

# Lenguajes de Patrones de arquitectura de software Una aproximación al mundo del arte

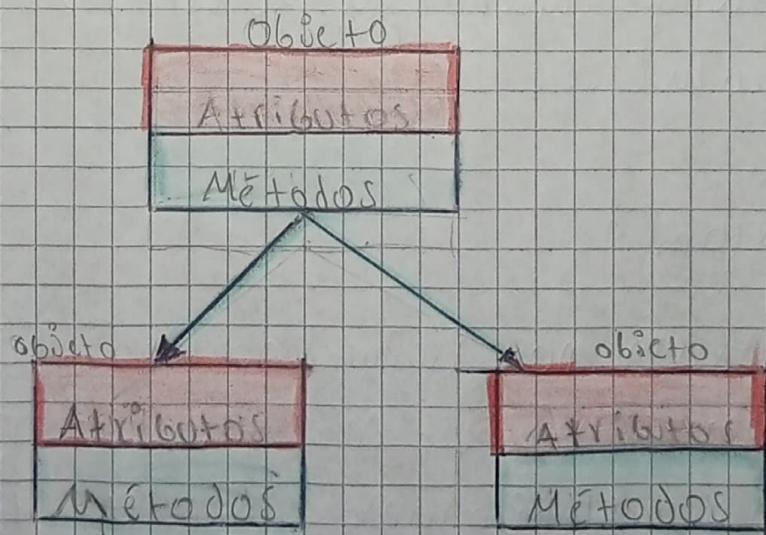
- Este artículo aborda los lenguajes de Patrones en la arquitectura de software, explorando su evolución, aplicaciones y contribuciones. Comienzan con sus raíces en los patrones de diseño introducidos por Christopher Alexander, quien desarrolló un enfoque sistemático para resolver problemas comunes mediante Patrones reutilizables. Posteriormente, se adoptó este concepto al desarrollo de software, permitiendo diseñar arquitecturas más eficientes y escalables.

## - Reflexión

- El artículo nos proporciona una visión detallada sobre los lenguajes de Patrones en la arquitectura de software, explorando su origen, evolución y aplicaciones. Los lenguajes de Patrones son herramientas poderosas para resolver problemas comunes en el diseño de sistemas, permitiendo soluciones reutilizables y eficientes.

Una idea de descomponer problemas complejos en subproblemas más manejables y luego aplicar patrones específicos para resolverlos es un enfoque práctico y adaptable a distintos contextos.

Jiménez-Torres, V. H., Jiménez-Flores, W. R., Flores-Palacios, J. I. (2014). Combinación de patrones de construcción de software: una aproximación al diseño del arte. *Science et technique*, 39(8), 229-236.



# Arquitectura de Software con Programación

## Orientada a objeto

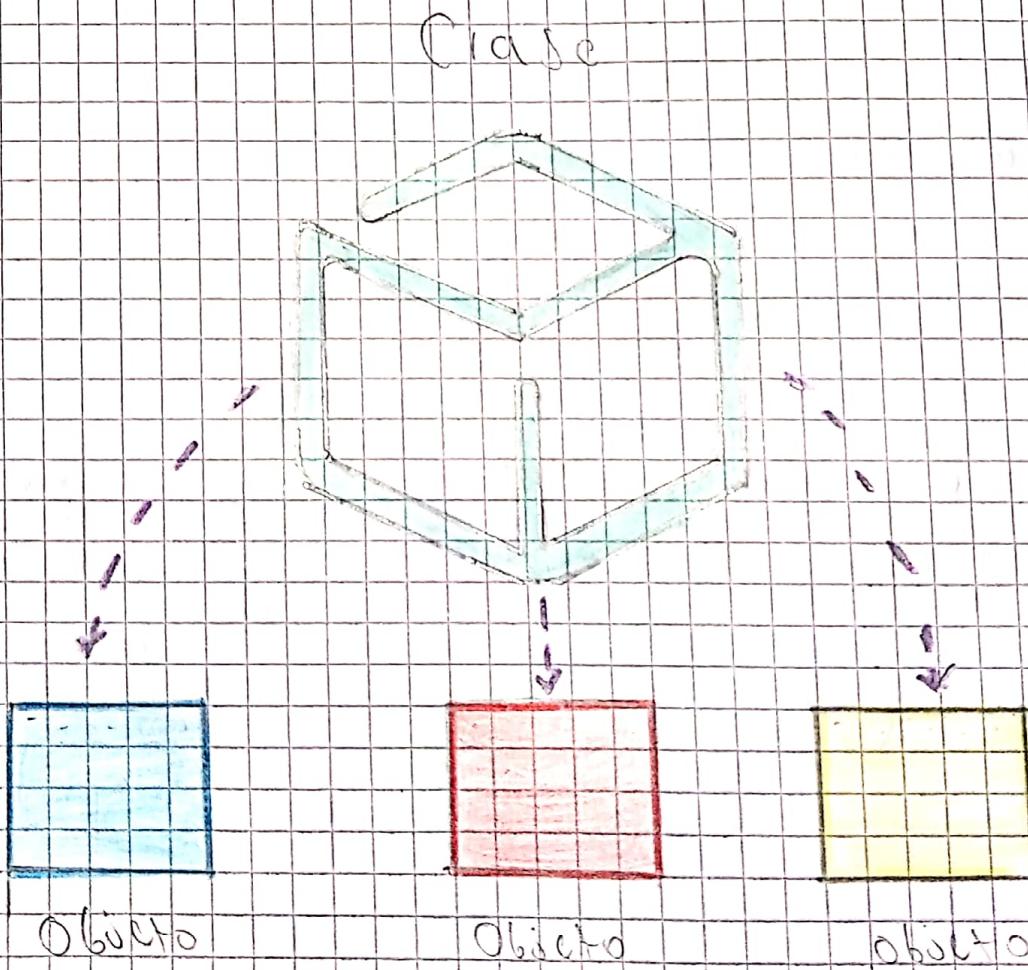
- El artículo analiza como la programación orientada a objetos (Poo) influye en la arquitectura de software. Describe a la Poo como un enfoque
- que organiza programas en términos de clase de objetos, permitiendo un diseño más estructurado.
- reutilizable y cercano al mundo real. En este
- se destaca que tiene las siguientes ventajas:
- han sido clave para implementar estos
- técnicos, facilitando la creación de sistemas
- complejos.

## Reflexión

Más bien el análisis en este se ha centrado en cómo la arquitectura de software basado en programación orientada a objetos de objetos se diferencia de otros en su diseño. Esta forma de diseño de sistemas más complejos y funcionales. El artículo explica que esta metodología significa el uso de los componentes reutilizables llamados clases y objetos, lo que permite modular

Problemas del mundo real de forma más intuitiva.

Vera, J.B. (2023). Arquitectura de Software con programación orientada a objetos. Año del Conocimiento. Revista Científico - Profesional, 8(11), 1497-1568.



## Incidencia de los Patrones de diseño de Software en la seguridad de aplicaciones

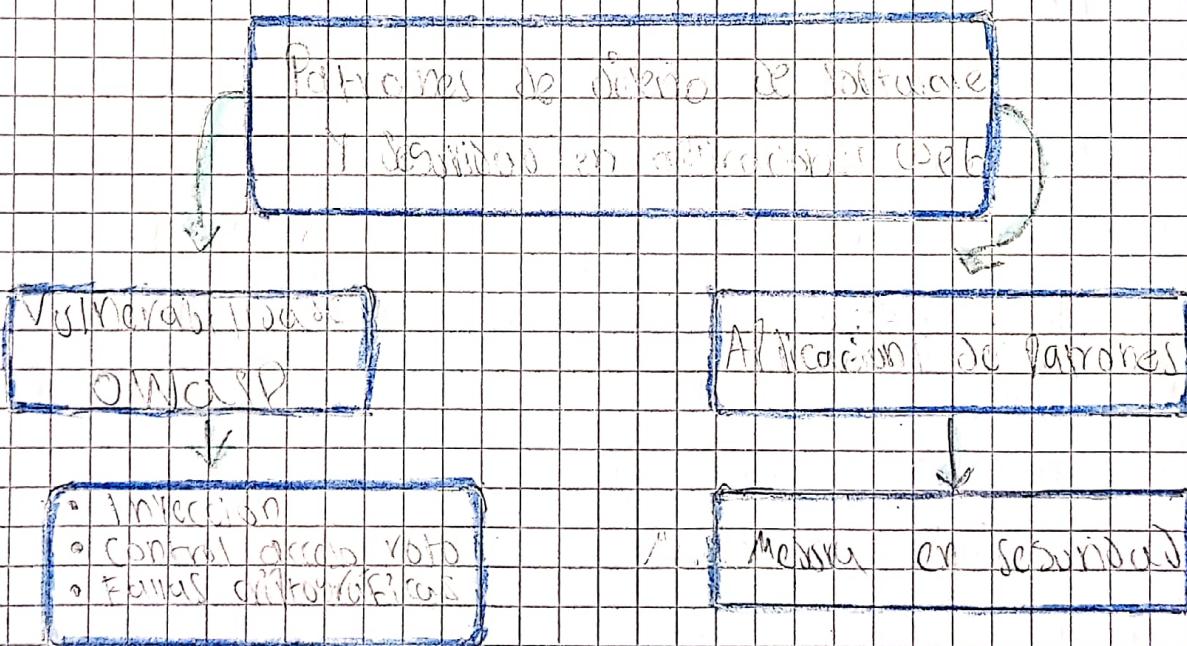
El artículo analiza como los patrones de diseño pueden mejorar la seguridad en aplicaciones web, destacando que estos ofrecen soluciones estructuradas a problemas comunes en el desarrollo. Se enfoca en tres vulnerabilidades críticas identificadas por OWASP: Control de acceso, criptografía débil y ataques de inyección como SQL o XSS. La investigación demuestra que el uso de patrones como Inversion of Responsibility, Proxy y Rate Limiting no solo mitiga estas fallas, sino que también facilita el mantenimiento y la escalabilidad del código.

### Reflexión

- Al leer el artículo, entiendo sobre cómo la arquitectura de software juega un papel crucial en la seguridad de las aplicaciones web. Se relata en el artículo que estos no solo ayudan a organizar el código, sino que también ofrecen soluciones prácticas para mitigar vulnerabilidades comunes.

Comprendí cómo cada patrón aborda problemas específicos, como el control de acceso, la inyección de código o el Cross-Site Scripting, mediante estrategias claras y reutilizables.

Mesías-Valencia, J.J., & Cevallos-Moroz, F.D. (2014). Incidencia de los patrones de diseño de software en la seguridad de aplicaciones web. 8(1), 236 - 259.



## Implementando estrategias de resiliencia en una arquitectura basada en microservicios

- El artículo analiza la implementación de estrategias de resiliencia en una arquitectura de microservicios, utilizando como caso de estudio el sistema de Pedidos Ya. Se enfoca en el microservicio Niles, encargado de proporcionar el menú de los restaurantes.
- El texto explica cómo los fallos en sistemas distribuidos pueden afectar el funcionamiento de toda la arquitectura y destaca la importancia de la resiliencia para mitigar estos problemas. Se describen tres patrones de diseño prácticos: Timeout, Retry y Circuit Breaker, explicando su funcionamiento y beneficios.

## Reflexión

- Después de leer este artículo sobre la implementación de estrategias de resiliencia en una arquitectura de microservicios, me quedaron varias reflexiones.
- Interesantes lo que más me llamó la atención es cómo estos patrones de diseño pueden marcar una gran diferencia en la estabilidad y rendimiento de

○ Sistemas distribuidos complejos. El caso de estudio de PedidosYa muestra claramente los beneficios tangibles de aplicar patrones como Timeout, Retry y CircuitBreaker.

○ Me parece fascinante cómo algo sencillamente simple como establecer tiempos de espera adecuados puede mejorar drásticamente los tiempos de respuesta.

Suárez, S., Kuczi, L., Hyun, V., & Montecinos, D. M. (2024). Implementando estrategias de resiliencia en una arquitectura MVC. In XXIX Congreso Latinoamericano de Computación (Luganíq al 12 de octubre del 2023).

Estrategias de resiliencia

en un microservicio

Patrones de diseño	Resultados	PedidosYa
Timeout	Incremento gradual	Master utilizada
Retry	Incremento gradual	Microservicio NICS
Circuit breaker		

Patrones de diseño, refactorización y antipatrones.  
Ventajas y desventajas de su utilización en el software orientado a objetos.

El artículo incluye tres conceptos importantes en la arquitectura de software:

Patrones de diseño, refactorización y antipatrones.

- Los patrones de diseño son soluciones probadas a problemas comunes de diseño, que ayudan a crear software más flexible y reutilizable. La refacto-  
rización es el proceso de mejorar el código  
existente sin cambiar su diseño. También, el mantenimiento  
es más fácil de entender y mantener.
- Los antipatrones, por otro lado, son soluciones comunes  
para problemas que también existen.

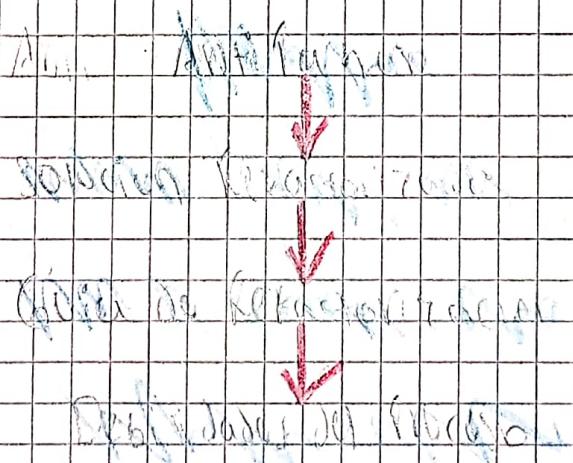
### Reflexión

- Los patrones de diseño son soluciones frecuentemente probadas para resolver problemas comunes de diseño en software. La utilización de estos patrones tiene la ventaja de que facilita la creación de sistemas más flexibles y confiables.

Sin embargo, el diseño obviamente que el

uso excesivo de recursos de diseño puede llevar a la complejidad innecesaria, lo que podría resultar contraproducente en términos de mantenimiento y comprensión del sistema.

Camilo G.D (1009) factores de diseño, (reducción y priorización). Diferencias entre las versiones no se utilizan en el software Orientado a Objetos. Cuadernos de trabajo, (A), 107-136.



## Evolución e integración de aplicaciones legacy.

Comenzar de nuevo o actualizar?

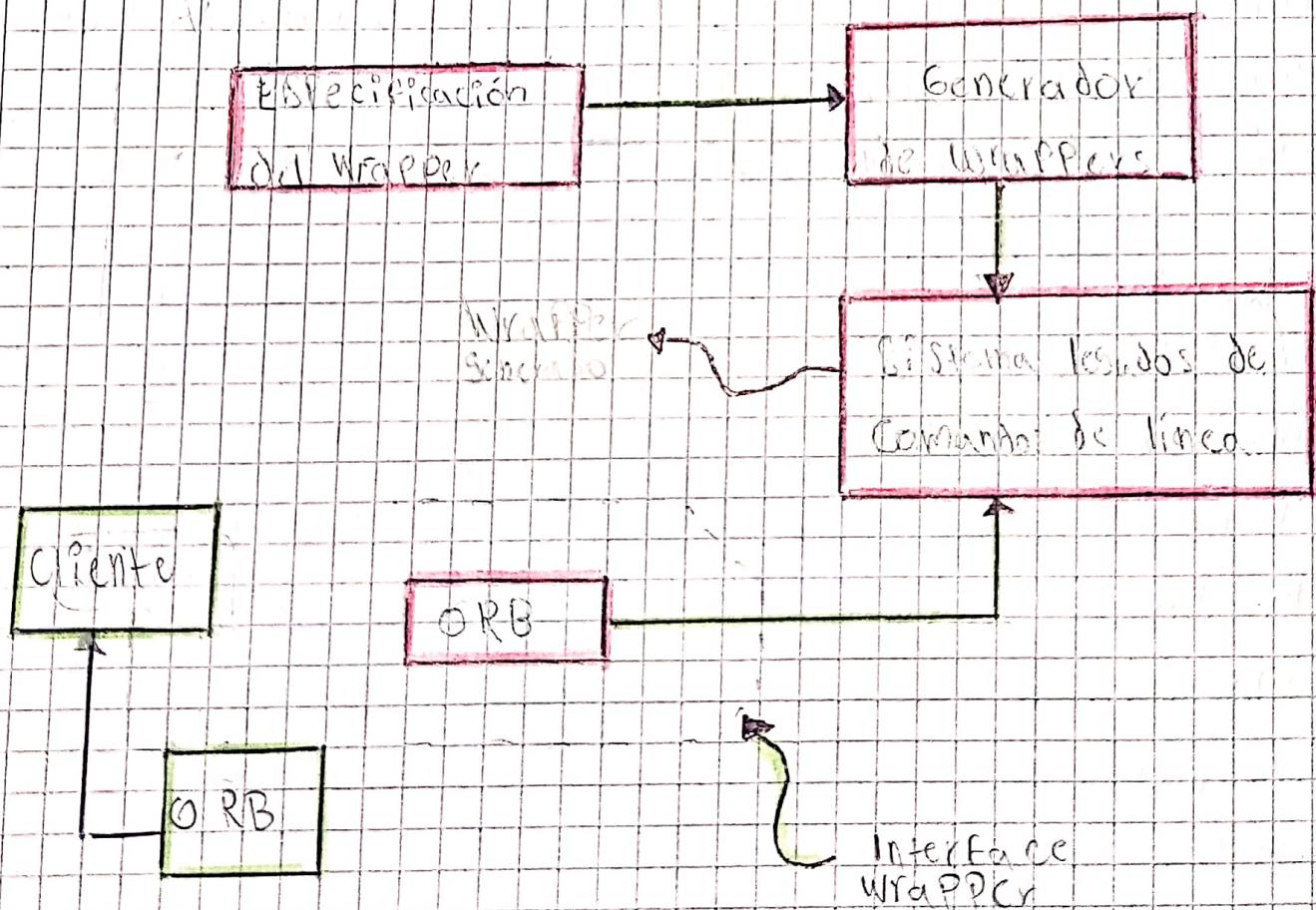
El artículo explica que muchas organizaciones aún dependen de estos sistemas viejos que son difíciles de modificar y mantener. Se presentan varias opciones para abordar ese problema, como reemplazar completamente el sistema, mantenerlo como está, o aplicar técnicas de "reingeniería" para minimizar gradualmente. El artículo se enfoca principalmente en la reingeniería, describiendo diferentes enfoques como el uso de "wrafsatz" para envasar el sistema viejo, la conversación de características en componentes más pequeños, y la integración con servicios web.

### Reflexión:

Al leer este artículo sobre la evolución e integración de aplicaciones legacy, me quedó con varias cuestiones interesantes. El artículo expone la problemática de los sistemas de software antiguos que siguen siendo vitales para muchas aplicaciones, pero que se han vuelto difíciles de mantener y actualizar.

En cuanto a la arquitectura de software, el éxito destaca cómo los sistemas legados suelen tener diseños monolíticos que dificultan su evolución, y propone técnicas como el uso de wrappers, patrones de diseño y programación orientada a aspectos para modularizar y flexibilizar estas arquitecturas antiguas.

Pedraza-García, G. (2008). Evolución e integración de aplicaciones Business: Comenzar de nuevo o actualizarse? Congreso (domótica) de Configuración



Patrones de ataque y de seguridad como guías en el desarrollo de software

Este artículo explora el uso de Patrones de Seguridad y Patrones de ataque como guías para desarrollar software más seguro. Se centra en la arquitectura de software al analizar cómo estos patrones pueden integrarse en las diferentes etapas del ciclo de desarrollo. Los patrones de seguridad ofrecen soluciones prácticas a problemas de seguridad comunes mientras que los patrones de ataque describen maliciosas utilidades para explotar vulnerabilidades. El artículo destaca la importancia de catalogar y relacionar ambos tipos de patrones para facilitar su aplicación práctica.

## Reflexión

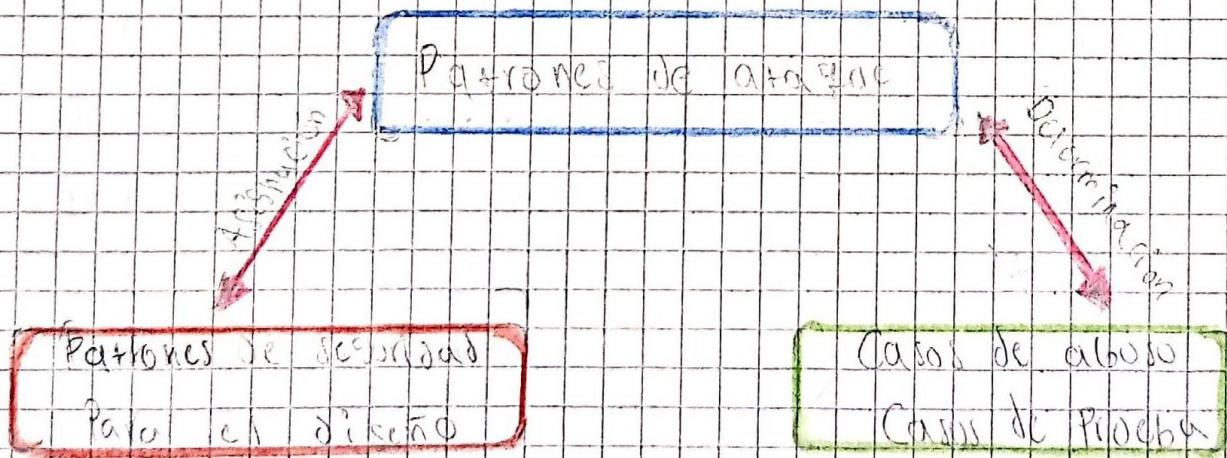
Después de leer este artículo sobre patrones de ataque y patrones de seguridad en el desarrollo de software, me quedó una impresión muy buena. El artículo explica cómo estos patrones pueden servir como guías para una implementación más segura de las aplicaciones del desarrollo. Me llamó

Tu atención como los patrones de ataque  
describen las técnicas que usan los atacantes  
métodos que los tienen de seguridad deben suscitar.

Ramos, J. (2015). Sistemas de Ataque y de Seguridad

como Guía en el desarrollo de Software. In VIII

Concurso de Trabajos Científicos (CST 2015) - Jairo  
Gómez (2015).



Desarrollo de Software Seguro y su relación con el Cuerpo de conocimiento para la Ingeniería de Software.

Este artículo analiza la perspectiva de la seguridad del software. Los autores argumentan que, aunque SWEBOK menciona aspectos de seguridad, no les da suficiente énfasis dado al creciente impacto de los sistemas de seguridad en el desarrollo de software. El artículo sugiere como integrar consideraciones de seguridad en cada una de las etapas del SWEBOK, ofreciendo herramientas, métodos y mejores prácticas específicas. Por ejemplo, en el área de requisitos, se sugieren técnicas hechas como SWARE para validar las necesidades de seguridad.

## Reflexión

Después de leer este artículo sobre el desarrollo de software seguro y su relación con el Cuerpo de conocimiento de la Ingeniería de Software, me acuerdo con varias reflexiones interesantes.

Me llamó la atención cómo los autores

Sugieren incorporar prácticas, herramientas y métodos específicos de seguridad en las distintas áreas de conocimiento del SWEBOK. Los autores enfatizan que la seguridad debe ser una preocupación transversal en todo el ciclo de vida del desarrollo, no solo una consideración posterior.

Vega, V., R. Garsia, G. P., Carrasco, J., S. TANAK, C. Vizcarra. Desarrollo de Software Seguro Y su Relación con el Ciclo de Consumo Para la Ingeniería de Software.

Desarrollo de Software

Seguro en SWEBOK

Propuesta de Integración

Técnica SQARE

Para Seguridad

Patrones de Seguridad

Análisis de riesgo

Técnicas de

Programación segura

Aportes para una Internet de las Cosas Seguras.

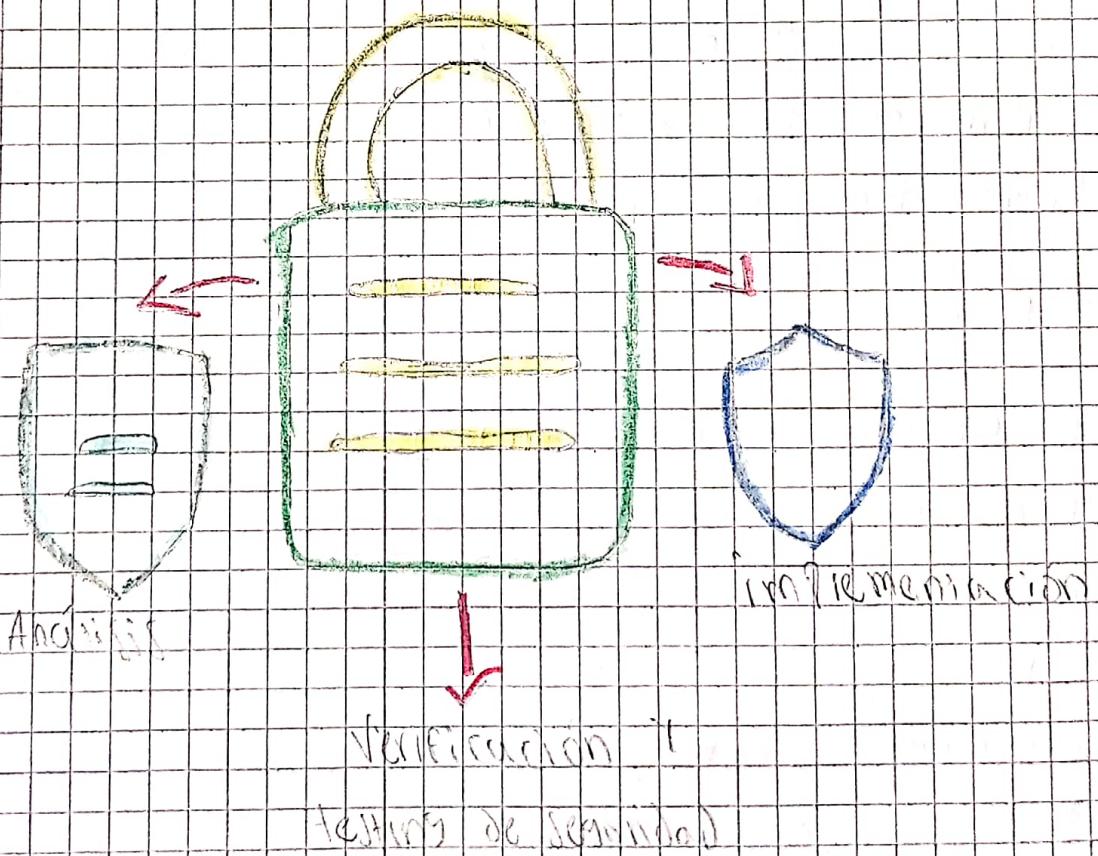
Este artículo aborda la siguiente situación:  
Se señalan en los sistemas embedidos que se  
están desarrollando actualmente, especialmente  
aquellos que aspiran a conectarse a Internet. Los  
autores proponen tres principios esenciales para  
mejorar lo siguiente en la arquitectura de estos  
sistemas; los autores argumentan que adoptar  
estos principios en la arquitectura de los sistemas  
embedidos y ciber físicos puede contribuir  
significativamente a crear una "Internet de las  
Cosas" más segura.

### Reflexión

El artículo responde tres principales cuestiones  
para mejorar la seguridad: tratan los requisitos  
de segundo orden (que funcionales deseó el inicio  
del diseño del sistema), tratan las metas para  
analizar detalladamente el comportamiento  
y las amenazas y análisis futuros de diseño  
y diseño de seguridad probados.

Siguen usando técnicas como casos de mal uso  
para requerimientos de seguridad, el modelo es el  
Sistema Y sus componentes para cumplir concretos.

Salinas, M., Jorge, J.A., Casanovas, E., & Tariq, C. (2011).  
Algoritmos para una Internet de los Casos Suspiciosos.



Hacer lo más de conocimiento para auditorías de  
seguridad en servicios web con SQL injection

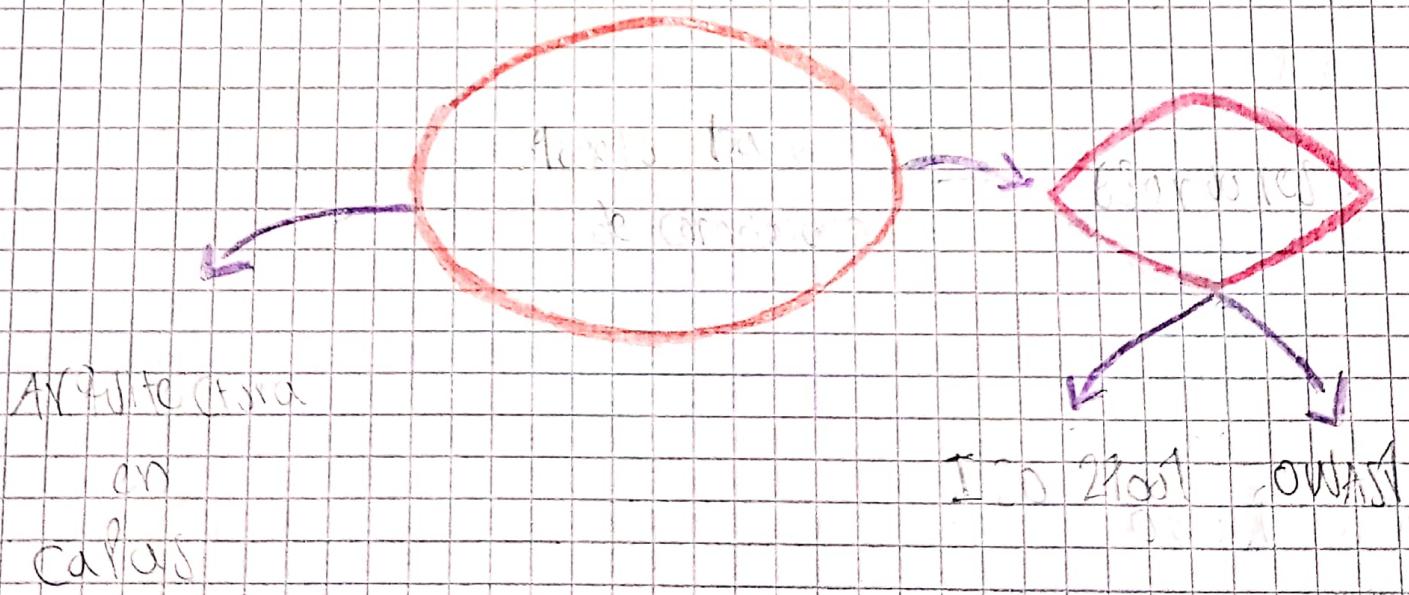
Este artículo presenta un modelo de base de conocimiento  
para realizar auditorías de seguridad en servicios web,  
entacandose específicamente en ataques de inyección  
SQL. El modelo propone una arquitectura de  
software que integra varias capas y componentes.  
Para abordar la seguridad web de manera integral,  
se basa en estándares y buenas prácticas de  
seguridad como ISO 27001, OUBAS y metodologías  
como OSSTMM. La arquitectura incluye un modelo  
de auditoría web, una base de conocimiento, y  
componentes para manejo, ejecución y generación de  
informes.

## Reflexión

Este artículo me hizo reflexionar sobre lo importante  
que es transferir la seguridad de los sistemas  
al mundo empresarial web. Me sorprendió ver cómo  
propone un enfoque integral, combinando una base  
de conocimientos con auditorías de seguridad. La  
idea de crear una arquitectura en capas para

organizar todos la información de servicios me pareció muy inteligente. Me gustó que no solo se enfocan en detectar problemas, sino también en educar y prevenir futuros ataques.

Marín J. E. M. (2019) Modelo base de conocimiento para auditores de seguridad en servicios Web con SQL Injection (Master's Thesis, Universidad Distrital Francisco José de Caldas (Caracas)).



Modelo Y Herramientas Software Para la Gestión  
de Riesgos en el Desarrollo de Aplicaciones  
Web Sostenido en el Estándar ISO/IEC 29100

El modelo ISO/IEC 29100 propone una arquitectura  
de software con tres perspectivas concorrentes:  
lógica y física. La perspectiva conceptual  
de tiene 5 fases: Creación del Proyecto, Planificación,  
Evaluación, Identificación de riesgos y Gestión de  
riesgos. La perspectiva lógica utiliza diagramas  
UML para modelar las interfaces y relaciones  
del sistema. La perspectiva física define la  
estructura de la base de datos. El modelo solo  
soporta ciclos de vida incrementales e iterativos,  
y está orientado a proyectos medianos y grandes.

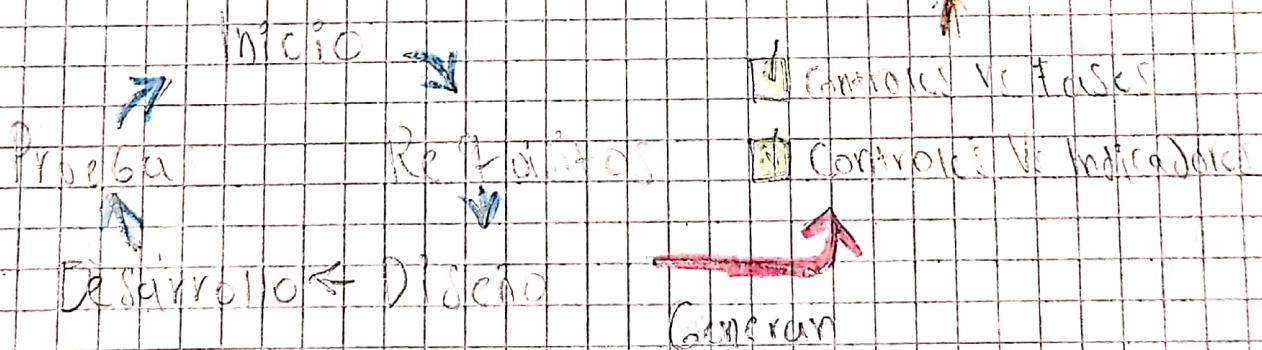
## Reflexión

Este artículo me hizo reflexionar sobre la importancia  
de considerar los riesgos desde el inicio al  
desarrollo de aplicaciones web. Me pareció interesante  
cómo trae un modelo integrado que combina  
estándares de seguridad con las fases típicas de  
desarrollo de software.

Me gusto que no solo se enfoca en la matemática, problemas, sino también en las aplicaciones prácticas de la ciencia. Con estos me hice pensar en lo utilizable que pueden ser las aplicaciones web si no se siguen adecuadamente los mejores goede standards.

Anaíta Cuellar; A.J. (2017). Microlo : herramienta software para la gestión de riesgos en el desarrollo de aplicaciones web. Se obtiene en el estándar ISO/IEC 27005.

Listas de check list



Capítulo 5: Los Patrones de Diseño en un contexto de desarrollo dirigido por Pruebas

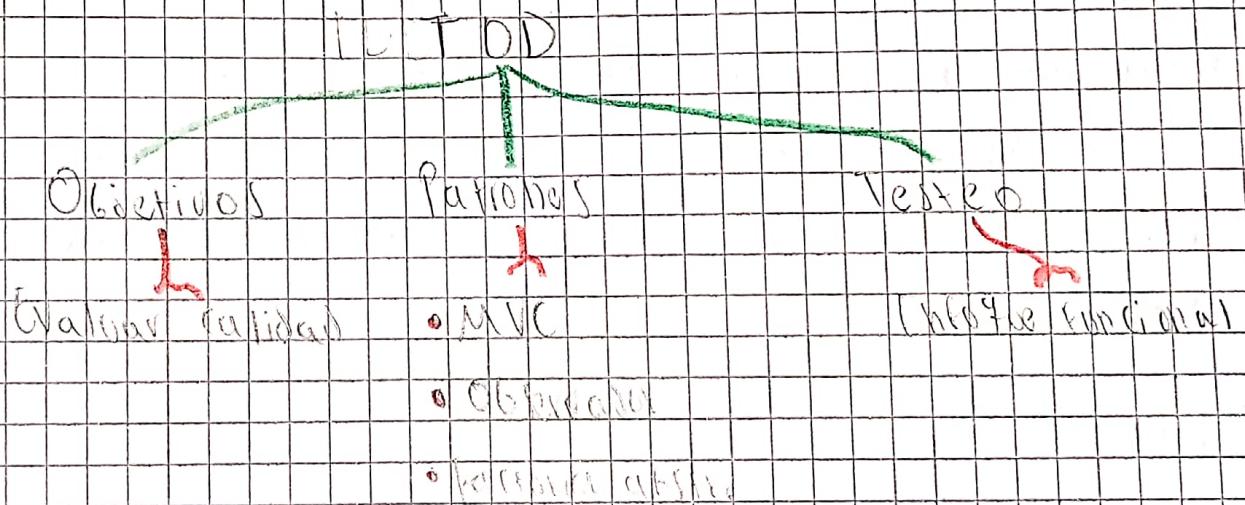
- Este capítulo analiza la aplicación de los patrones de diseño en un contexto de desarrollo dirigido por pruebas (TDD), enfocándose en su aplicación.
- Los autores seleccionan tres patrones (observador, delegado, composito) y evalúan si pueden usarse efectivamente con pruebas unitarias.
- Autores también evalúan si los patrones de bien común como tener un efecto fundamental y no mantener este efecto intenso.

### Reflexión

- Este artículo me hizo reflexionar sobre como la aplicación de software puede ayudar en las necesidades de desarrollo, específicamente en desarrollo dirigido por pruebas (TDD).
- Cómo los patrones de diseño
- nos permiten ver si son compatibles con TDD.
- La idea de establecer criterios para determinar si un patrón se puede probar eficientemente me pareció muy útil.

Me gustó que no solo se enfocó en la teoría, sino que aplica estos conceptos a un caso específico de un sistema de control de velocidad.

Catalán, M. I., Giménez, A. C., & Gómez, L. (2011).  
Patrones de Diseño en un contexto de Desarrollo  
Dirigido por Pruebas.



Suite de pruebas auto-ejecutable como examen de laboratorio: una aproximación (automática) con Spring Boot y GitHub

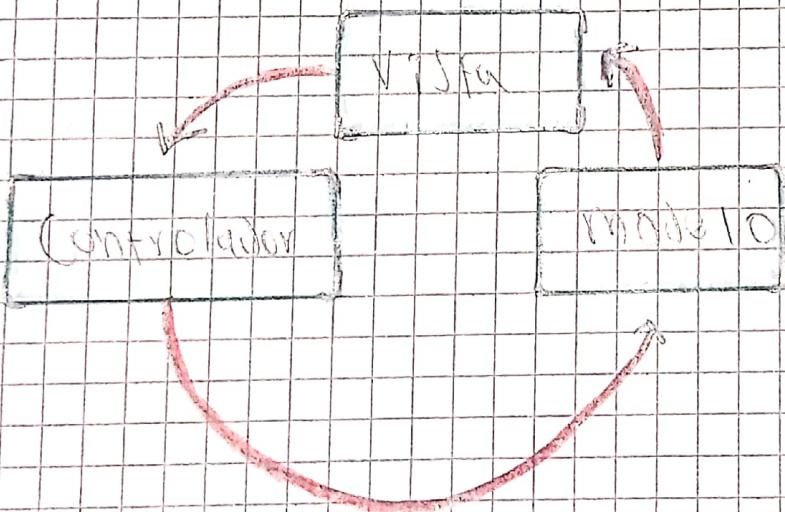
Este ejercicio presenta una aplicación Java sencilla que realizan exámenes prácticos de programación que se pueden evaluar automáticamente mediante librerías universales. Utiliza el framework Spring Boot y su arquitectura en capas (Modelo-Vista-Controlador) como base para los ejercicios del examen. Evita la creación de aplicaciones de páginas de diseño como implementación de interfaces y modo Objeto-relacional. Muestra el cumplimiento de las siguientes características como la separación entre capas.

### Reflexión

Este ejercicio me hizo reflexionar sobre cómo el aprendizaje de software puede darse de manera práctica en un entorno educativo. Me pareció interesante cómo los autores proponen usar pruebas automatizadas para ejercicios no solo en código, sino también aspectos de diseño y arquitectura.

Por ejemplo mencionan que pueden comprobar si los estudiantes aplican correctamente el patrón MVC y la arquitectura en capas, restando náculas incorrectas entre capas.

Pareja, J. A., Calvarius, C., Jiménez-Torres, B., García, J. M., Müller, C., & Resinas, M. (2023). Suite de herramientas auto-evaluables como examen de laboratorio: Una aproximación pragmática (en Spring Boot + GitLab).



## Conceptualización del proceso de implementación de software: perspectivas ágil y disciplinada

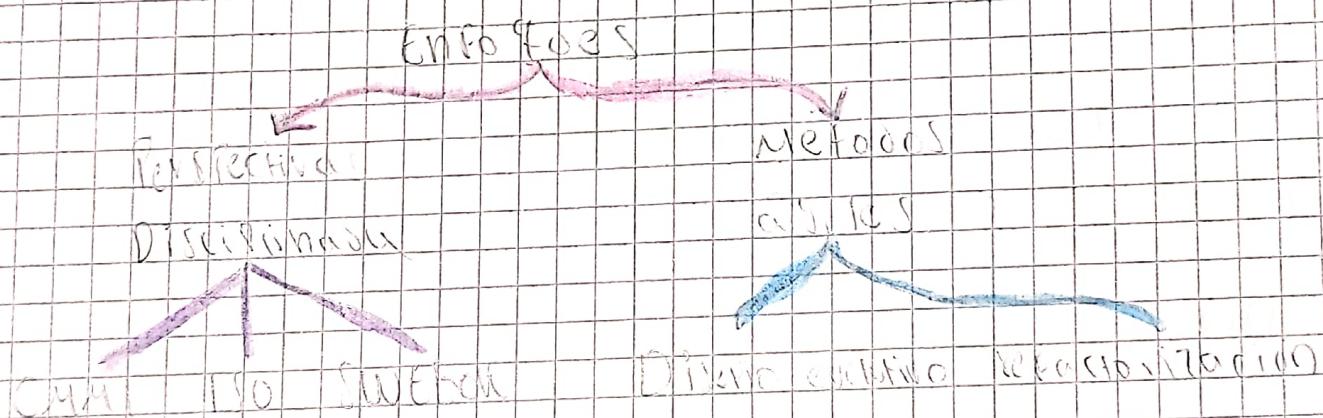
- El artículo aborda la implementación de software desde dos enfoques: ágil y disciplinado. Se propone un modelo híbrido que combina ambas perspectivas para lograr procesos eficientes y de calidad en el desarrollo de software. Desde la perspectiva disciplinada, se mencionan estandares como CMMI, ISO/IEC 12207, y SWEBOK, los cuales enfatizan procesos rigurosos y documentados. Por otro lado, los métodos ágiles como XP y AgileUP se centran en la simplicidad, la retroalimentación constante y la entrega incremental.

### Reflexión

Este artículo me ha dejado una impresión sobre cómo se puede lograr un equilibrio entre las enfoques ágiles y disciplinados en la implementación de software. Me pareció interesante cómo los autores analizan diferentes modelos y estrategias para optimizar los resultados claves relacionados con la arquitectura de software.

Noté que tanto los métodos existentes como los adicionales  
consisten en que van formar debe ser este diseño  
mencionado en el artículo.

Castillo, A., Bartolos, J., Montaña, J. & Rivero, O. (2010).  
Concentración del acceso se implementando de  
Software: *Access 2007* y disciplina. *Ciencia e  
Ingeniería*, 21(3), 143-152.



Arquitecturas de software para el desarrollo de videojuegos sobre el motor de juego Unity 3D

El artículo se enfoca en el desarrollo de una arquitectura de software para videojuegos utilizando Unity 3D, detallando la importancia de su uso colectivamente los componentes del sistema. Muestra una combinación de arquitecturas en capas y basadas en componentes para la ejecución de las funciones principales, garantizando calidad, mantenibilidad y extensibilidad. Se detallan el uso de patrones como Singleton y observadores para eventos, así como buenas prácticas de diseño y código reutilizable.

En resumen, el artículo destaca la necesidad de una arquitectura modular y flexible que facilite la reutilización de componentes del sistema.

## Reflexión

Al explorar la propuesta de una arquitectura modular para juegos con Unity 3D, resulta útil implementar y extender los componentes de un videojuego de manera organizada y eficiente.

Lo interesante es cómo la arquitectura se conecta en el suerte que une creatividad y funcionalidad, permitiendo a los desarrolladores construir juegos que sean mantenibles, flexibles y reutilizables.

Además, reflejan la evolución de la industria del videojuego donde las exigencias técnicas y de experiencia del usuario son cada vez mayores.

Pau, H.H., Falco J.D., Cret, A.A.S. (2018) La evolución  
de software para el desarrollo de videojuegos.

Capítulo 10

Java Swing

\* Game Manager

\* Level Manager

\* UI Manager

\* Transiciones

\* State Machine

\* Eventos

Modelo para la ayuda a la toma de decisiones en la selección de patrones de desarrollo de software

El artículo trata sobre un modelo para ayudar en la toma de decisiones al seleccionar patrones de diseño en el desarrollo de software. La dificultad de mantener un diseño consistente se facilita con el uso de una estructura jerárquica.

El modelo propuesto divide el diseño en tres niveles jerárquicos: estrutural, funcional y patrón. Los patrones se dividen en tres tipos: adaptadores, modificadores y personalizadores, según las necesidades de un diseño.

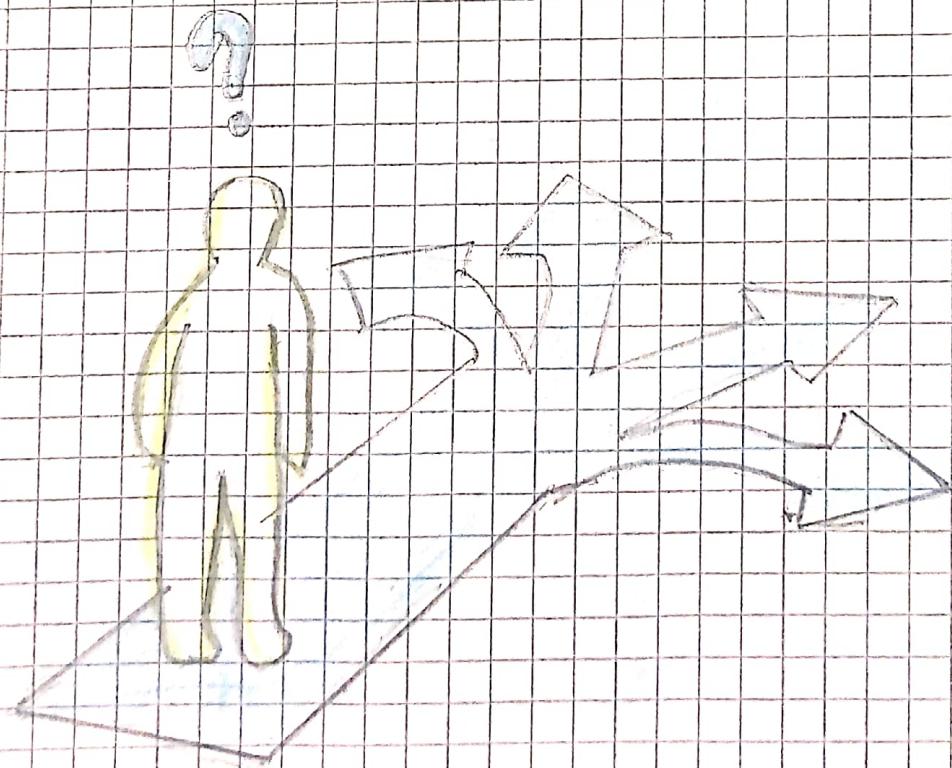
## Resumen

Después de leer este artículo, me ha quedado claro que la arquitectura de software juega un papel crucial en la toma de decisiones cuando se trata de elegir patrones de diseño adecuados.

El texto indica cómo los patrones de diseño son instrumentos valiosos para resolver problemas comunes en el desarrollo de software, pero

Tambien reconoce la dificultad que enfrentan  
muchos desarrolladores, especialmente aquellos con  
menos experiencia, para identificar los fallos o errores  
correspondientes.

Villa, M.C. & Torello, P. (1990) Modelos para la ejecuci  
on de tipos de decisiones en la sustitucion de factores  
de desarrollo de sistemas.



Modelo teórico para la implementación del  
acoplamiento "Stoneware System" en la etapa  
de la implementación de una estructura de  
software.

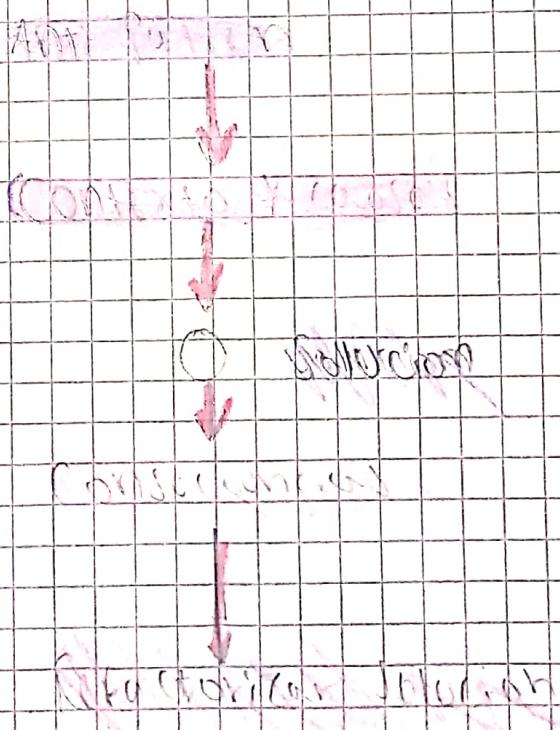
El artículo trata sobre el análisis de  
arquitectura de software conocido como "Stoneware  
System", que se basa cuando los subsistemas de un  
sistema están más integrados, ofreciendo una  
interacción, manejo y mantenimiento.  
Este sistema tiene una alta calidad de integración  
y clasificación y clasificaciones claras, lo  
que garantiza sistemas más simples de manejo.  
Al hacer el documento finales es importante tener  
en cuenta las necesidades de identificación de  
los problemas y sus soluciones.

### Reflexión

Después de leer el artículo sobre el análisis  
"Stoneware System" en arquitectura de software,  
me he reflexionado sobre cómo se necesitan  
las soluciones más sencillas que se conviertan  
en fuertes obstáculos para los proyectos.

La arquitectura de software no solo garantiza sistemas, sino que debe permitir flexibilidad y buena integración entre componentes. Este anti pattern suele ocurrir cuando los subcomponentes están bien diseñados y no se comunican bien, pudiendo provocar inefficiencias, costos elevados.

Referencia: I. C. C. Modelo Técnico para la Identificación de Arquitecturas "Star Trek System" en la Etapa de la Implementación en los Procesos de Software.



Una arquitectura para una aplicación infinitamente evaluación basada en patrones de diseño  
componentes y agentes bajo paradigma de VBE  
la arquitectura se basa en el modelo UML  
UML + LTSA, haciendo elementos como las  
rule o conocimiento y metáreas autorizadas  
sobre flujos de trabajo y existencias del sistema.  
Entre los nuevos usuarios están el MVR que  
está en la ejecución y presentación, y el Observador  
que maneja "información" la mantiene con los  
agentes, también "navegar" componentes de diferentes  
siguientes que facilitan el manejo del cambio y  
el reutilizable.

### Reflexión:

- Se presenta la idea de factores de diseño y  
componentes para constituir un sistema robusto
- Que permite personalizar las evaluaciones para  
cada estudiante. Parece interesante como  
combinan herramientas como agentes y evaluaciones
- Y bases de conocimiento para establecer y aplicar  
los criterios de los estudiantes autorizaciones.

- Esto demuestra como una buena arquitectura puede ir más allá de resolver problemas técnicos, ayudando a mejorar experiencias humanas en este caso, el aprendizaje.

Nardijsma, J., Vanekoma, M., & Oerlemans, B. (2012).

Una arquitectura para una educación inteligente de evaluación basada en factores de diseño constructivos. La mejoría hacia el aprendizaje de WBL.

Moveto de Currículum



## FATIA's Framework para la enseñanza de agentes

### (ii) IA

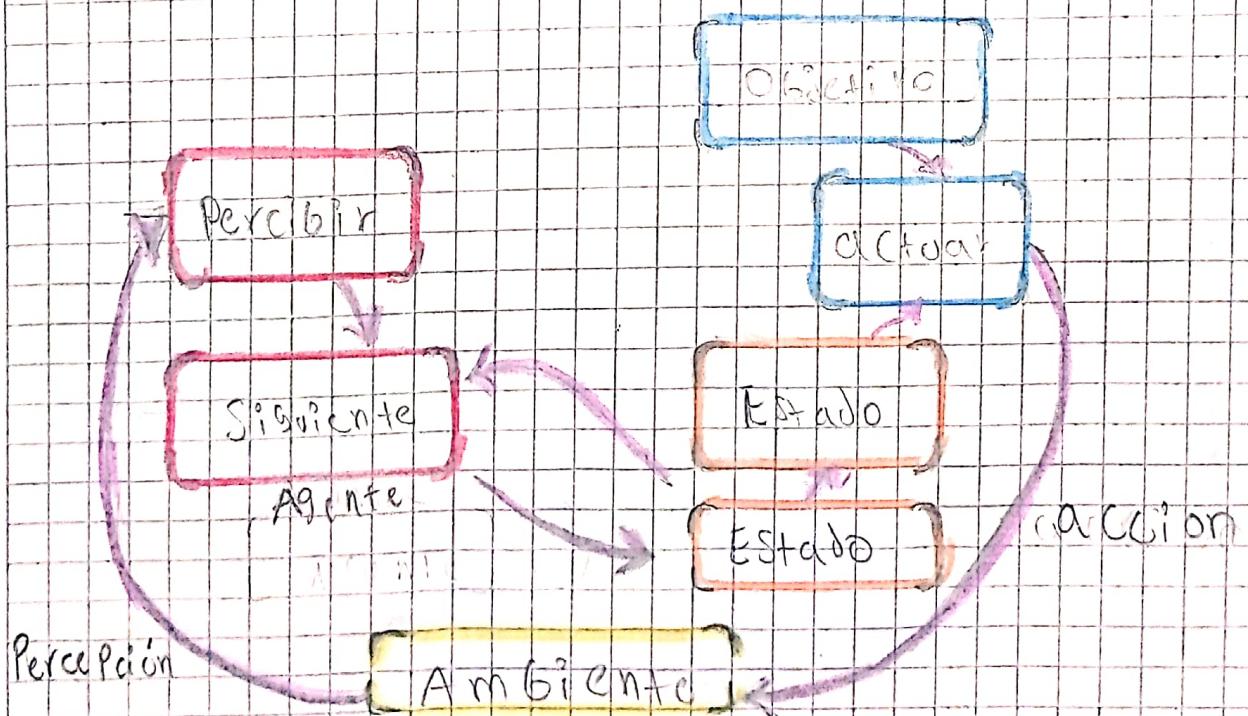
- Este artículo había llamado FATIA, que fue desarrollado para ayudar a estudiantes universitarios a aprender sobre Inteligencia Artificial ejecutando caminos sobre agentes inteligentes. La más importante de todo el punto de vista de arquitectura de Software es que el framework facilita fases de diseño como Strategy y técnicas de trabajo en paralelo organizadas en objetos para crear una estructura base mediante la cual los estudiantes pueden extender.

### Reflexión

- El artículo "FATIA: Framework para la enseñanza de agentes en IA" aborda el diseño de un marco de trabajo (framework) destinado a ayudar a estudiantes de ingeniería (como implementar agentes inteligentes en el campo de la Inteligencia Artificial (IA)).

FAIA permite a los alumnos enfocarse en los aspectos conceptuales del diseño de agentes, mientras abstraen la complejidad técnica del simulador.

Roa, J., Gómez, L., M., & Segura, G. (2008). FAIA: Evolución para la enseñanza de agentes. IA TEC (Introducción a las técnicas de programación de robots móviles), 1(1), 43-77.



Método de automatización del desarrollo consiste en la rule para la implementación de microservicios.

Este artículo analiza la implementación de la arquitectura de software basada en microservicios, un enfoque que permite dividir aplicaciones complejas en pequeños servicios independientes. La arquitectura de microservicios aborda necesidades como la escalabilidad y el despliegue continuo. Los microservicios permiten una mayor flexibilidad y adaptabilidad para el desarrollo. Destaca el uso de herramientas como Kubernetes que facilitan la ejecución y funcionamiento de servicios constituyentes en entornos de prueba y producción.

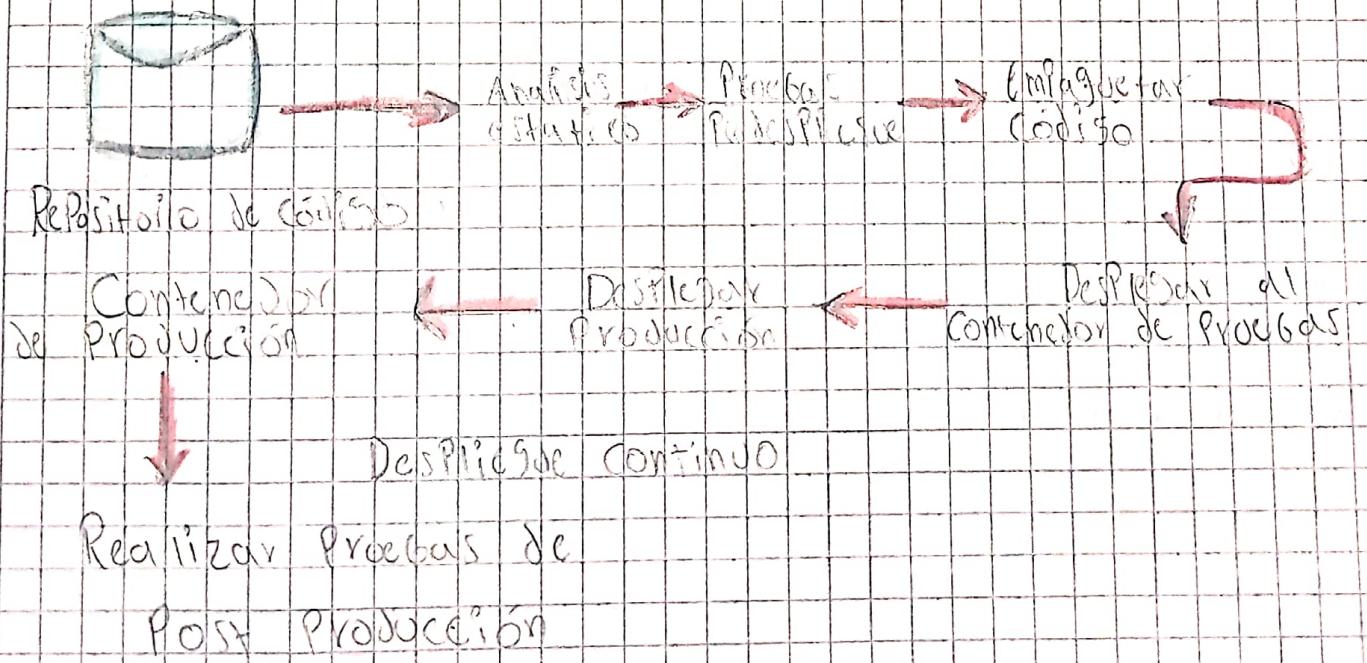
## Reflexión

Este artículo me hizo reflexionar sobre cómo la arquitectura de software se adapta a las necesidades actuales de las empresas, específicamente con los microservicios. Es interesante ver cómo esta metodología divide aplicaciones grandes en pequeñas partes independientes, lo que permite una flexibilidad increíble.

lo que más me llamó la atención es cómo DevOps  
y la automatización del despliegue continuo  
transforman los procesos de desarrollo y despliegue  
en algo más ágil y seguro.

Vera-Rivera, F.H. (2018). Método de automatización del  
despliegue continuo en la nube para la implementación  
de microservicios. In: IEEE. 2018 Latin American  
Workshop on Cloud Computing (CLOUD) (VI. 2018 - 60A).

Integración continua:



Comparación de uso del patrón de diseño  
decorator Y la programación orientada a objetos  
en .NET para manejar las inconsistencias (chimeneas)

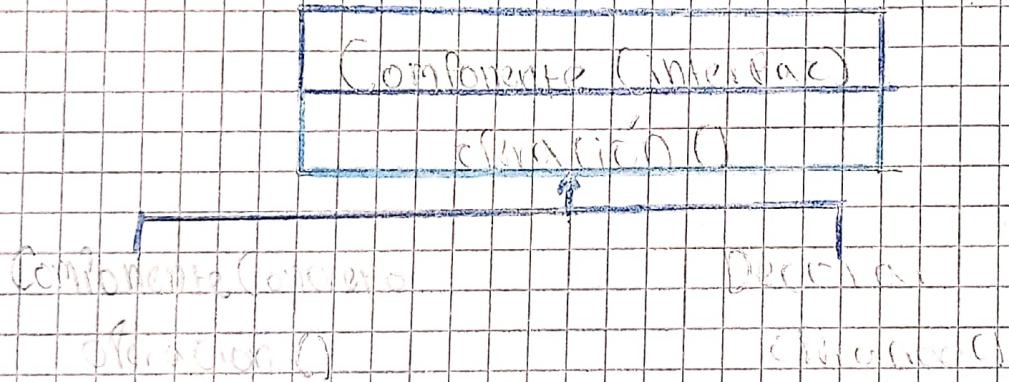
En el artículo se mencionan como los principales:  
- En el desarrollo de software .NET, comando 21 (el uso  
del patrón de diseño Decorator Y la programación  
orientada a objetos (POO) con PostSharp. Además  
estudiarán brevemente el problema de las  
inconsistencias (chimeneas) que son funciones comunes  
como la validación de acciones (logging) (que afectan  
a todos los partes del sistema) Y comunicar su  
funcionamiento. Detallarán también el efecto  
específico de los sistemas que tienen una  
mejoría en la compilación de código.

## Reflexión

El artículo hace un repaso de cómo manejar  
problemas comunes en desarrollo, como las "inconsistencias  
(chimeneas)", que son características que se refieren  
a todos los módulos Y comunican su gestión.  
Se comparan las ventajas del patrón de diseño  
Decorator Y la programación Orientada a Objetos.

Ambos tienen como objetivo mejorar la  
modularidad, pero siguen caminos diferentes  
para lograrlo.

Palma-Vasquez, C. A., Viñuel-Silva, C. L., & Montes,  
M. A. (2017). Comunicación de uso del patrón  
de Diseño Decorador y la Abstracción Orientada  
a Aspectos (AO) para modularización en sistemas  
Gráficos de Información tecnológicas (SIST). *32*, 49.



Especificando una arquitectura de software

El artículo habla sobre la arquitectura de software  
y como se usan los patrones para implementar  
de manera más eficiente. Explica que con  
el avance de la tecnología y el aumento de  
tareas complejas, es necesario tener estructuras  
básicas que cumplen los fines; segun el artí-  
culo fundamentalmente, los patrones arquitectónicos  
nos ayudan a organizar y categorizar los componentes  
y sus interacciones en categorías segun  
- su tipo de trabajo o función.

## Reflexión

Despues de leer el articulo sobre la especificación  
de la arquitectura de software, que he aprendido  
es que las soluciones tecnológicas deben ser  
diseñadas de manera organizada y eficiente.  
se basa en la implementación de utilitarios  
y recursos disponibles, que no son más  
que guías para tratar a estrucuturado y  
simplificando la comprensión de los sistemas.

Los Patrones no son recipientes mágicos, sino  
herramientas que, son experiencia (creativa),  
permiten resolver problemas comunes en el desarrollo  
de software.

Quigley, L. & Vargiu, S. (2010). Evolucionando una  
arquitectura de software. Tesis doctoral, Investigación y  
Academia, 16(2), 136-179.

### Patrones arquitecturales

Estructura de dominio de diseño

La programación Reactiva. Un nuevo enfoque para trabajar con código asíncrono en la programación Web.

La programación reactiva es un enfoque innovador para gestionar eventos y código asíncrono, específicamente en el desarrollo Web con JavaScript. Esse paradigma permite tratar flujos y datos (events) que suelen transformarse y combinarse usando operaciones como map, filter y reduce. A continuación se detallan algunos ejemplos prácticos, las ideas clave como remitentes, técnicas simbólicas basadas en variables libres y combinaciones posibles con sus respectivas descripciones:

- En función de la ejecución (mediante la reusabilidad del código).

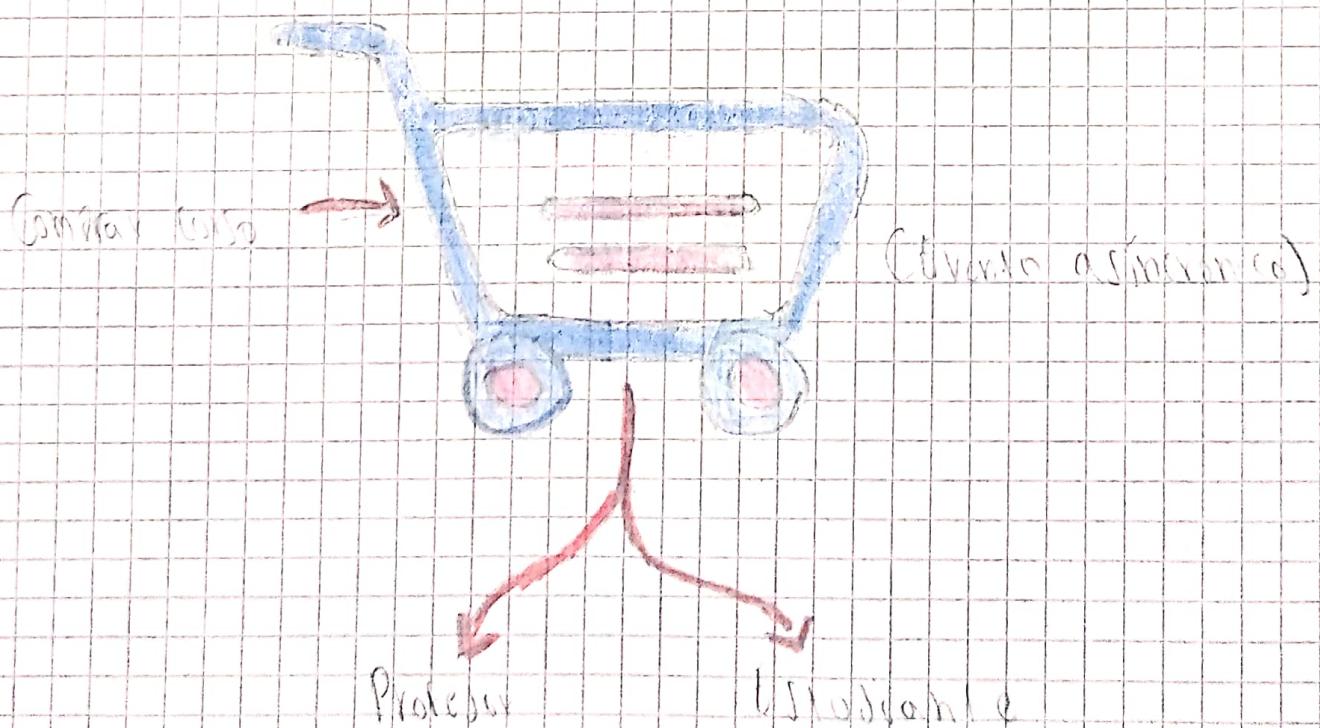
### Reflexión

- Describir de qué se trata, su programación con React, no es solo una herramienta técnica, sino un enfoque completamente nuevo para organizar la ejecución interna de software. Esse paradigma aborda un problema muy común en la programación

Modernas como vienen en los VAGENS  
asincronizadas de forma más eficiente y limpia.

Al comparar la sincronización individual, uno se  
varía las fases y esas fases comienzan con  
el empleo de variadores.

Fernández Viñas, W. (1991) La sincronización individual  
en motores同期化のための回路の構成とその効率の評価  
en la sincronización individual. Tesis doctoral. Universidad  
Nacional de La Plata.



## Desarrollo de aplicaciones basadas en microservicios. Tendencias Y Desafíos de investigación

El artículo nos enseña un diseño que divide las aplicaciones en pequeños servicios independientes que interactúan a través de APIs bien definidas. Este modelo ofrece beneficios como escalabilidad, agilidad en cambios y actualizaciones rápidas. Se distinguen los servicios y su interacción con las interfaces de usuario. También se mencionan ejemplos reales como Netflix, que agiliza la entrega de contenido y el nombramiento de los servicios individuales.

### Reflexión

Después de leer el artículo sobre el desarrollo de aplicaciones basadas en microservicios, entiendo que como una organización multistrukturada, la idea es que se dividan las aplicaciones hoy en día. La idea de dividir una aplicación en pequeños servicios individualmente hace muy atractivo que realmente un desarrollo más sencillo sea posible.

Se restauran ventanas como la escalera, y  
la facilidad para realizar actualizaciones.

Vera-Sivera, F.H., Asturillo, H., & Gómez, L. (2019).

Desarrollo de aplicaciones basadas en tecnologías,

tendencias y desafíos de investigación. *Educación*, 20(19) 107-110

Características de

aplicaciones

y

Características estructurales de comportamiento

# Influencia de la evolución de Software en Plataformas móviles de comercio electrónico

Los avances tecnológicos de información han facilitado el surgimiento de nuevos tipos de servicios, como las aplicaciones móviles y el comercio electrónico. Los cambios tecnológicos han permitido a las empresas ofrecer una amplia gama de servicios y productos. Estos cambios han impulsado la transformación digital de las empresas, lo que ha llevado a una mayor eficiencia y productividad.

## Reflexión

La evolución constante de la tecnología en aplicaciones móviles y el comercio electrónico ha llevado a una mayor demanda de información y comunicaciones entre los usuarios. Estos cambios representan nuevas formas de consumo y acceso a productos y servicios. Estos cambios presentan una oportunidad para combinar la fuerza de las tecnologías de información con las necesidades de los consumidores.

• a las necesidades de usuarios que existen  
Plataformas variadas, seguras y confiables.

• Un factor crucial en el diseño web es:

Buenas J.D.F.C., user, diseño M.E. y diseño  
(UIA). Involver a la audiencia de  
clientela en plataformas variadas de consumo web con el  
fin de obtener todo el conocimiento posible científico -  
profesional, etc.

L. Colegio

Casa en Colonia La Gata de  
Centenario y Paseo de los

Diseño de una arquitectura de servicios  
informáticos para implementar la seguridad  
de información en la empresa Batink S.A.C

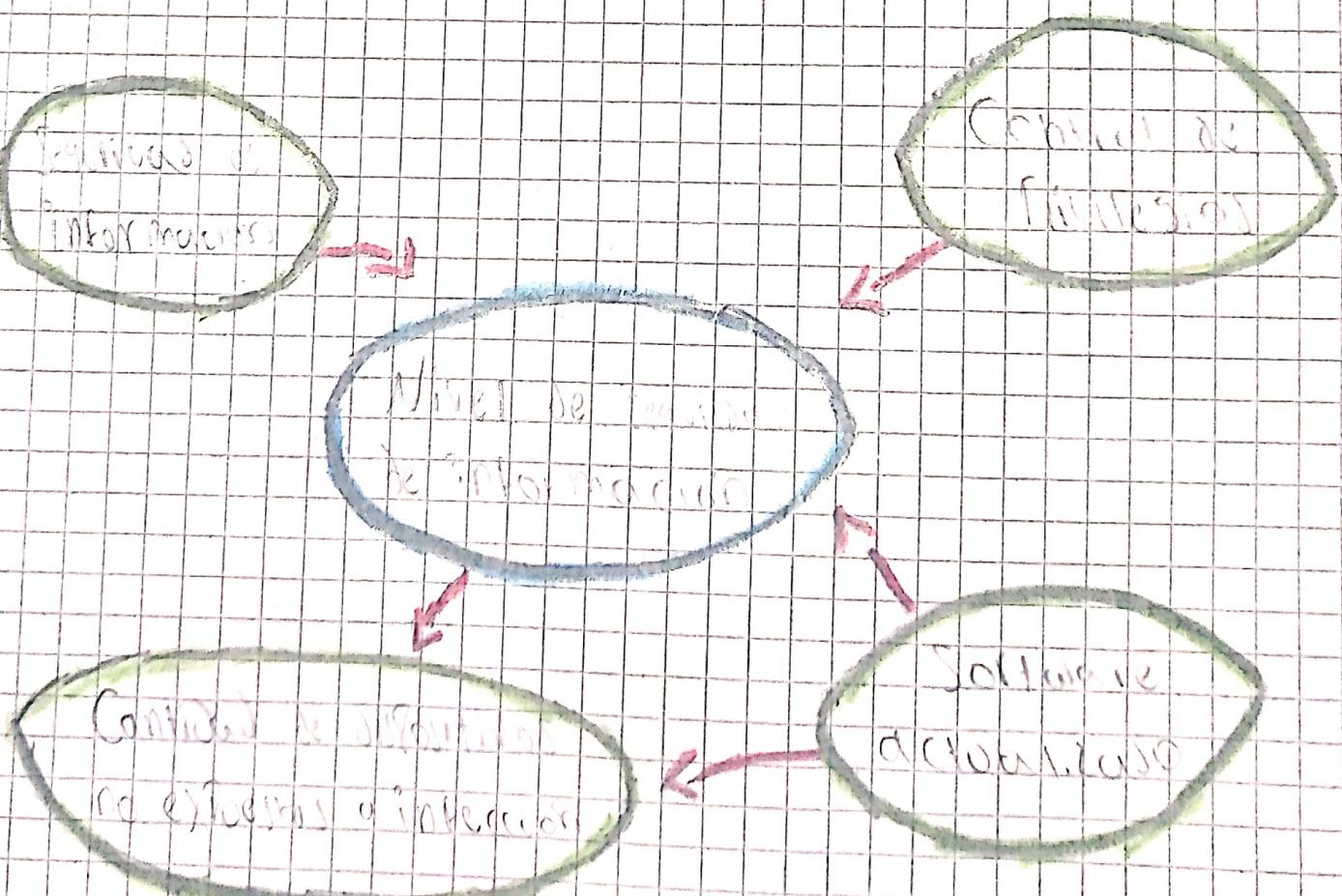
El proyecto busca desarrollar como una organización  
de servicios informáticos bien diseñada, puede  
mejorar la protección de la información en la  
empresa BATINK S.A.C., ejecutando un  
número de servicios de información. Con un  
enfoque estratégico para todo lo que se requiere  
• Implementación de Seguridad Intel-  
ligencia para optimizar la información en la empresa  
• Identificación de riesgos y vulnerabilidades  
• La integración de servicios de información.

## Referencias

Este trabajo incluye la información necesaria con  
una arquitectura de servicios informáticos, voluntad  
para proteger la información, un diseño  
fundamental en una empresa en la medida  
que evaluar diferentes componentes de seguridad, se  
demonstra. Que no solo es posible mejorar la  
protección de datos, sino también optimizar

- Revision de Selección de los componentes más adecuados. La experiencia de DATING S.A.C
- sobre en el diseño un análisis demandado.

ASORIA GARCIA, M.D. (2012). Típico de una arquitectura de servicios informáticos para implementar la gestión de información en la empresa DATING SAC en 2011.



Arquitectura de Convergencia de la información  
para servir de la salud mediante Big Data.

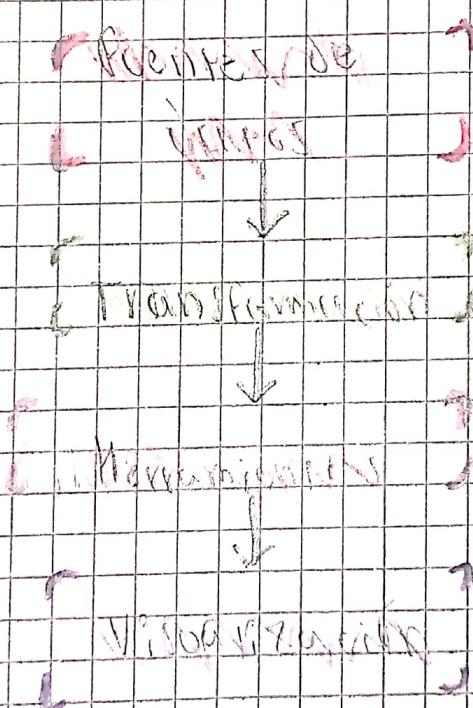
El trabajo tiene como objetivo desarrollar una  
arquitectura para consolidar la información en el  
ambito de los temas de salud dentro del Big  
Data. Se incluye un enfoque socio-económico  
y tipo quasi estandarizado (omnipresente, basado  
en la red de redes) y la ejecución  
de las funciones de los diferentes niveles.  
Estas funciones se refieren a los servicios  
de salud.

#### Reflexión

La implementación de Big Data en el sector de  
los servicios se está realizando de manera  
significativa en la forma en que se generan  
y manejan los datos. La capacidad de consolidar  
información proveniente de diversos fuentes  
permite obtener una visión más completa y  
accesible de los datos y su comportamiento  
de los usuarios.

Esto, a su vez, optimiza la forma de desarrollo, mejorando la eficiencia.

Dato; B. Al querer una consolidación de la información para servicios de la salud mediante Big Data.



Implementación de una arquitectura de software  
Sistema de dominios

El diseño de sistema basado en los patrones  
Sólido y Sistémicos permite heredamiento  
y técnicas que simplifican la creación  
de negocio, reutilizando el código ya  
existente. El diseño de dominio (DD) es  
el más usado en frameworks y aplicaciones  
que se basan en la separación de responsabilidades.  
Este diseño consiste en dividir el dominio  
en componentes independientes que tienen  
que hacer operaciones de acuerdo a su contexto  
con los entornos donde se implementa.

## Referencias

El diseño de sistema diseño por el  
dominio (DDD) nos permite la extensión sobre  
la implementación de objetos a Archivos con  
las necesidades reales del negocio. Este  
diseño no solo permite desarrollar soluciones  
más efectivas sino que también facilita  
una colaboración profunda entre expertos  
del dominio y desarrolladores, creando un lenguaje

Común que ralentiza la comunicación y el entendimiento mutuo.

Camberi, M., Gómez, V., & García-Pérez, M. (2020). Implementación de una actividad de creación gráfica sobre el dominio. In M. García-Pérez & M. Camperi (Eds.), *Introducción al pensamiento visual* (ASSL 2020) - Dpto. de Arquitectura Visual).



MDA y el rol de los modelos en el diseño  
de desarrollo de software.

Los modelos desempeñan un papel crucial en  
el desarrollo de software al formando la  
base de elementos y técnicas de trabajo  
de los distintos roles involucrados. La  
Arquitectura Unidad de Modelos (UML) introduce  
un enfoque basado en la creación y  
administración de modelos. Los enfoques  
se basan en las técnicas de modelación,  
automatización y ejecución para manejar  
función de MDA dentro de las formas más

### Reflexión

La metodología con la que se trabaja y la  
ejecución de los procesos de desarrollo  
se basa sobre todo en la funcionalidad y  
cómo se usa. Al principio todo lo que se hace  
desde la conceptualización del sistema hasta  
la implementación efectiva se basa en un  
enfoque silencioso y centrado, donde no  
se considera la idea de su propia implementación.

A) Permitir que los modelos evolucionen desde la conceptualización del problema hasta la implementación efectiva.

Quintero, J. B & Anaya N. (2007). MDA y el papel de los modelos en el proceso de desarrollo de software, Revista EIA, (2) 139-146.

Requisitos

↓

Análisis

↓

Diseno

↓

Implementación

↓

Pruebas

↓

Desarrollo

## ANÁLISIS DE REFERENCIAS LOGÍSTICAS PARA LA INDUSTRIA

Las organizaciones de Logística Común son un tipo de soluciones para reducir la complejidad y optimizar el uso del ancho de banda en aplicaciones del Internet de las cosas (IoT). Estas tecnologías son fundamentales en áreas como ciudades inteligentes, aplicaciones industriales 4.0 y consumo eficiente de energía. En este estudio revisa los principales resultados publicados por organizaciones como el Logística Común Consortium Project FAIR Log.

### Resumen

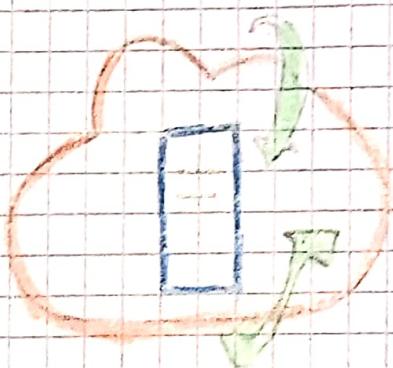
El IoT Común representa un avance significativo en la optimización de operaciones eficientes para obtener los beneficios del Internet de las cosas (IoT). Su complejidad reside en la integración de datos de proveedores y optimización de la red de banda ancha en tiempo real. La solución FAIR Log es una iniciativa que combina

la industria 4.0, las empresas

la oficina, los hogares y los  
comunidades urbanas

Sinton-Carmona L., Alcalde J. J., Muñoz L. J. (2019).  
(Coronavirus 2019-nCoV). Una revisión de literatura.  
Ugo Camilli (ara la industria 4.0: una visión).

Un momento de crisis (pp. 16-23).



Centro de

# Introducción de características de software en el ciclo de vida de los microtrabajos

Las Metáforas Áticas (MA) forman un sistema flexible y no obedece el criterio de la lógica formal o la lógica clásica, facilitando la asociación de las categorías y la recurrencia de los temas de la mitología. A pesar de algunas críticas retóricas dirigidas al final de la obra, se asume que estos chavacanos contienen el anhelo de una otra vida. La autoridad de Sustanciar (AS) se enfoca en tomar decisiones fundamentales y establecer

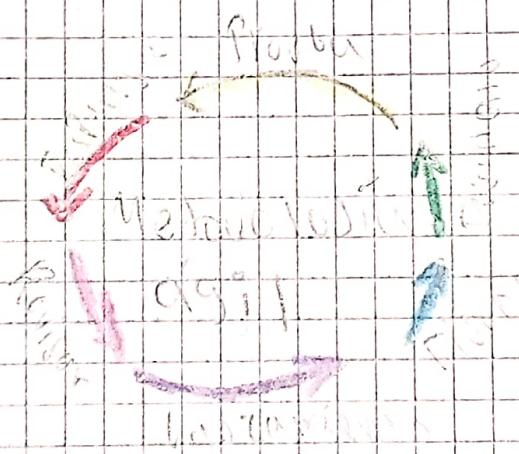
~~10.00~~

6) Los abusos y malos tratos en el  
entorno de los niños y niñas de la infancia  
y adolescencia. Los abusos y malos tratos  
en la infancia son considerados como los  
más graves, ya que tienen implicaciones  
inmediatas y a largo plazo. Estos son definidos  
como acciones que causan daño físico, psicológico  
o emocional a los menores. Los abusos y malos  
tratos en la infancia suelen ser cometidos por  
los padres, tíos, hermanos, amigos, maestros, etc.

en el menor los anteriores. Al incisivo anterior  
sobresale en suelen varios compuestos o  
laminillas.

Norberto M. E. Morales M. T. Montero J. Gaviria.

Rev. de Entomología C. (2013, September). Integración  
de Aracnidae (Araneoidea) en el ciclo de vida  
de las metacaracterísticas ágiles. (VNAIC 2013, Potosí,  
Bolivia).



## Arquitectura de Software para Sistemas de Tiempo Real Funcionados

Los sistemas de tiempo real críticos para misiones aeroespaciales enfrentan requisitos de seguridad y fiabilidad debido a las condiciones extremas de su entorno. Esos requisitos afectan tanto al hardware como al software, velando su evolución.

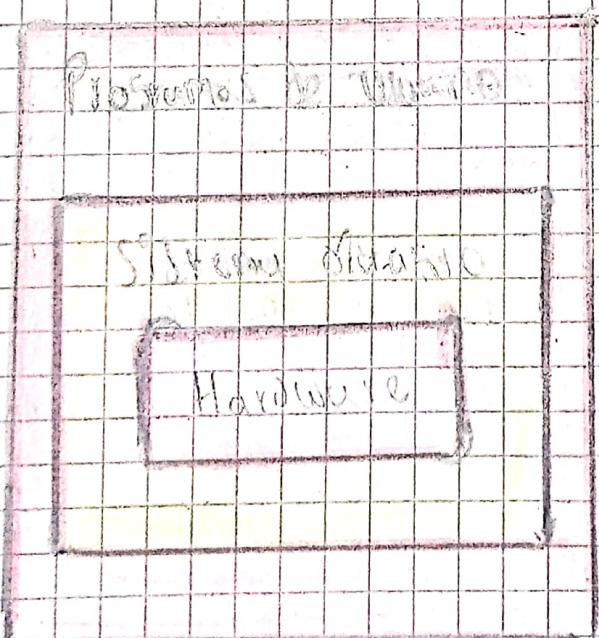
Tradicionalmente, los sistemas disponían entre componentes específicos de la misión y los de conservación procesos dedicados para cada una de sus funciones.

### Reflexión

La reflexión me ha enseñado que es más eficiente y seguro usar componentes de diseño ya existentes en misiones con restricciones de tiempo. Además, necesito un equipo capaz de innovar. La innovación debe combinar las ideas de gente con la habilidad.

- La evolución de los procesadores más potentes.
- Mantener una alta tasa de velocidad (long. oficinas).
- Reducción de los costos.

Favon, J.L. (2001) Arquitectura de software para sistemas de tiempo real (soluciones para la elaboración de sistemas de control).



Patrones de diseño para el modelo de redes  
en sistemas de información geográfica

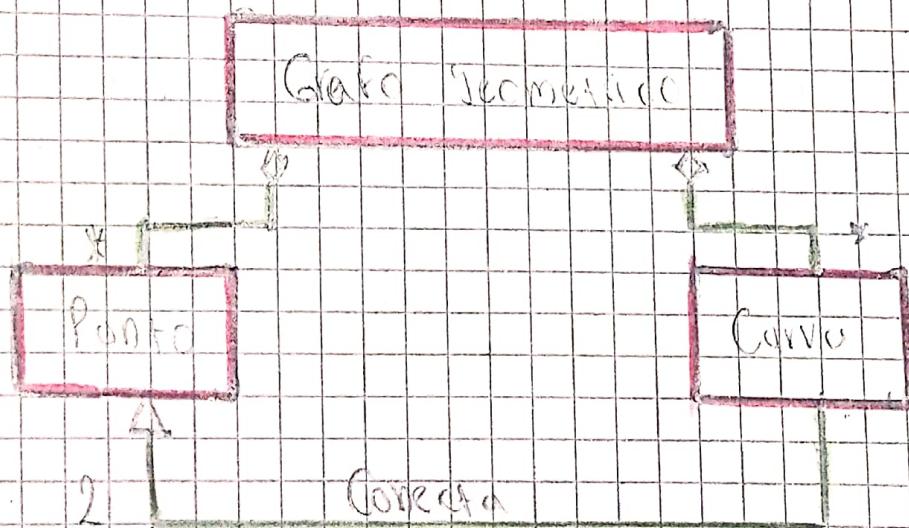
Este artículo presenta los patrones de  
diseño que se aplican al desarrollo de sistemas  
geográficos (SIG) orientados a objetos. Un sistema  
Objeto Usual tiene las características  
estructurales y dinámicas comunes de los  
entidades diferentes en distintos SIG, siendo  
un diseño unificado y consistente. Por otro  
lado, el patrón Grado Usual se enfoca  
en la representación, manipulación y visualización de  
los servicios y datos dentro de una solución integrada  
para ese propósito.

## Resumen

Los patrones de diseño presentados en el artículo  
reflejan la necesidad de tener en cuenta la  
la estructuración en el desarrollo de los sistemas  
compartidos como los SIG. El patrón Objeto  
Usual describe cómo estos sistemas comparten  
los comunes de los diferentes servicios.  
Para simplificar la estructura de un sistema,

Permitiendo una mayor flexibilidad en los resultados. Por otro lado el software gráfico es más intuitivo o fácil de usar. Sin embargo, también son más caros.

Montaña J.A., S. Ramos, J. (2002). Patrones de diseño para un modelo de redes en sistemas de información geográfica. Revista Colombiana de Computación, 1(1), 91-105.



Análisis de un parque de software en  
seguridad y la creación de un modelo.  
Este análisis incluye las etapas de desarrollo  
asociadas con la adopción de la arquitectura  
de microservicios, desarrollando su estructura,  
como las principales funcionalidades  
adicionales. Aunque los microservicios tienen  
diferencias como modularidad, distribución,  
implementación y funcionalidad, su  
rotación depende de la implementación de  
nuevos o cambios en la constante va-  
lencia de seguridad.

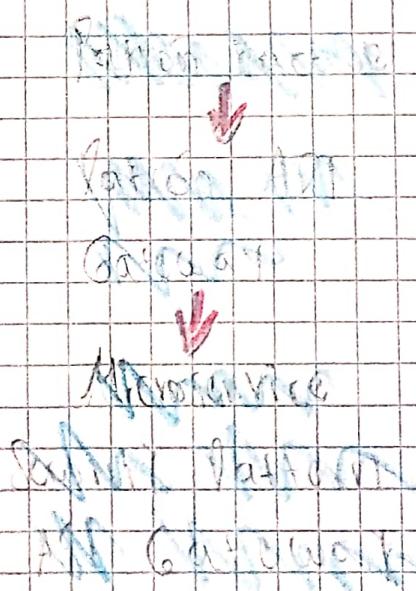
## Resumen

La solución de la arquitectura de microservicios  
proporciona una separación entre los servicios  
secundarios y los servicios de seguridad.  
Porque con su separación, la estabilidad y  
robustez se flexibiliza en el desarrollo  
de software, sin embargo, hay un límite de  
utilizar más servicios debido a la descentralización

- Esto refleja la necesidad de monitorear la seguridad desde el diseño, ya que un diseño malo fomenta la inseguridad entre los usuarios (el error).

Heyman, R.; Catman, K.M. (1993). Análisis de un fallo de software en el diseño de la aviónica de un microavión.

Esquema de evolución de errores



Arquitectura de integración del proceso de  
desarrollo de conocimiento con sistemas de  
gestión de bases de datos

(1) rápido crecimiento de los volúmenes de datos  
ha superado los métodos tradicionales de análisis,

como hojas de cálculo y consultas ad-hoc,

lo que ha surgido la necesidad urgente de

nuevas herramientas para transformar estos

datos en conocimiento útil. Esta tendencia emerse

se conoce como "Desublimación de Conocimientos

en Bases de Datos (DCBD), que se refiere al

trabajo de formar las bases de datos para su uso

en un todo de bases continuas de datos

## Resumen

El desarrollo de bases de datos

(DCBD) reflejan un creciente interés en la

gestión de grandes volúmenes de datos, un resultado

cada vez más presente en el mundo actual. A

medida que la cantidad de información crece

exponencialmente es fundamental contar con

herramientas que no solo manipulen esos datos,

sino que sean capaces de organizarlos y extraer

Patrones simples de madera ejecutados en  
madera, que involucra tanto la ejecución de datos  
como la interpretación de los mismos.



Código limpio: cómo aprenderlo a través de un ejercicio práctico

Este artículo explora la filosofía del código

limpio y cómo se puede enseñar  
a través de ejercicios prácticos. Se

Sobre la importancia de estos: Cómo  
seas clavo, fácil de mantener y

eficientes y como estos principios siguen

implimentarse mediante ejercicios prácticos.  
(los ejemplos expusieron varias formas de

malas prácticas y cómo corregirlos y combinarlo  
en el desarrollo de los mejores resultados de  
programación.

## Algunas

La idea de escribir código bien organizado

es algo que estoy programando,

desarrollar e implementar estos principios de  
manera práctica es una metáfora

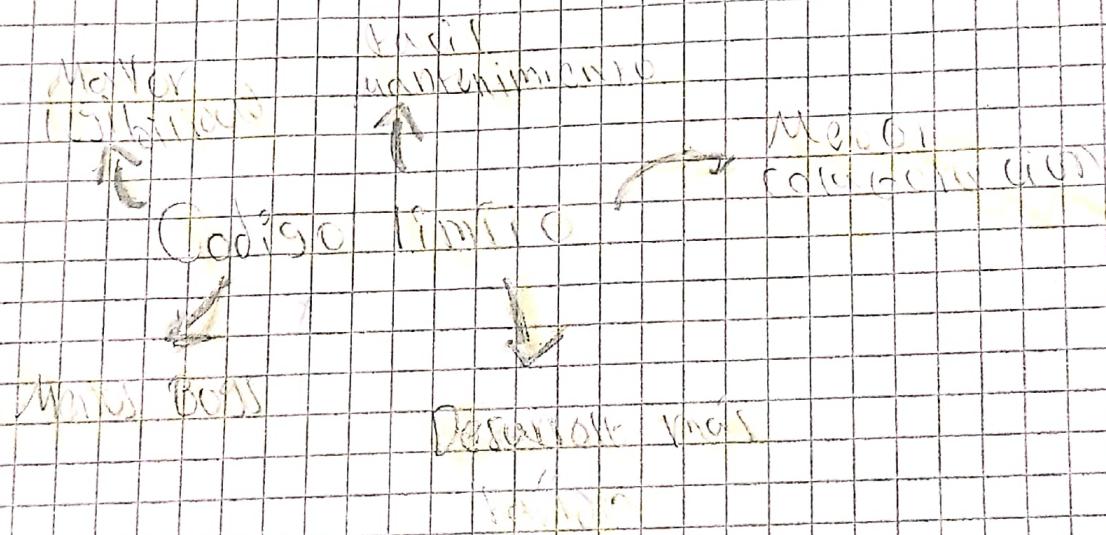
de posa. Al leer este artículo me di

cuenta de lo fundamental que es incorporar

buenos hábitos desde el principio del

Desarrollo. Adoptar un enfoque simple / sencillo no solo reduce los errores / fallas al implementarlo sino que también mejorará la comprensión del trabajo.

Felix Lusunus J.A. (2023) Asumirse a sí mismo como simple a nivel de un ejercicio físico.



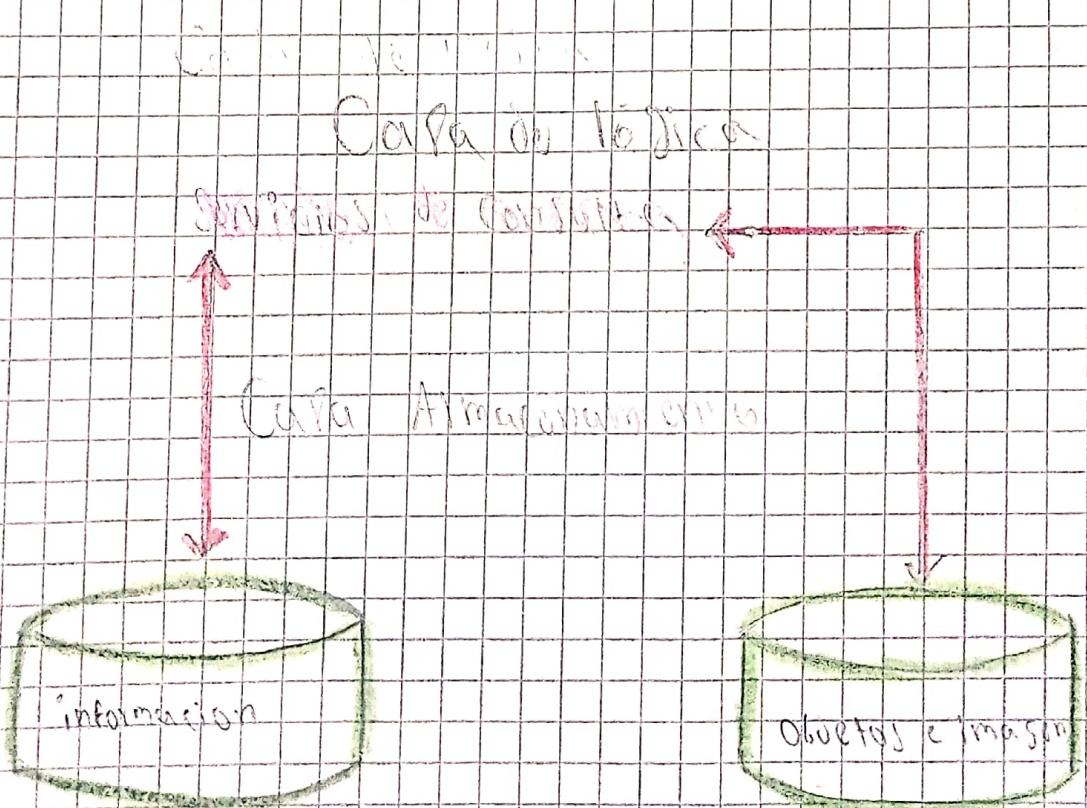
Propuesta de una administración turística basada en servicios turísticos para el desarrollo de ofertas de turismo cultural.

El turismo cultural es uno de los más relevantes y sostenibles del turismo, ya que incluye el patrimonio histórico, artístico y cultural de las naciones. En este contexto, el desarrollo propone una administración de servicios turísticos en servicios turísticos para desarrollar ofertas turísticas de turismo cultural. Estas administraciones utilizan diversas herramientas y tecnologías en su infraestructura tecnológica facilitando la creación de distintas vivencias.

## Reflexión

La propuesta de una administración de servicios turísticos basada en servicios turísticos para el desarrollo cultural. Una de las principales en este campo es permitir un acceso más amplio y flexible a la información y a las dinámicas bídicas. Esto es fundamental para el desarrollo cultural.

La creación de museos y yacimientos virtuales  
no solo enriquecen la experiencia del usuario,  
sino fue también contribuye a la conservación  
de los bienes culturales al reducir el desgaste físico  
de las instalaciones.



Patrón de diseño Beacon Action manager solo  
comunicar aplicaciones móviles (10)

El impacto de las cosas (10) suministra tanto  
información como objetos están conectados de  
manera constante formando el procedimiento  
y selección de información en tiempo real.

Este avance ha impulsado la popularidad de los  
teléfonos inteligentes, convirtiendo a las aplicacio-  
nes móviles en herramientas clave para comprender  
el mundo de los usuarios. Con la ayuda de  
los servicios web en nuevos dispositivos y servicios,  
como los beacon, que transmiten datos, informa-  
ción y conexión.

Reflexión

El impacto de las cosas es efectuado de una  
manera se interactúa entre el mundo físico y el  
mundo digital interconectado y seleccionar de recibir  
la mejor información de manera personalizada.

Esta conexión consiste entre personas y objetos  
que tienen formas y dimensiones físicas.  
Y estos sistemas necesitan para los desempeños  
de aplicaciones móviles, quienes deben adaptarse

o a tecnologia emerge como IoT. Vamos  
comunicar-nos através.

