciones más formales para permitir un análisis más riguroso.

Al leer el artículo, reflexioné sobre la importancia de representar adecuadamente las arquitecturas de software. Descomponer y definir claramente sus elementos esenciales es vital para el éxito de cualquier proyecto. El lenguaje natural, al ser accesible, facilita la comunicación entre todos los involucrados, permitiendo que tanto desarrolladores como no desarrolladores comprendan la estructura general del sistema. Sin embargo, también es cierto que puede generar ambigüedades que, en un entorno técnico, pueden resultar problemáticas.

Por eso, me parece acertado complementar el lenguaje natural con representaciones más formales.

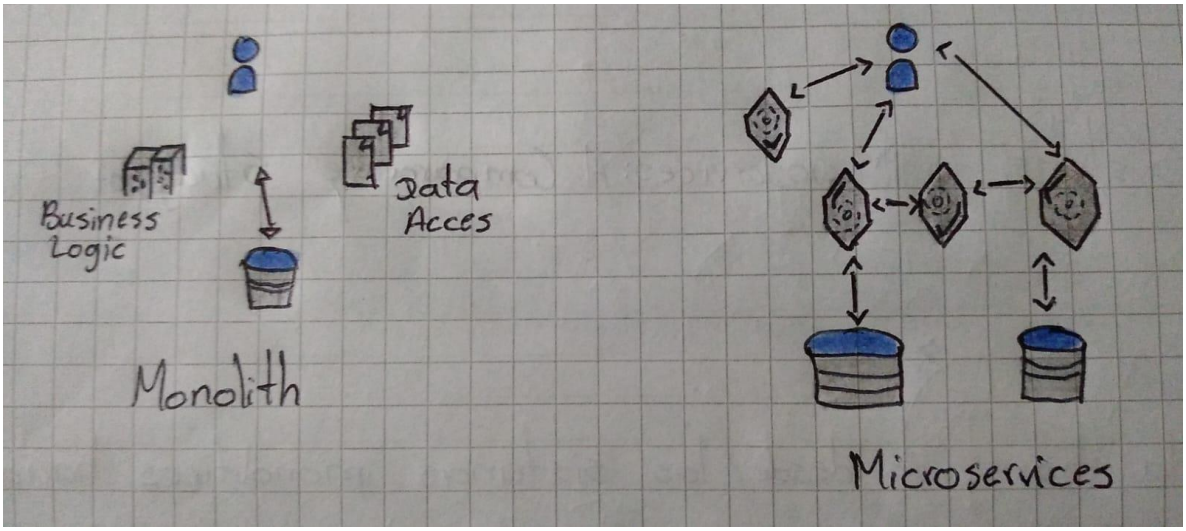


(2019). *Revista Cubana de Ciencias Informáticas*.

From Monolithic Systems to Microservices: A Comparative Study of Performance

El artículo aborda la transición desde los sistemas monolíticos hacia los microservicios, explorando las ventajas y desventajas de ambos enfoques y los retos que surgen en este proceso. Se realiza un estudio de caso que compara el desempeño de una aplicación web implementada en una arquitectura monolítica con otra en una arquitectura de microservicios. Los resultados evidencian que los microservicios presentan una mayor eficiencia en términos de consumo de CPU, utilización de memoria y velocidad de escritura en disco. Sin embargo, se identifican ciertas desventajas en los microservicios, como la complejidad de la gestión y una mayor probabilidad de fallos en las comunicaciones.

El artículo destaca la transición de arquitecturas monolíticas a microservicios, subrayando que, aunque los microservicios ofrecen mejoras en eficiencia y escalabilidad, también introducen desafíos como la complejidad de gestión y fallos en la comunicación. Esto resalta la importancia de evaluar las necesidades específicas de cada proyecto. Mientras que los microservicios son ideales para aplicaciones de alta demanda, la arquitectura monolítica puede ser más adecuada para sistemas menos complejos. La elección debe basarse en el contexto y los objetivos del software, enfatizando que no hay una solución única para todos los casos.



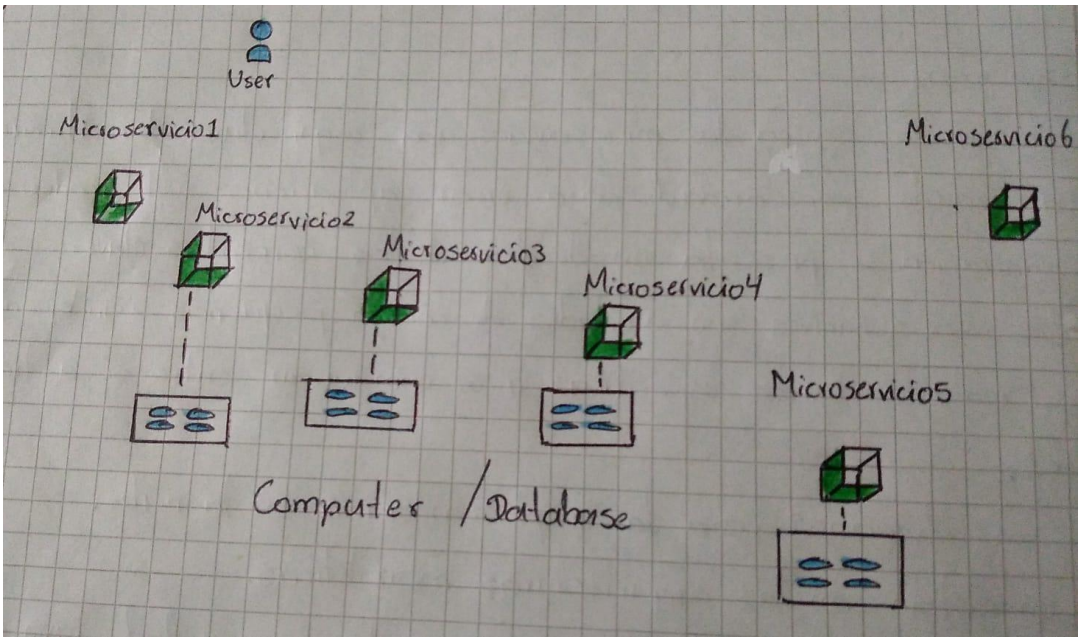
https://www.proquest.com/docview/2437268966/F05CEE9B819045B5PQ/1?accountid=31491&sourcetype=Scholarly%20Journals

What are microservices? Your next software architecture

Este artículo explica los microservicios, una arquitectura de software que ha ganado popularidad para crear aplicaciones web modernas. A continuación, se presenta un desglose de los conceptos clave:

- Los microservicios son unidades de código pequeñas e independientes que realizan tareas específicas y se comunican entre sí a través de API.
- Son adecuados para la implementación en la nube.
- Contrastan con las arquitecturas monolíticas.
- Los microservicios son populares en la comunidad Java, con marcos como Spring Boot y Spring Cloud que facilitan su desarrollo.
- Los contenedores, como Docker, permiten una implementación y gestión más eficientes de los microservicios.

Este artículo destaca cómo los microservicios han revolucionado el desarrollo de aplicaciones web al permitir un enfoque más modular y flexible. Su capacidad para escalar y adaptarse a entornos en la nube, junto con el uso de tecnologías como Docker, facilita la creación de aplicaciones complejas. La comparación con las arquitecturas monolíticas resalta la necesidad de evolucionar en las prácticas de desarrollo para abordar las demandas actuales del mercado. Este enfoque modular no solo mejora la eficiencia en el desarrollo, sino que también facilita el mantenimiento y la actualización de las aplicaciones sin afectar a todo el sistema. Además, la posibilidad de implementar microservicios de manera independiente permite un ciclo de vida de desarrollo más ágil y dinámico.

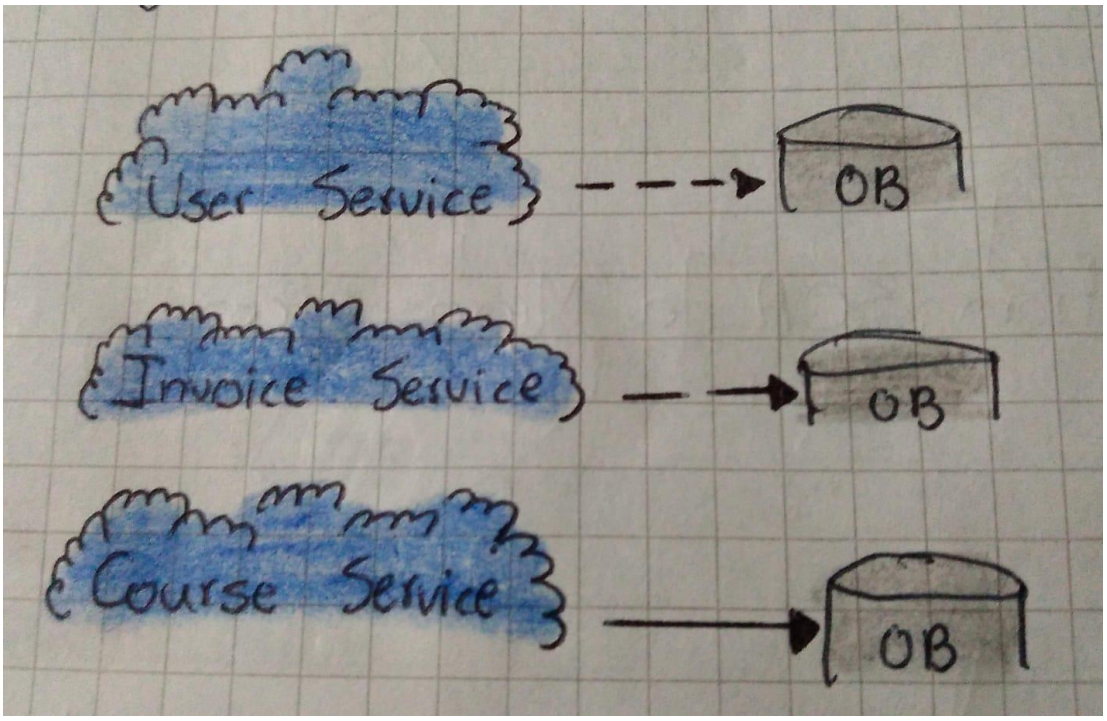


Blas et al. (2019)

Assessing the Impact of Migration from SOA to Microservices Architecture

El Artículo explora la transición de aplicaciones basadas en SOA (Arquitectura Orientada a Servicios) a microservicios, un estilo arquitectónico adoptado por empresas como Netflix y Twitter. Aunque el enfoque de microservicios presenta ventajas significativas, como la escalabilidad y flexibilidad, también plantea desafíos relacionados con el rendimiento y la complejidad durante la migración. El estudio utiliza métricas como la probabilidad de propagación de cambios y la estabilidad arquitectónica para evaluar el impacto de esta migración. Los resultados muestran que, a pesar de los desafíos, los microservicios ofrecen beneficios sustanciales y son especialmente adecuados para aplicaciones empresariales grandes.

El artículo destaca la evolución de las aplicaciones de SOA a microservicios, un cambio adoptado por líderes como Netflix y Twitter. Aunque los microservicios ofrecen ventajas claras en escalabilidad y flexibilidad, también conllevan retos en rendimiento y complejidad durante la migración. Al evaluar esta transición mediante métricas específicas, los resultados indican que, a pesar de los desafíos, los beneficios de los microservicios son significativos, haciéndolos particularmente adecuados para grandes aplicaciones empresariales. Esta reflexión subraya la importancia de sopesar tanto los pros como los contras al considerar una migración arquitectónica.



https://www.proquest.com/docview/2921192054/72CF03D124D94837PQ/1?accountid=31491&sourcetype=Scholarly%20Journals

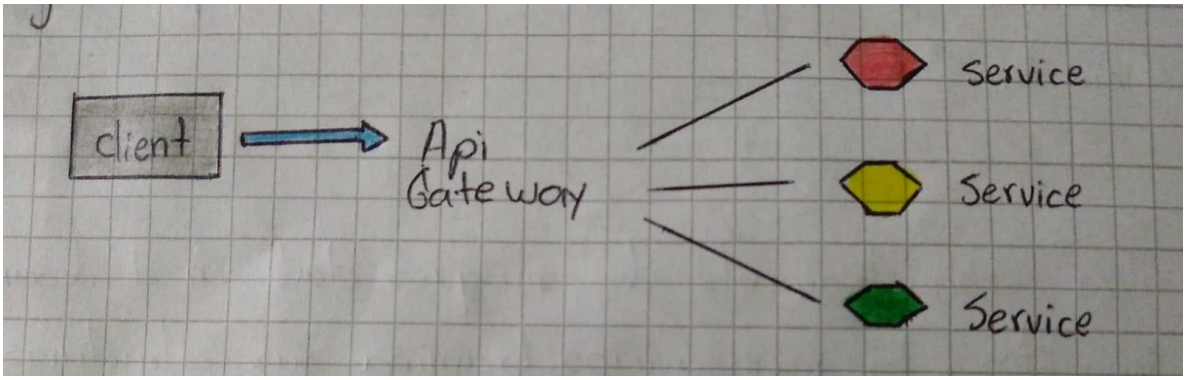
Monolitos vs. Microservicio en Arquitectura de Software: Perspectivas para un Desarrollo Eficiente

Este estudio analiza exhaustivamente dos enfoques clave en el desarrollo de software: los monolitos y los microservicios. Examina sus estructuras, interacciones de componentes, desafíos, ventajas y su implementación práctica en entornos reales. También evalúa casos de estudio de empresas líderes que han adoptado o migrado entre ambas arquitecturas. El análisis incluye el despliegue, mantenimiento y escalabilidad a largo plazo, proporcionando una visión completa para que los profesionales puedan tomar decisiones informadas sobre qué arquitectura utilizar, considerando las implicaciones técnicas, estratégicas y de costos a largo plazo.

Este estudio ofrece un análisis profundo de dos enfoques en el desarrollo de software: monolitos y microservicios. Al examinar sus estructuras, ventajas y desafíos, se proporciona un contexto práctico mediante casos de estudio de empresas que han migrado entre ambas arquitecturas.

El enfoque en despliegue, mantenimiento y escalabilidad permite a los profesionales evaluar las implicaciones técnicas, estratégicas y de costos a largo plazo. En

resumen, este análisis es esencial para que los desarrolladores tomen decisiones informadas y elijan la arquitectura más adecuada según las necesidades de sus proyectos.



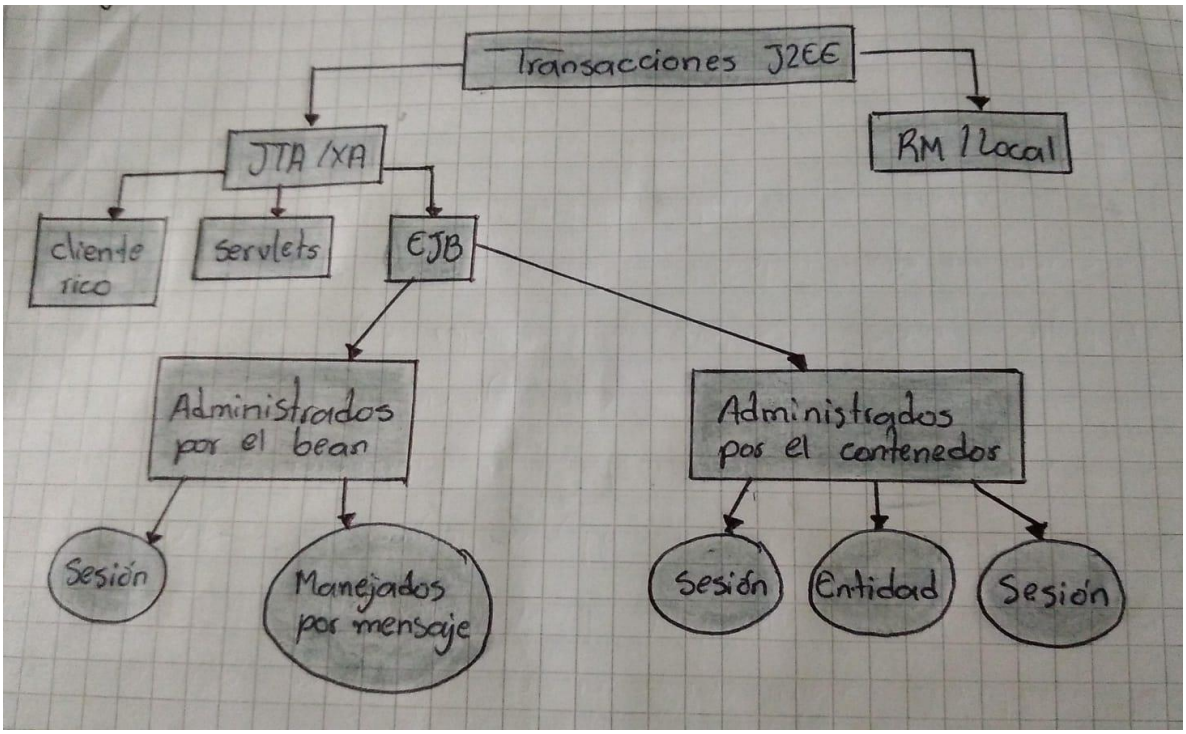
Colombero et al. (2024)

Arquitectura de Software y Entornos de Trabajo (Frameworks) de Contenedores Ligeros

El objetivo del trabajo radica explorar con respecto a las arquitecturas de software J2EE y los contenedores ligeros, dedicando especial atención a las características de los POJOs (Plain Old Java Objects). Pese a que no serían considerados como una nueva tecnología, los POJOs llegarían a ser considerados como una filosofía que permite a los desarrolladores que trabajen en la lógica de negocio, a la que determinan que ni la infraestructura ni el framework influyan en la determinación del diseño. Por otro lado, se establece como importante el garantizar que el dominio se mantenga aislado del resto del sistema.

Este trabajo pone de manifiesto la manera en la que las modernas arquitecturas de software están comenzando a adoptar así un menor intrusismo y mayor flexibilidad

de manera que los desarrolladores puedan centrarse en lo importante, la lógica de negocio. La filosofía de los POJOs pone de manifiesto una tendencia que va en aumento de querer reducir la complejidad de los OO frameworks y tecnologías más pesados para poder dejar al dominio de la aplicación algo aislado y desacoplado, aspecto este que ayudara a conservar la mantenibilidad del sistema o su escalabilidad.



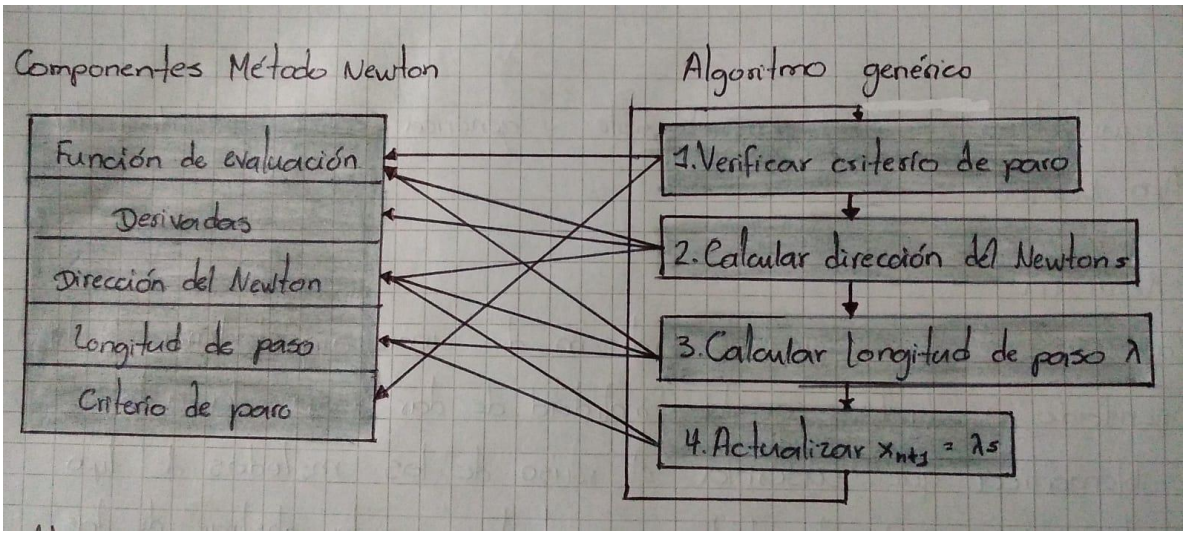
Tesis_Damian_Ciocca.pdf

Arquitectura de software flexible y genérica para métodos del tipo Newton

La tesis aborda el análisis y diseño de una arquitectura de tipo orientado a objetos con la finalidad de dar respuesta a la problemática que suscita el reúso de los métodos del tipo Newton, correspondiendo dicha arquitectura a la abstrae de las principales características de los mismos métodos mediante la aplicación de patrones de tipo arquitectura y de diseño de la misma arquitectura. Dicha arquitectura está pensada para su uso en diversos problemas no lineales que juegan

en calidad de interfaz genérica entre los distintos métodos del tipo Newton. En el documento se exponen las razones que sustentan las decisiones de diseño y representación de las dificultades que se presentan cuando se debe construir una arquitectura del tipo orientado a objetos.

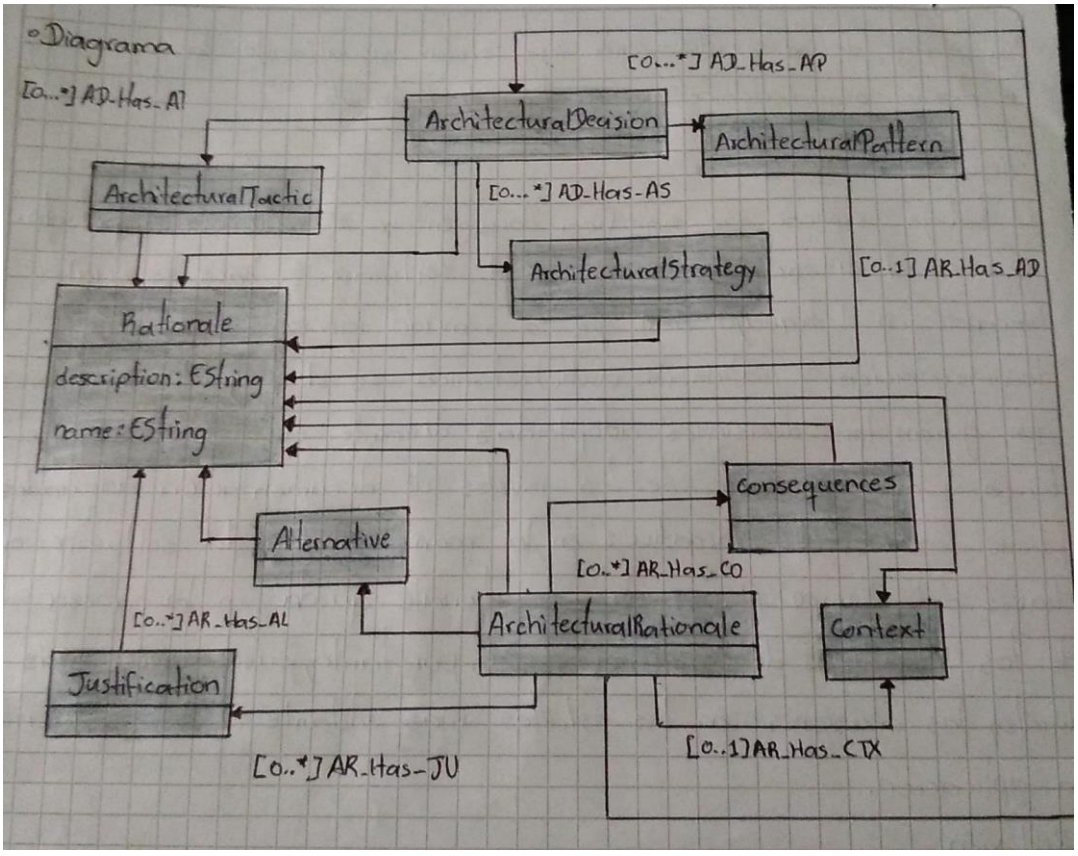
Esta tesis desarrolla un reto habitual en la elaboración de software: la reutilización del código. En concreto, se preocupa de cómo conseguir mejorar la flexibilidad y la extensibilidad de los métodos numéricos, como por ejemplo los métodos de Newton, mediante una arquitectura orientada a objetos. Con la adopción de los patrones de diseño y de la arquitectura, es posible conseguir una estructura que responde la situación existente, y que al mismo tiempo permite integrar nuevos métodos o problemas, y no hace necesario reestructurar el sistema completo.
Un aspecto determinante de esta solución consiste en la creación de una interfaz genérica que abstrae los detalles relacionados con cada uno de los métodos de Newton.



Un lenguaje de modelado para representar visualmente las decisiones de diseño arquitectónico y su rationale: Rationale

El trabajo explica qué significa documentar el rationale arquitectónico en el diseño de sistemas de software. Se hace especial hincapié en el trabajo ágil que a menudo ignora ese tipo de documentación. El rationale archictectónico es el conjunto de razones que guían las decisiones adoptadas durante el diseño arquitectónico, y que careciendo de los niveles de documentación adecuados puede tener efectos negativos en la mantenibilidad del software a través del tiempo. A menudo el rationale queda en la cabeza de los creadores del sistema y eso, por tanto, puede hacer más difíciles las decisiones en los estados más adelante de la evolución de la aplicación.

La documentación del rationale arquitectónico se convierte en algo crítico en su desarrollo, en particular cuando se están empleando paradigmas ágiles donde la práctica de la documentación más exhaustiva se antepone a una flexibilidad y rapidez del trabajo a realizar. El rationale arquitectónico actúa como una base compositiva que permite conocer las decisiones que han sido tomadas en el diseño, y que si están bien documentadas ayudan a que el sistema se mantenga a largo plazo y siga siendo sostenible. El presente trabajo muestra un reto habitual en contextos ágiles.

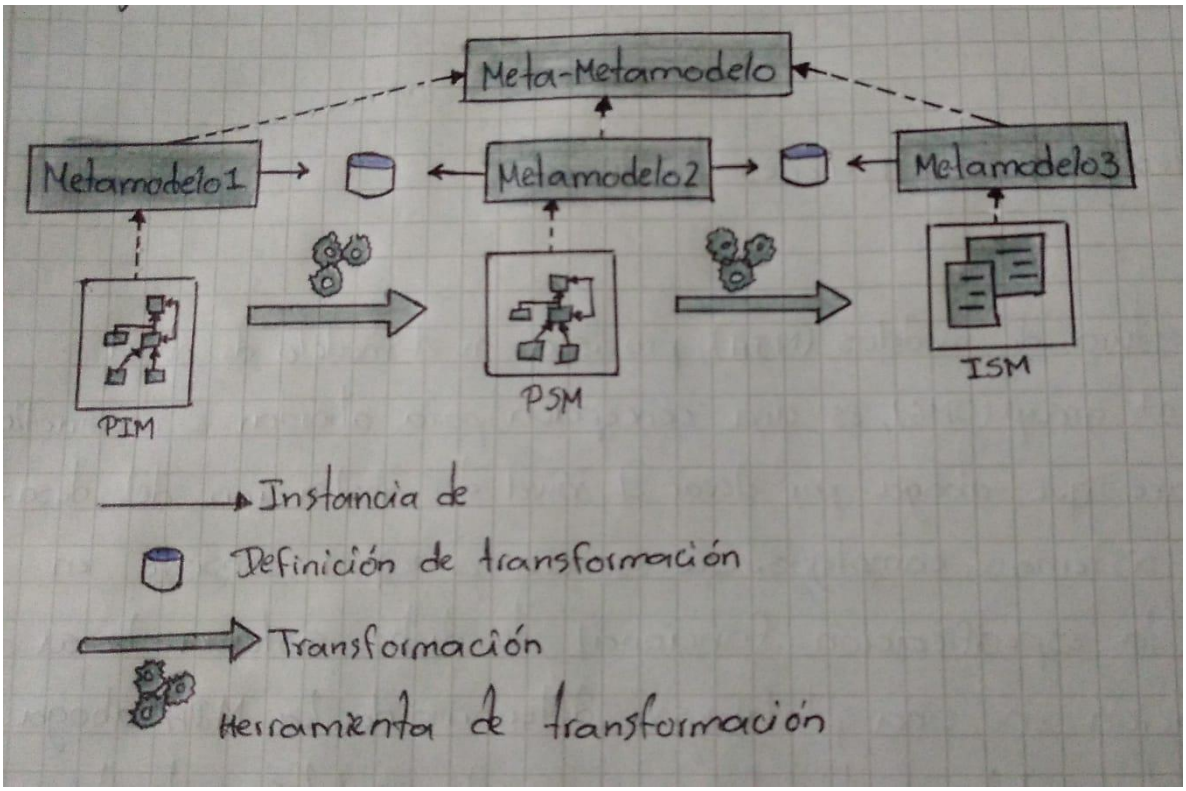


Un lenguaje de modelado para representar visualmente las decisiones de diseño arquitectónico y su rationale | Informador Técnico

Componentes MDA para patrones de diseño

La Arquitectura de Modelos (MDA), provista en el modelo del Object Management Group (OMG), es una concepción para abordar el desarrollo del software que aboga por elevar el nivel de abstracción del desarrollo de sistemas complejos. Su esencia principal consiste en desacoplar la especificación funcional de dicho sistema de su implementación en una plataforma determinada. La MDA aboga por el uso de modelos y transformaciones de modelos a lo largo del proceso de desarrollo.
Los cuatro tipos de modelos son:
Modelo Independiente de la Computación (CIM)
Modelo Independiente de la Plataforma (PIM)
Modelo Específico a la Plataforma (PSM)
Modelo Específico a la Implementación (ISM)

La propuesta de la Arquitectura Model-Driven MDA es una parte significativa de la automatización del desarrollo de software, así como de la mejora de la productividad en el desarrollo de software. Al separar la funcionalidad del sistema de su implementación tecnológica, MDA permite a los desarrolladores centrarse en el diseño de soluciones de alto nivel. La separación de preocupaciones no solo mejora la claridad, además de la mantenibilidad del sistema, sino que también facilita su adaptación a diferentes plataformas y distintas tipologías de entornos a lo largo del tiempo.



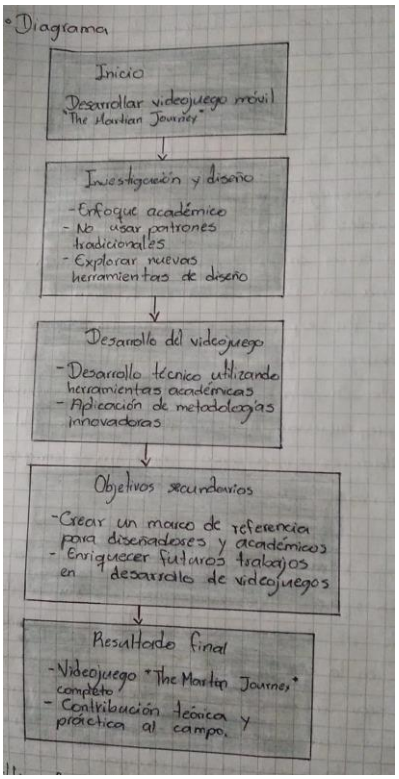
Componentes MDA para patrones de diseño

Herramientas de diseño : exploración y construcción de caminos que aporten a los procesos de desarrollo de videojuegos

El presente documento expone el proceso investigativo, el diseño y el desarrollo del videojuego móvil The Martian Journey, un proyecto innovador en el desarrollo de videojuegos. En lugar apoyarse en patrones y modelos de diseño tradicionales del sector doméstico, el proyecto se sustentó en la teoría así como en la experiencia académica acumulada en la Maestría en Diseño de la Universidad de los Andes. El principal objetivo fue experimentar nuevas formas de apoyar los marcos de trabajo y los procesos existentes en el sector del entretenimiento en relación con herramientas diseñadas. Adicionalmente el proyecto se orientó a proporcionar un marco de referencia útil tanto para diseñadores y desarrolladores de videojuegos como para académicos e interesados con el fin de enriquecer sus futuros trabajos y proyectos.

La inducción de The Martian Journey es interesante por la forma en que concibe a los videojuegos como algo más que una forma de entretenimiento. Aparte de ello, el estudio de los videojuegos puede representar una forma en la que las herramientas

de diseño y las teorías académicas sean fundamentales a la hora de crear experiencias interactivas o de tipo videojuegos, de algún modo, además, el hecho de que este proyecto haya escapar a los estándares del diseño de videojuegos en la industria, decantándose por un diseño de videojuegos fundamentado en la teoría o experiencia académica, de alguna manera muestra una tentativa de introducir tendencias actuales en el proceso de diseño de videojuegos.

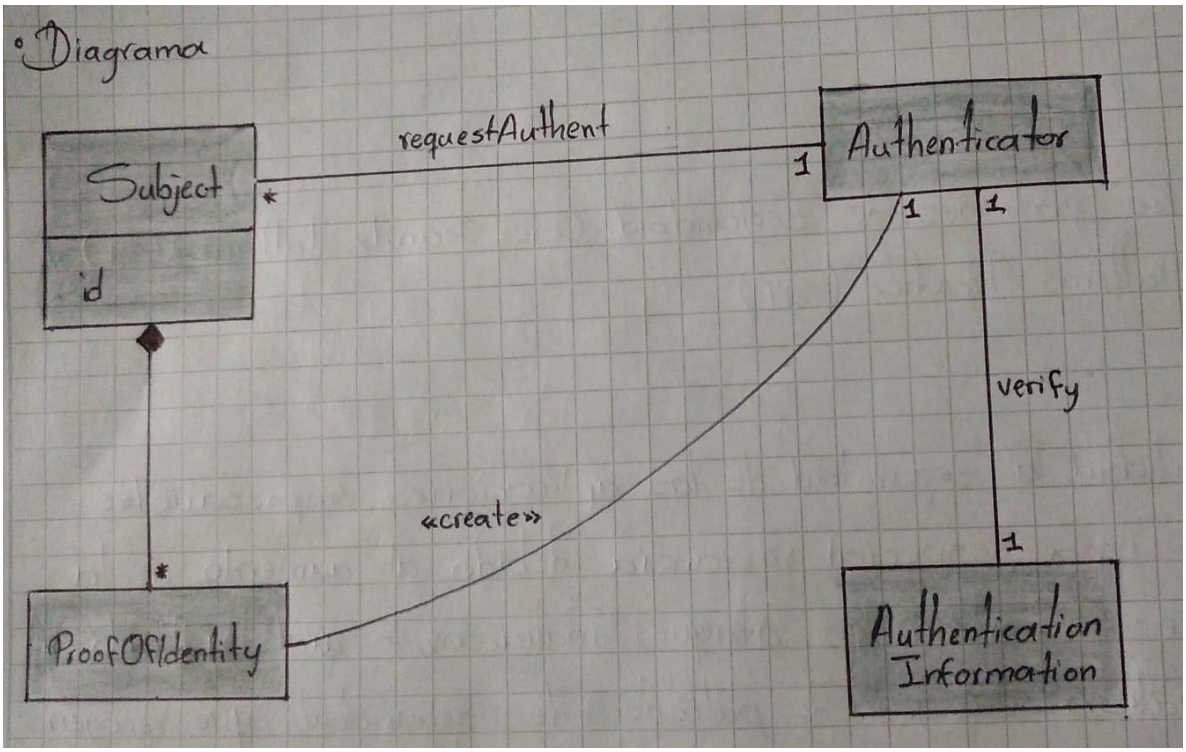


Herramientas de diseño : exploración y construcción de caminos que aporten a los procesos de desarrollo de videojuegos

Relación entre patrones de seguridad Core Security Patterns (CSP) y Security Patterns Practice (SPP)

En la actualidad, la seguridad de las aplicaciones empresariales ha cobrado una especial relevancia debido al aumento de la amenaza que comportan los ataques maliciosos, lo cual ha llevado a que se utilicen catálogos de patrones de seguridad, que recogen buenas prácticas e implementaciones de la seguridad para aquellos sistemas a proteger. En este trabajo de investigación se realiza un análisis de dos de estos catálogos de patrones de seguridad: Core Security Patterns (CSP) y Security Patterns in Practice (SPP) que abordan una serie de patrones de seguridad para los desarrolladores de los sistemas que incrementan la seguridad de las aplicaciones empresariales.

La seguridad en el desarrollo del software no es un enfoque trivial ni tampoco es una labor trivial. Las amenazas son cambiantes y obliga a los desarrolladores utilizar patrones de seguridad sobre los que pueden adelantarse y a partir de los cuales pueden realizar aplicaciones más extensibles.
Los catálogos sobre patrones de seguridad como CSP y SPP pasan a ser recursos imprescindibles de los equipos de desarrollo ya que aportan soluciones.



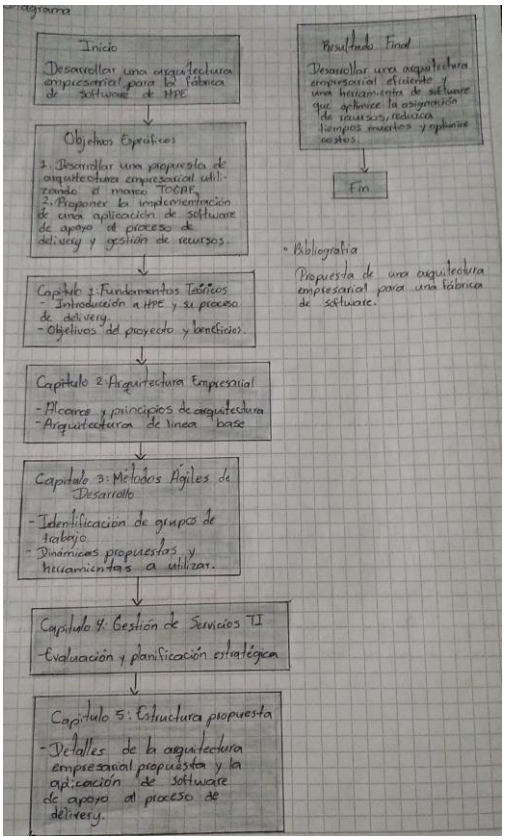
Relación entre patrones de seguridad Core Security Patterns (CSP) y Security Patterns Practice (SPP)

Propuesta de una arquitectura empresarial para una fábrica de software

El presente proyecto tiene como fin el diseño de una arquitectura empresarial para la fábrica de software de Hewlett Packard Enterprise (HPE) con el firme propósito de resolver el principal problema que en ella se presenta en lo que se refiere a los procesos de entrega de servicios, que es la mala asignación de recursos humanos a los proyectos de software que se ofrecen a los clientes, problema que incide en el proceso de delivery y por lo tanto en la eficiencia y calidad del servicio. Para ello, el

objetivo principal de este proyecto es poder llevar a cabo el diseño de una arquitectura empresarial que garantice la optimización.

El enfoque propuesto a lo largo del presente proyecto tiene una fuerte implicancia sobre la gestión de recursos humanos en una fábrica de software, algo esencial en la industria del suministro de software, donde el capital humano se considera, en muchas ocasiones, uno de los recursos más importantes de la misma. Una mala asignación de recursos puede ya ser en sí misma un motivo de retraso, sobrecostes o baja calidad del producto final, lo que necesariamente repercute sobre el cliente final y la propia empresa, ya que afecta tanto a su imagen como a su cuenta de resultados. Siguiendo el marco de trabajo TOGAF, que es conocido por su enfoque estructurado y exhaustivo en la creación de arquitecturas empresariales.



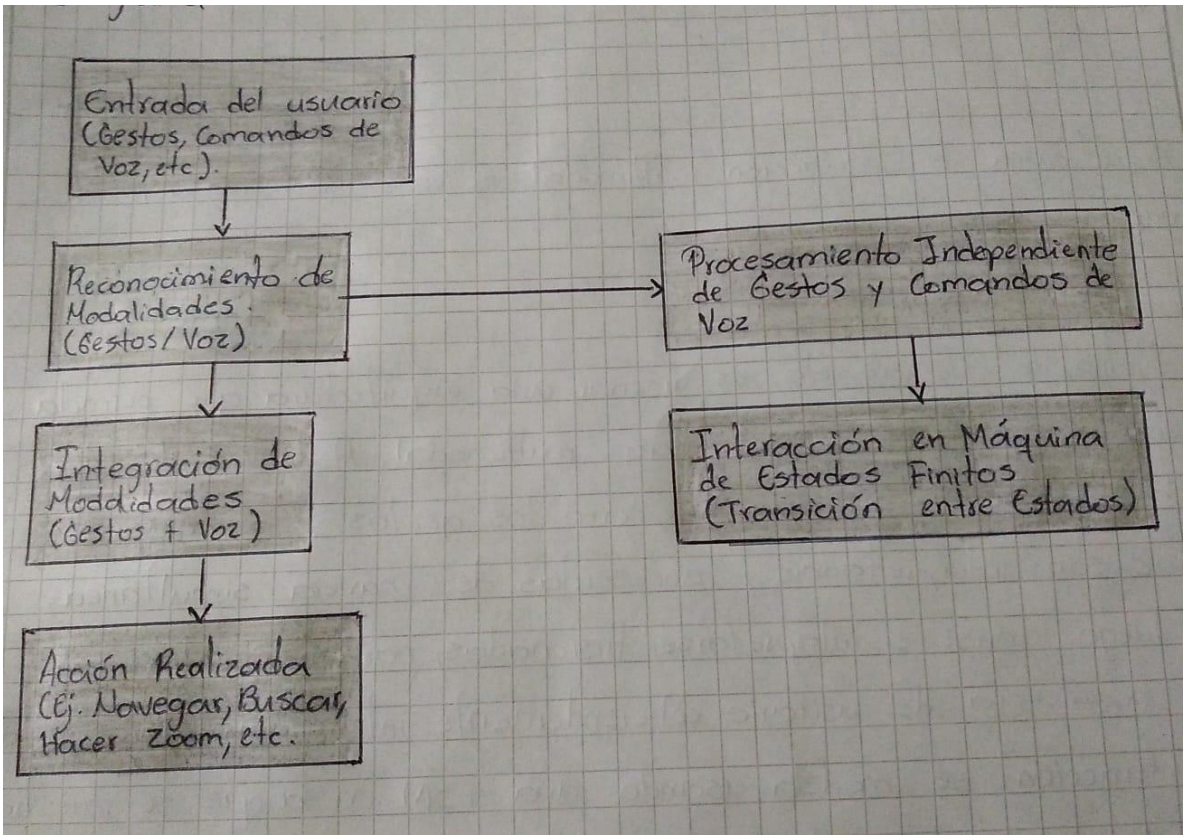
Propuesta de una arquitectura empresarial para una fábrica de software

Arquitectura de interacción multimodal aplicado a navegación en mapas

El objetivo de este proyecto es diseñar una arquitectura que permita desarrollar interfaces de interacción multimodal. A través de ella, modalidades de diálogo distintas, como los gestos y los comandos de voz, son interpretadas y procesadas de manera simultánea, para luego unirse e interpretarse, fusionadas, para presentar una sola a experiencia de usuario coherentemente integrada. Esta modalidad de interacción se modela usando una FSM en la que se muestran todos los movimientos de los usuarios son vistas como transiciones de estado. Este enfoque facilita la organización y agrupación de movimientos multimodales y los prepara para su manejo.

La arquitectura multimodal para el desarrollo de interfaces de usuario es un paso significativo hacia la creación de experiencias más accesibles. Las interfaces multimodales no solo mejoran la interacción con dispositivos, sino que también permiten que los usuarios elijan el modo de interacción más fácil para ellos, como gestos, voz o ambos.

El uso de máquinas de estados finitos (FSM) para modelar la interacción del usuario es una estrategia eficaz, porque permite organizar las transiciones de las diferentes acciones en un flujo controlado. Este enfoque facilita el manejo de interacciones complejas.



Arquitectura de interacción multimodal aplicado a navegación en mapas

Desarrollo de un sistema web para la gestión de historias clínicas y asistencia telemática mediante un chatbot utilizando la metodología Lean Thinking Developer en la clínica veterinaria “Mundo Mascotas”

Este proyecto busca implementar un sistema web para gestionar historias clínicas y ofrecer asistencia telemática en la clínica veterinaria "Mundo Mascotas". Para ello, se empleó la metodología Lean Thinking Developer, que se centra en mejorar continuamente y entregar valor rápidamente(similar a la metodologia scrum). Esta metodología se dividió en cuatro fases clave:

Investigación

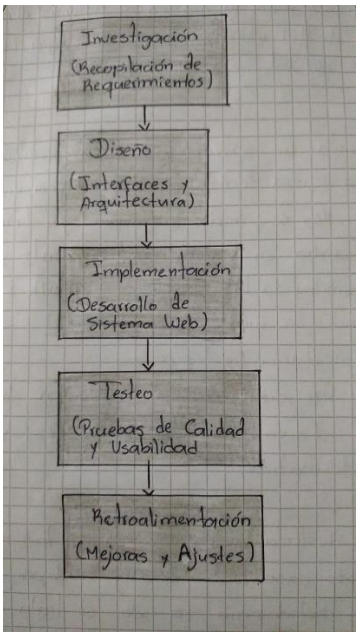
Diseño

Implementación

Testeo

El sistema web MediVen fue diseñado para mejorar la gestión de las historias clínicas de los pacientes y facilitar la asistencia telemática.

Finalmente, el desarrollo de MediVen destaca la importancia de las metodologías ágiles, como en este caso Lean Thinking Developer, en el campo del software. Esta metodología no solo se trata de cómo crear un producto funcional rápidamente, sino de mejorar constantemente con la ayuda de la retroalimentación. La fase de prueba, durante la cual los usuarios finales validan el sistema creado, es opcional pero crucial en proyectos ágiles. Demuestra que el producto final será el uno que realmente satisfaga sus necesidades nativas y que también el productor trabajará de manera intuitiva y estable.



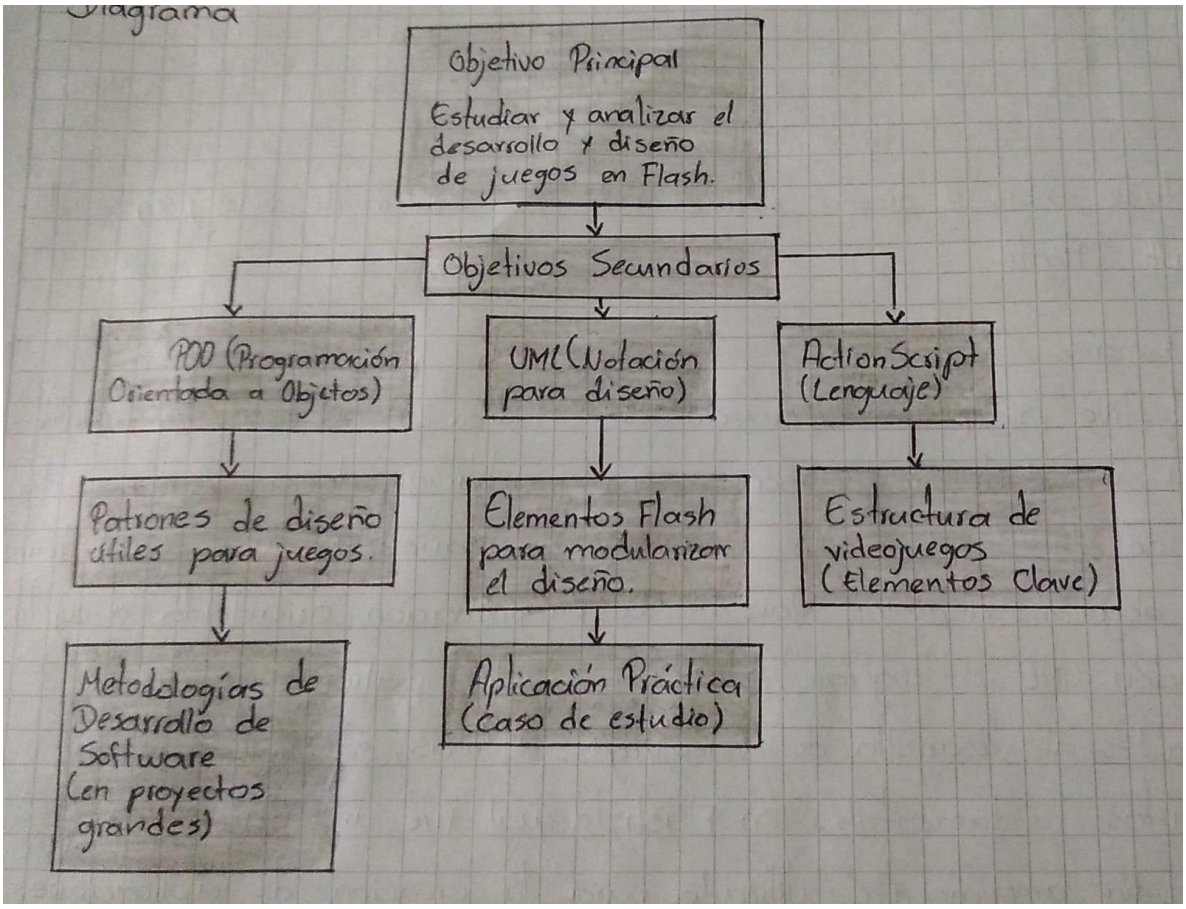
Desarrollo de un sistema web para la gestión de historias clínicas y asistencia telemática mediante un chatbot utilizando la metodología Lean Thinking Developer en la clínica veterinaria “Mundo Mascotas” del Cantón Guaranda

Estudio sobre el diseño estructurado y modular de video juegos utilizando Flash

El objetivo principal del proyecto es estudiar y analizar la metodología en el desarrollo, diseño e implementación de videojuegos utilizando Flash. Para ello, se han definido varios objetivos secundarios que abarcan aspectos clave de la programación orientada a objetos, notación UML, el lenguaje ActionScript y los patrones de diseño aplicables al desarrollo de videojuegos. Además, se identifican los elementos fundamentales en Flash; la cual fue una plataforma multimedia ampliamente utilizada para la creación de aplicaciones interactivas, animaciones,

videojuegos y contenido web, por ende facilitan la estructuración y modularización del diseño de los videojuegos.

este proyecto mostró la importancia de comprender y aplicar una combinación de conceptos de programación, herramientas de diseño y metodologías de desarrollo. Cualquier proyecto de software, incluidos los videojuegos, requiere POO para organizar el código de una manera escalable y reutilizable. Del mismo modo, UML desempeña un papel fundamental al brindarles a los desarrolladores un lenguaje estandarizado de diseño con el que planificar y permitir de la mejor manera posible que las ideas se expresen en estructuras funcionales antes de la implementación.



Estudio sobre el diseño estructurado y modular de video juegos utilizando Flash

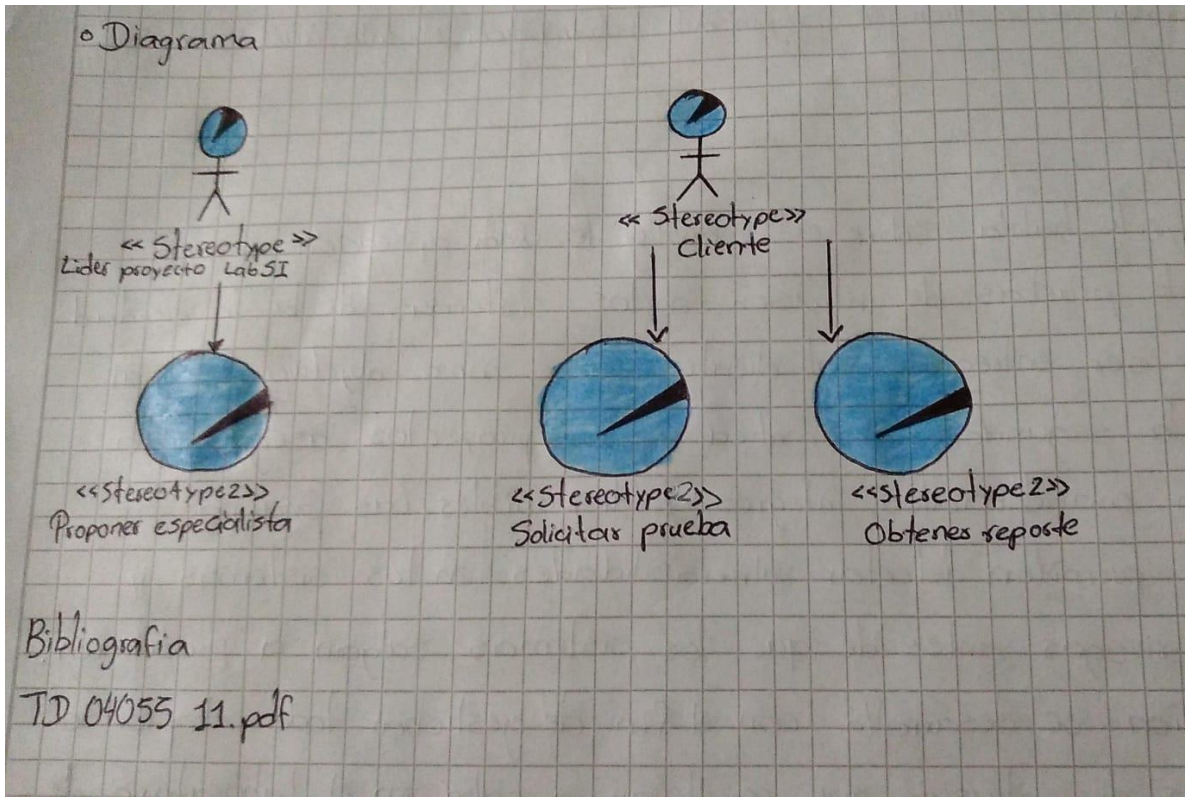
Plataforma de Gestión de Seguridad Informática. Módulo Pruebas de Intrusión

Este proyecto habla sobre el desarrollo de una aplicación para la gestión de pruebas de intrusión en los sistemas de la Universidad antes de ser entregados al cliente y tiene como objetivo principal mejorar la seguridad informática de los proyectos, asegurando que sean confidenciales, íntegros y disponibles. Las pruebas de intrusión permiten detectar vulnerabilidades en los sistemas y reducir riesgos antes de que los sistemas salgan a producción.

La aplicación se desarrolló con el fin de gestionar todo el proceso de pruebas de intrusión que se llevan a cabo en el Laboratorio de Seguridad Informática (LabSI).

La seguridad informática es un área fundamental para proteger la confidencialidad, la integridad y la disponibilidad de la información, especialmente en sistemas que manejan datos delicados. Las pruebas de intrusión son una herramienta clave para detectar vulnerabilidades antes de que los sistemas se encuentren en producción. Este tipo de pruebas simulan posibles ataques y ayudan a identificar puntos débiles que podrían ser explotados por atacantes maliciosos.

El desarrollo de una aplicación para gestionar las pruebas de intrusión mejora significativamente la eficiencia.



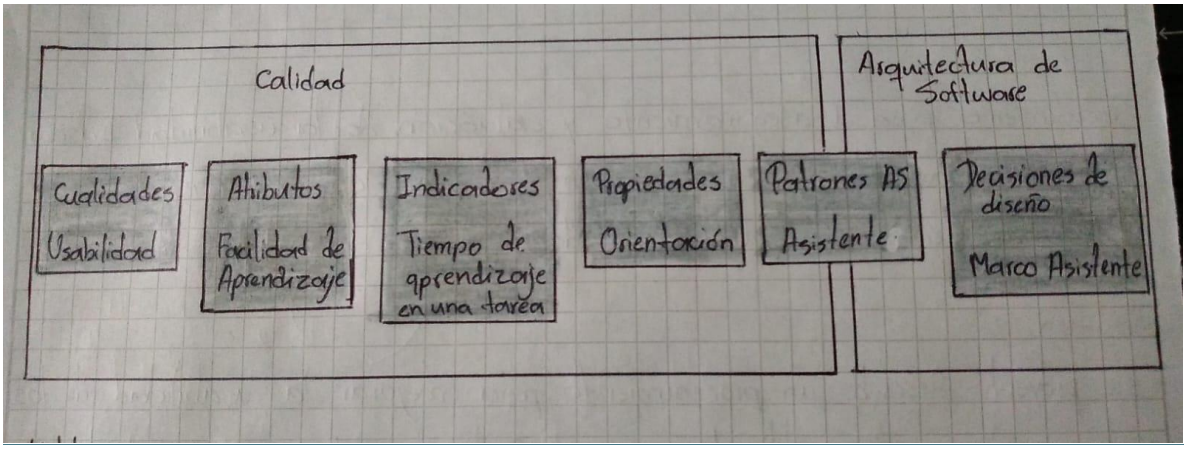
TD_04055_11.pdf

Procedimiento para el aseguramiento y evaluación de la usabilidad basado en patrones arquitectónicamente sensibles para los sistemas de gestión del Centro de Informatización de la Gestión de Entidades.

Este proyecto describe un procedimiento para mejorar la usabilidad de los sistemas de gestión utilizados por el Centro de Informatización de la Gestión de Entidades (CEIGE). La metodología está basada en tres fases, tomadas de la Ingeniería de Usabilidad, que se alinean con el modelo de desarrollo del CEIGE.

Tiene como propósito identificar y dar solución a problemas de usabilidad en las primeras etapas del desarrollo de los sistemas, cuando es más fácil y barato corregirlos. Este procedimiento propone detectar los errores desde el principio, lo que permite hacer mejoras antes de que el sistema salga a producción.

La usabilidad es uno de los aspectos más importantes de cualquier sistema informático, especialmente cuando se trata de plataformas usadas por muchos usuarios, como en el caso de los sistemas de gestión. La facilidad con la que los usuarios interactúan con el sistema no solo afecta la experiencia del usuario, sino también la eficiencia y productividad del trabajo diario. Por ello, detectar problemas de usabilidad en las primeras fases del desarrollo es clave para evitar que se conviertan en grandes obstáculos una vez que el sistema esté en producción.



Repositorio Digital:Procedimiento para el aseguramiento y evaluación de la usabilidad basado en patrones arquitectónicamente sensibles para los sistemas de gestión del Centro de Informatización de la Gestión de Entidades.

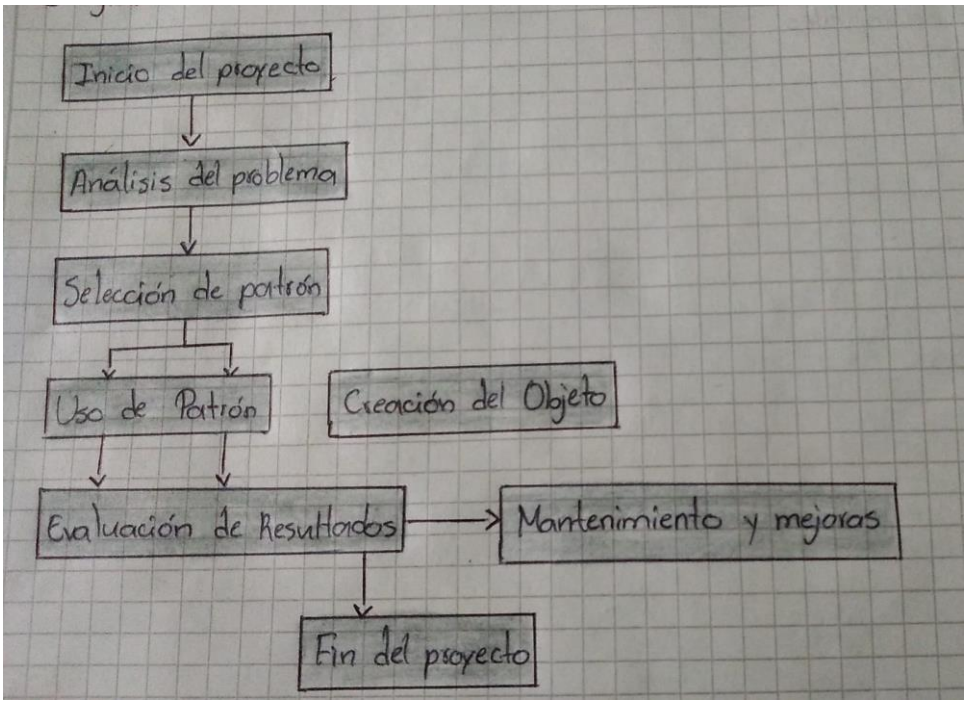
Construyendo software innovador implementando patrones de diseño creacionales

La programación orientada a objetos (POO) es un enfoque ampliamente utilizado en el desarrollo de software que organiza el código en objetos, los cuales modelan entidades del mundo real. Esta técnica permite crear programas más fáciles de mantener y expandir a medida que el software crece.

Con el tiempo, los programadores enfrentaron problemas comunes durante el desarrollo de aplicaciones, por lo que se crearon patrones de diseño, que son soluciones reutilizables para problemas recurrentes en la programación. Los patrones de diseño, se empezaron a documentar en los años 90 y se han vuelto esenciales en la programación moderna.

La programación orientada a objetos es una de las mejores prácticas que ha permitido transformar el desarrollo de software en algo más estructurado y organizado. Gracias a este enfoque, los programadores pueden representar conceptos del mundo real de una manera mucho más natural y comprensible, lo que facilita tanto la creación como el mantenimiento de las aplicaciones.

Por otro lado, los patrones de diseño son como "recetas probadas" que nos permiten resolver problemas comunes de manera eficiente, evitando reinventarlo.

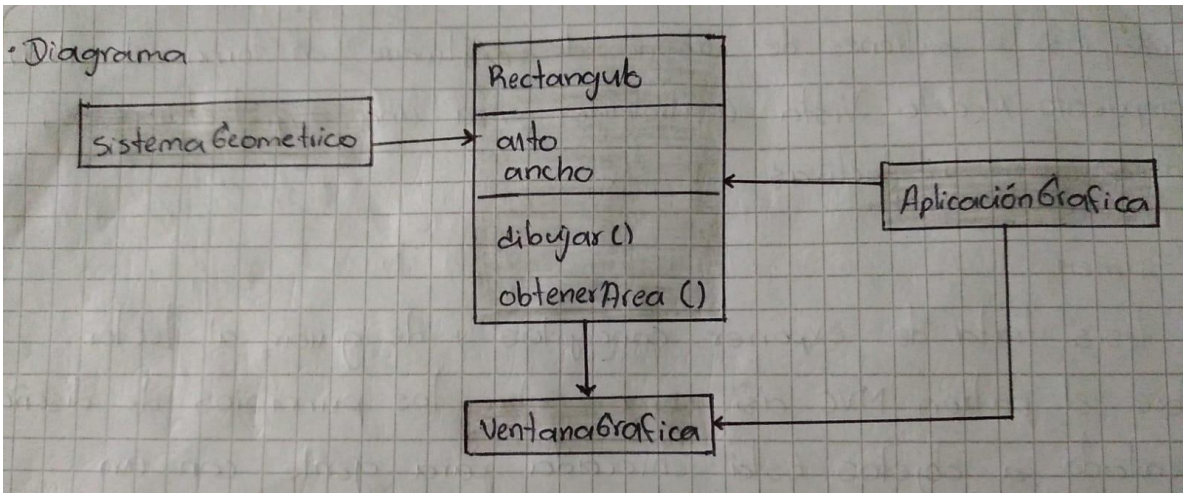


2023-12-06_construyendo_software_innovador.pdf

Principios y patrones de diseño de software en torno al patrón compuesto modelo vista controlador para una arquitectura de aplicaciones interactivas.

La tesis trata de exponer conceptos y de guiar al lector sobre el patrón MVC así como sobre los principios de diseño orientado a objetos. Está enfocada para gente con un conocimiento básico de desarrollo orientado a objetos y conocimiento UML. La tesis quiere hacer un hincapié claro de la importancia de una comunicación clara así como la de manejar la terminología técnica en el desarrollo de software que debe ser para conseguir un diseño flexible, mantenible y reutilizable. La tesis no es generadora de nuevos principios o patrones pero se espera que sirva la forma de enseñar.

La presente tesis es un exponente de una metodología pedagógica en el desarrollo de software al priorizar la claridad en la exposición de conceptos así como en la comunicación técnica, no para proponer innovaciones, sino para consolidar el conocimiento existente sobre la técnica MVC y principios de diseño orientado a objetos, destacando, además, la importancia de una comunicación clara y precisa, así como de un vocabulario técnico común, subrayando un aspecto clave en la colaboración y el aprendizaje en este campo: la comprensión mutua. Esta tesis no solo trata de entender, sino también de enseñar y comunicar.



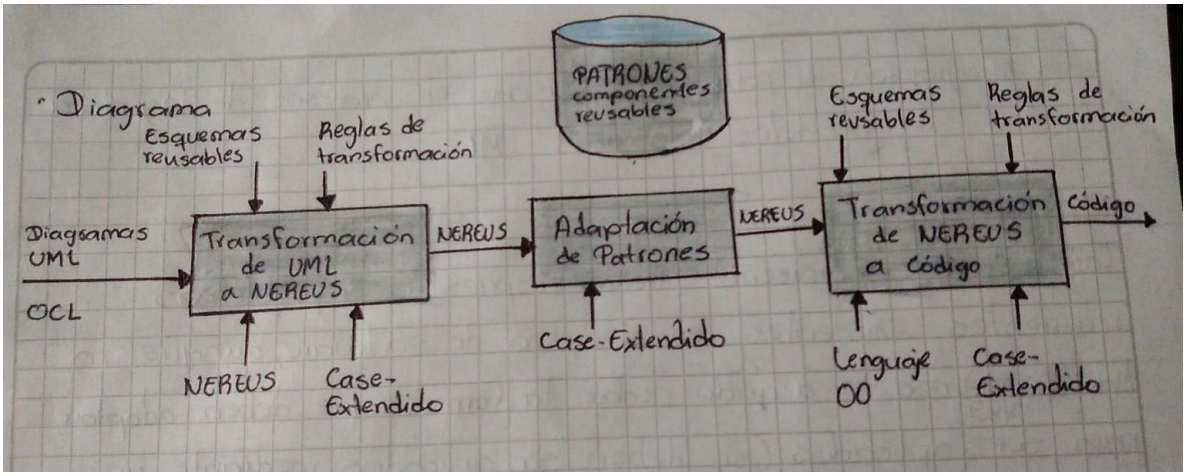
https://acortar.link/nrL2n7

Una creación de patrones de diseño en procesos de ingeniería forward de modelos estáticos UML

los patrones de diseño ofrecen respuestas a problemas recurrentes existentes en el diseño de software, aunque no hay consenso ni acuerdo sobre la forma que deben adoptar para ser aplicados(si deben ser aplicados mediante un determinado lenguaje, plataforma o herramienta, etc), sí hay acuerdo en que su aplicación debería ser automática o en las óptima atención incorporando técnicas para obtener un buen

rendimiento, porque su aplicación manual conlleva mucho tedio y una cierta propensión a errores. La industria ha aceptado la creencia de que los patrones de diseño automatizados o asistidos mejoran la comunicación y permiten un desarrollo más rápido.

Los patrones de diseño son cruciales en el desarrollo de forma material, pues ofrecen soluciones consolidadas para problemas recurrentes. Pero bien es cierto que, aunque su valor se ha sabido, los desacuerdos sobre su práctica, es decir, hoy si debería ser gracias a herramientas, lenguajes concretos o plataformas, son un inconveniente. Lo que está claro es que su práctica manual es lenta y propensa a errores. Por lo tanto, automatizar o asistir su uso es, por regla, hoy 1 de los elementos claves para explotar sus ventajas, mejor comunicación entre los desarrolladores y tiempos de desarrollo más cortos.



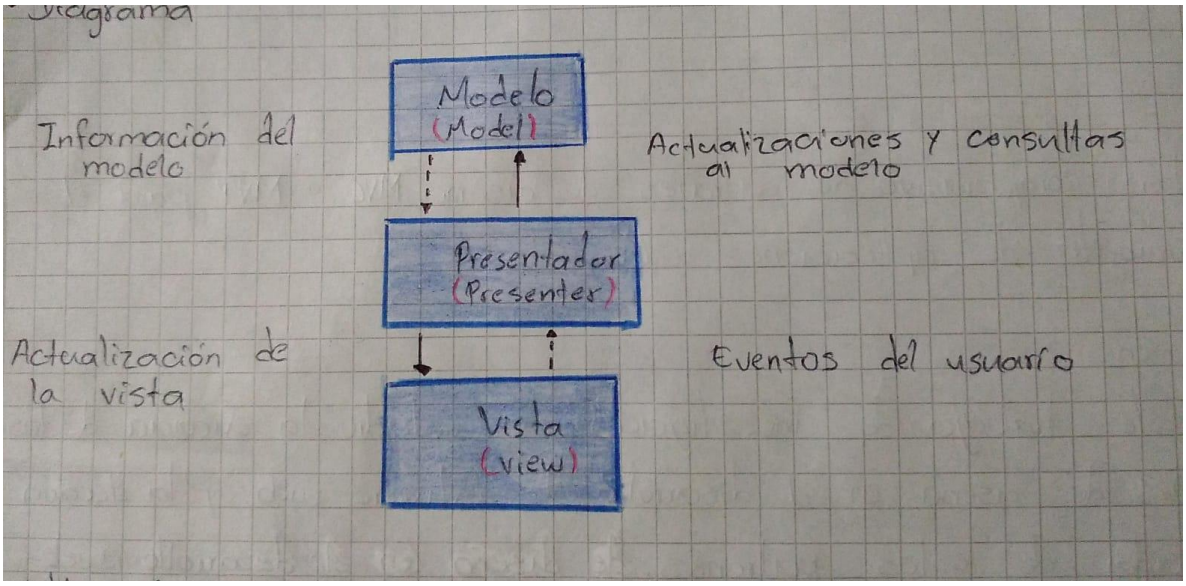
https://sedici.unlp.edu.ar/handle/10915/21285

Análisis comparativo de patrones de diseño MVC y MVP para el rendimiento aplicaciones web.

En este trabajo de investigación, se estudió la eficacia de los patrones de diseño en el desarrollo de software web y la elección adecuada a estos patrones. Para ello se encontraron varios patrones, se evaluaron los resultados obtenidos y se seleccionaron los dos siguientes: MVC y MVP. Cada 1 de ellos se aplicó a los proyectos de software web y se evaluó utilizando los siguientes criterios: Tiempo de desarrollo, líneas de código, uso de la memoria RAM, uso de la CPU y tiempo de respuesta.

Nuestros resultados muestran que el patrón MVC es significativamente más eficaz que el patrón MVP.

El trabajo explica la importancia de realizar una correcta elección de los patrones de diseño del software para aplicaciones web. La comparación que se establece entre MVC y MVP se explica en el sentido de que ambos patrones de diseño persiguen de alguna manera la mejora y la organización del código, aunque el grado de pertinencia de uno y otro patrón dependerá del contexto de aplicación y de los objetivos de cada proyecto de software. Lee evidente superioridad de MVC que ha logrado cosechar en esta investigación sugiere, de cierta manera, que no todos los patrones son apropiados para todos los proyectos y que, por lo tanto, es importante estudiar y analizar sus beneficios y desventajas.

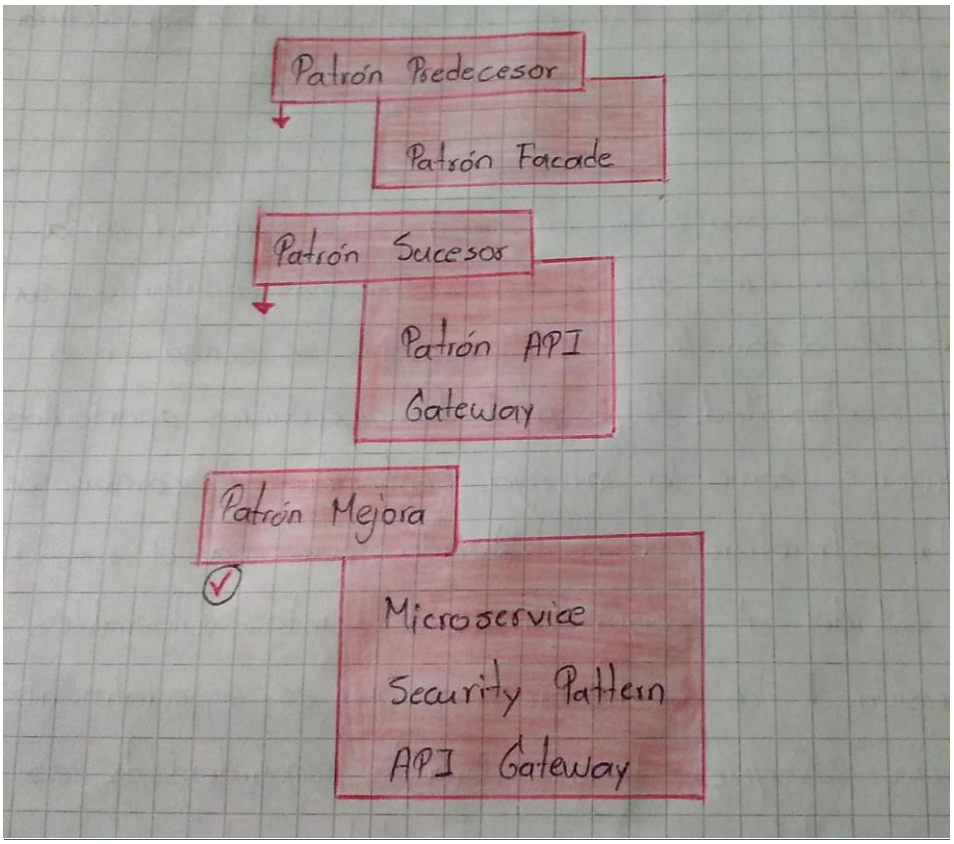


https://repositorio.uss.edu.pe/handle/20.500.12802/11182

Un patrón de software en seguridad a la arquitectura de un microservicio

En este artículo se han analizado los riesgos de seguridad en la arquitectura de microservicios, que se ha vuelto más popular por su concepto moderno el desarrollo de software, que se centra en el la modularidad y la escalabilidad. A pesar de los beneficios aportados por los microservicios como la entrega continua y el cambio en las tecnologías, la estructura descentralizada aumenta el riesgo de ser atacado, debido al mayor número de puntos vulnerables. La principal conclusión de la investigación es que es especialmente crítico optar un enfoque integral para la seguridad de los microservicios y tomar medidas que protejan los datos y el sistema.

Esta se hace pensar en el equilibrio que es necesario mantener entre innovación y seguridad llevando el desarrollo del software moderno. Si bien la creación de las aplicaciones de esta forma se ha beneficiado de la adopción de microservicios ya que mejoran la escalabilidad, flexibilidad y otros aspectos. Al mismo tiempo, hoy es un recordatorio de que traslapar el sistema en piezas más pequeñas y separadas conlleva un riesgo de seguridad diferente o generalmente mayor. La idea clave es que no debemos quedarnos atrás en términos de avances tecnológicos, si no de cuidar la seguridad. Además, deben avanzar juntos.

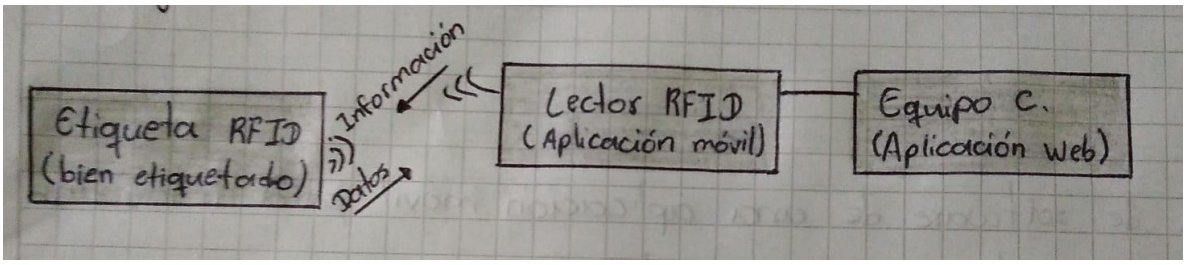


http://51.143.95.221/handle/TecNM/6574

Directorio de software de una aplicación móvil para desarrollar un sistema de identificación por radiofrecuencia.

Un sistema RFID, el cual permite la identificación mediante etiquetas, ayuda a realizar un control de inventario actualizando la información sobre la cantidad y localización de bienes. Esta tesis estableció una arquitectura de software para desarrollar una aplicación móvil que forme parte de un sistema RFID, el cual respalda el inventario el Instituto tecnológico de Orizaba y también se presentan las aplicaciones generadas a partir de la arquitectura de software propuesta. Esta solución busca optimizar la gestión de recursos y reducir errores en el seguimiento de activos, facilitando una administración más eficiente el inventario.

La integración de datos RFID como tecnología en la gestión del inventario está en contradicción con la forma en que la tecnología puede en último extremo cambiar procesos corrientes, un método que puede facilitar un control más efectivo y preciso de los recursos. Esta mejora no se reduce a una utilización óptima de los bienes, también puede disminuir las dificultades humanas y los tiempos de las operaciones, en la medida que las instituciones educativas y empresariales buscan la máxima eficacia. En otras palabras la tecnología debe adaptarse a las necesidades específicas para impulsar el cambio.



https://www.redalyc.org/pdf/5122/512251501004.pdf

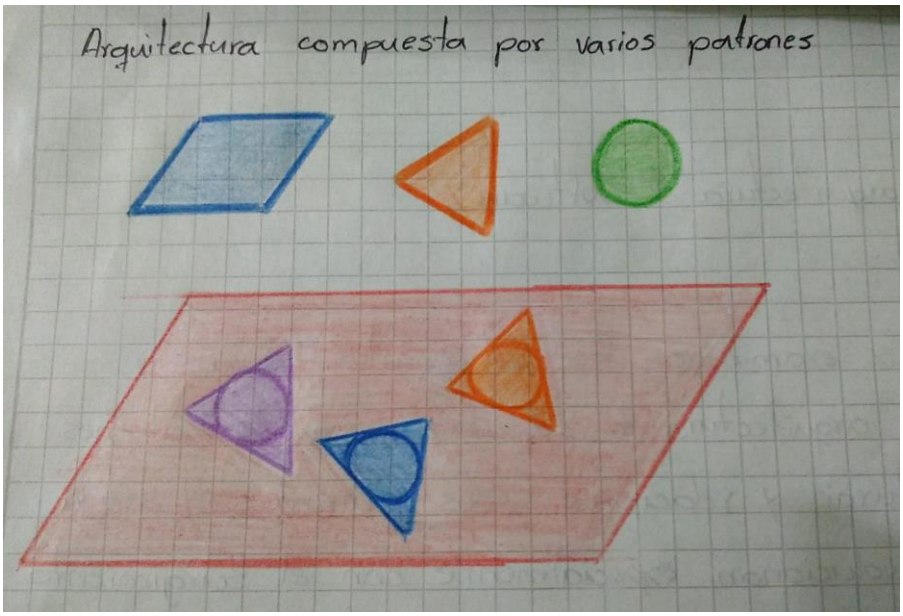
Especificación de la arquitectura de software

En la presente lectura, se examinarán los componentes básicos y fundamentales de una arquitectura de software y cómo los patrones arquitecturales estructuran y relacionar esos elementos. A medida que las tecnologías evolucionan, especialmente con el surgimiento de las plataformas web y móviles, las necesidades de infraestructura de diseño cambian; cada vez se impone mayores soluciones

mucho más rápidas en términos de tiempos de entrega y procesamiento de datos y más seguras en términos de transacciones. El artículo trata sobre los elementos importantes de una arquitectura de software y su contexto interpretativo desde la perspectiva de la actualidad.

Esta lectura no se enfatiza lo importante que es tener una buena base en el desarrollo de software, más ahora que las aplicaciones web y móviles están creciendo tanto. A medida que la tecnología avanza, hoy el software debe ser cada vez más rápido, seguro y capaz de manejar grandes cantidades de información sin perder eficiencia.

Y los patrones de diseño son como una guía que ayuda a organizar todos los componentes del sistema de forma clara y coherente, permitiendo que el software funcione bien, se adapte y mejore con el tiempo.

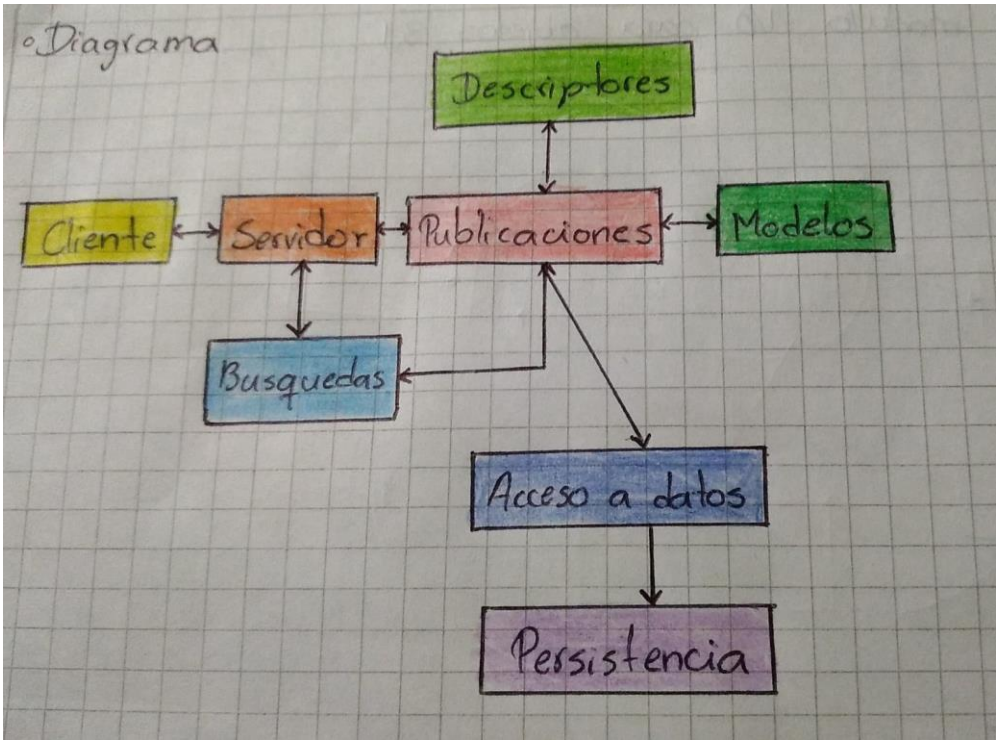


https://revistas.udistrital.edu.co/index.php/tia/article/view/18076

Arquitectura de un módulo 1/0 para objetos 3D

La presente investigación tiene por objeto el problema de la persistencia de los objetos 3D, el cual es una gestión de grandes volúmenes de datos en memoria y disco, así como de procedimientos en la comparación de objetos 3D, los cuales están basados en el teorema matemático que le sirve de base, un teorema que supone un alto coste computacional. Para ello era necesario realizar un exhaustivo análisis de requisitos, de diseño arquitectónico y de investigación de la gráfica por ordenador, y de arquitecturas de software. La solución que se propone responde a un enfoque integral, el cual intenta combinar teoría y práctica para garantizar su efectividad. El producto resultante supone una arquitectura independiente que asegura la funcionalidad requerida, mientras que el prototipo se ha desarrollado dentro de la arquitectura VITRAL, del grupo TAKINA, como módulo de la capa de servicios.

El trabajo destaca por el hecho de que se vive junto a retos técnicos importantes en el manejo de los objetos 3D. Lo anterior confirma la importancia del diseño arquitectónico cuidado, de una planificación metódica para abordar la problemática, y la incorporación del prototipo a una arquitectura cooperativa como VITRAL pone de manifiesto la importancia del trabajo modular y de la investigación aplicada, es una interesante y real mejora en la construcción de herramientas para el manejo gráfico, y también es un reconocimiento de que hay que tener presentes las soluciones escalables que podemos esperar para responder, por ejemplo, a problemas de almacenamiento y comparación de sistemas complejos.

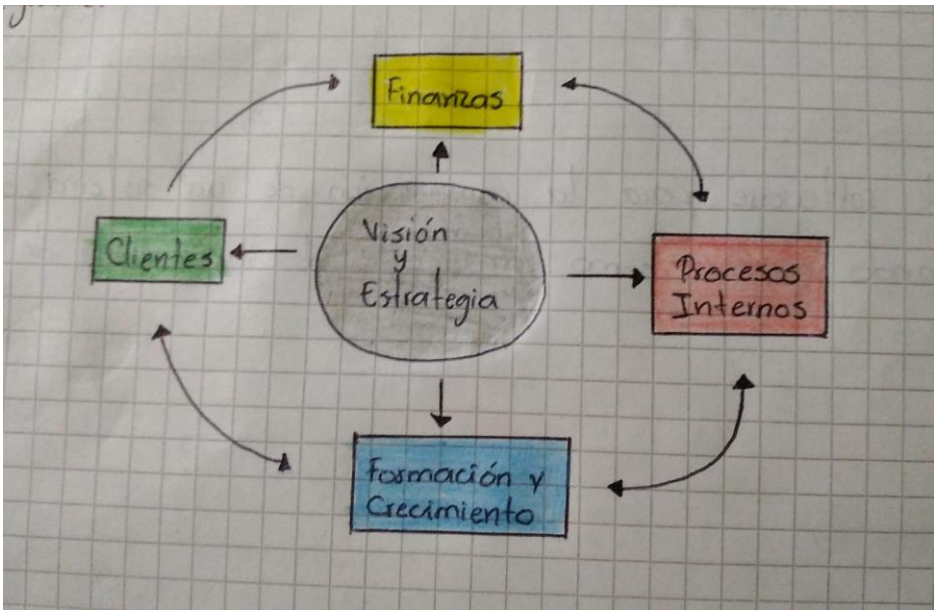


https://core.ac.uk/download/pdf/71418995.pdf

Arquitectura de software para la construcción de un sistema de cuadro de mando integral como herramienta de inteligencia de negocios.

Cuando las organizaciones han conseguido asumir diversas herramientas y metodologías de toma de decisiones, almacenan información errónea. La Inteligencia de Negocios (IB) actúa ante esta problemática, yuxtaponiendo una serie de herramientas que extraen, depuran y analizan datos, siendo apta para organizaciones de cualquier tipo o tamaño. Dentro de estas herramientas destaca el Cuadro de Mando Integral (CMI) por permitir la posibilidad de un seguimiento constante de la estrategia y la mejora de los procedimientos de gestión decisiva; por ello es necesario construir un modelo de CMI basándose en técnicas o metodologías actuales que de forma sistemática incorporen las buenas prácticas en el desarrollo del software y, por lo tanto, los sistemas y procedimientos organizacionales.

El escrito destaca cómo las herramientas de Inteligencia de Negocios, como el CMI, pueden transformar datos en decisiones clave para la estrategia de una empresa. Al aplicar buenas prácticas de desarrollo de software, las organizaciones no solo optimizan sus procesos, sino que también ganan en precisión y flexibilidad frente a un mercado cada vez más competitivo. Es un recordatorio de que la combinación de tecnología y estrategia es fundamental para tomar decisiones más informadas y acertadas. Además, al integrar estas herramientas, las empresas logran ser más ágiles, transparentes y orientadas a resultados. En resumen, los datos se convierten en un activo invaluable para cualquier organización que quiera crecer.



(*Vista de Arquitectura de Software Para la Construcción de un Sistema de Cuadro de Mando Integral Como Herramienta de Inteligencia de Negocios*, s. f.-b)