**RESÚMENES DE ARTÍCULOS**

**CATALINA COMETA FIERRO**

**SENA**

**ADSO-2901817**

**Jesús Ariel Gonzales Bonilla**

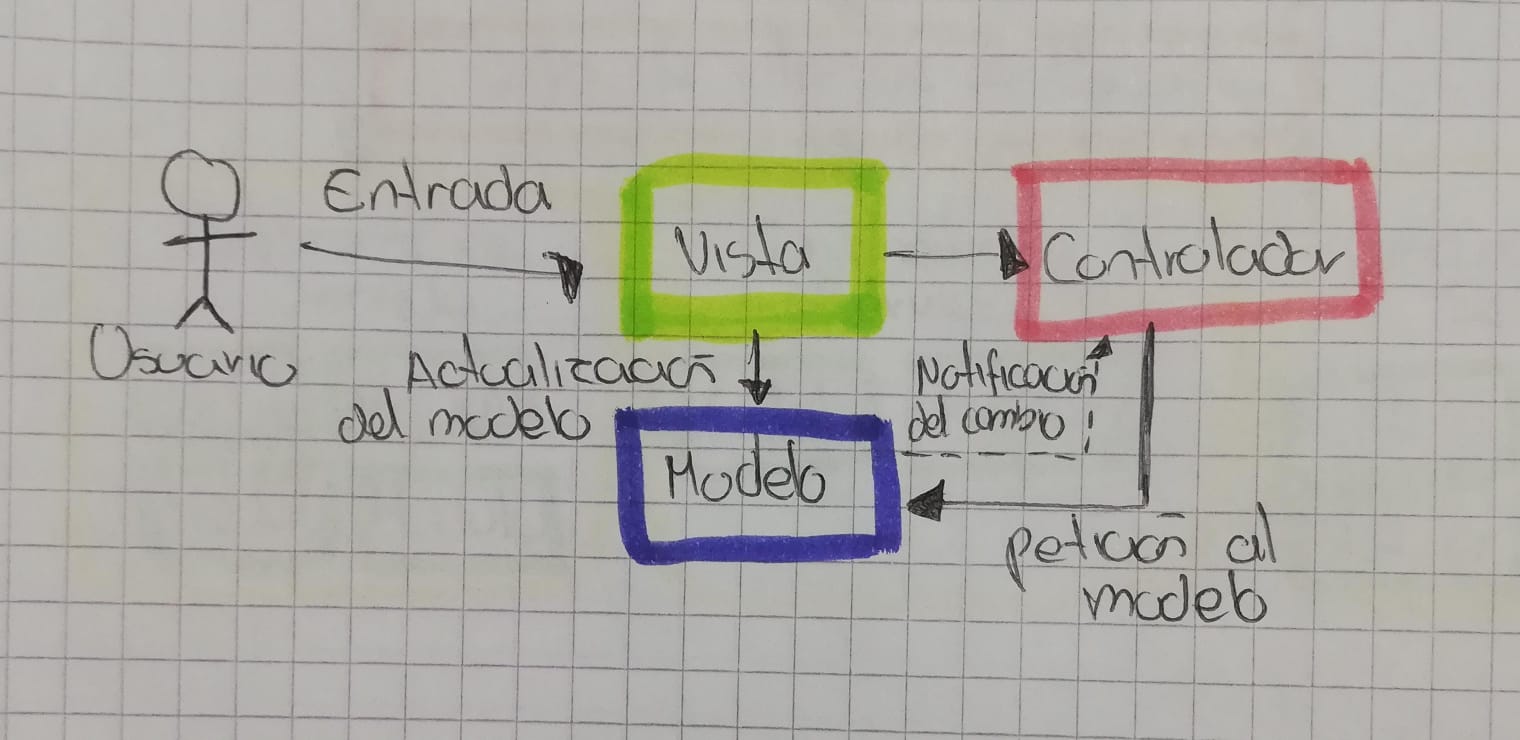
**AGOSTO-2025**

1. **Arquitectura de software**

La arquitectura de software, ha emergido como una disciplina de gran importancia dentro de la ingeniería de software. Una arquitectura adecuada es pieza clave para lograr tanto los requerimientos funcionales como no funcionales de un sistema. Por otro lado, una arquitectura no adecuada puede ser catastrófica.

La arquitectura también juega un papel importante en otros aspectos del desarrollo de software:

* Mejora la comprensión de sistemas grandes y complejos.
* Permite una mejor comunicación entre los diferentes interesados (stakeholders) en el sistema.
* Mejora las posibilidades de reuso.
* Proporciona planos para la construcción.
* Toma en cuenta la posible evolución del sistema.



**Reflexión**

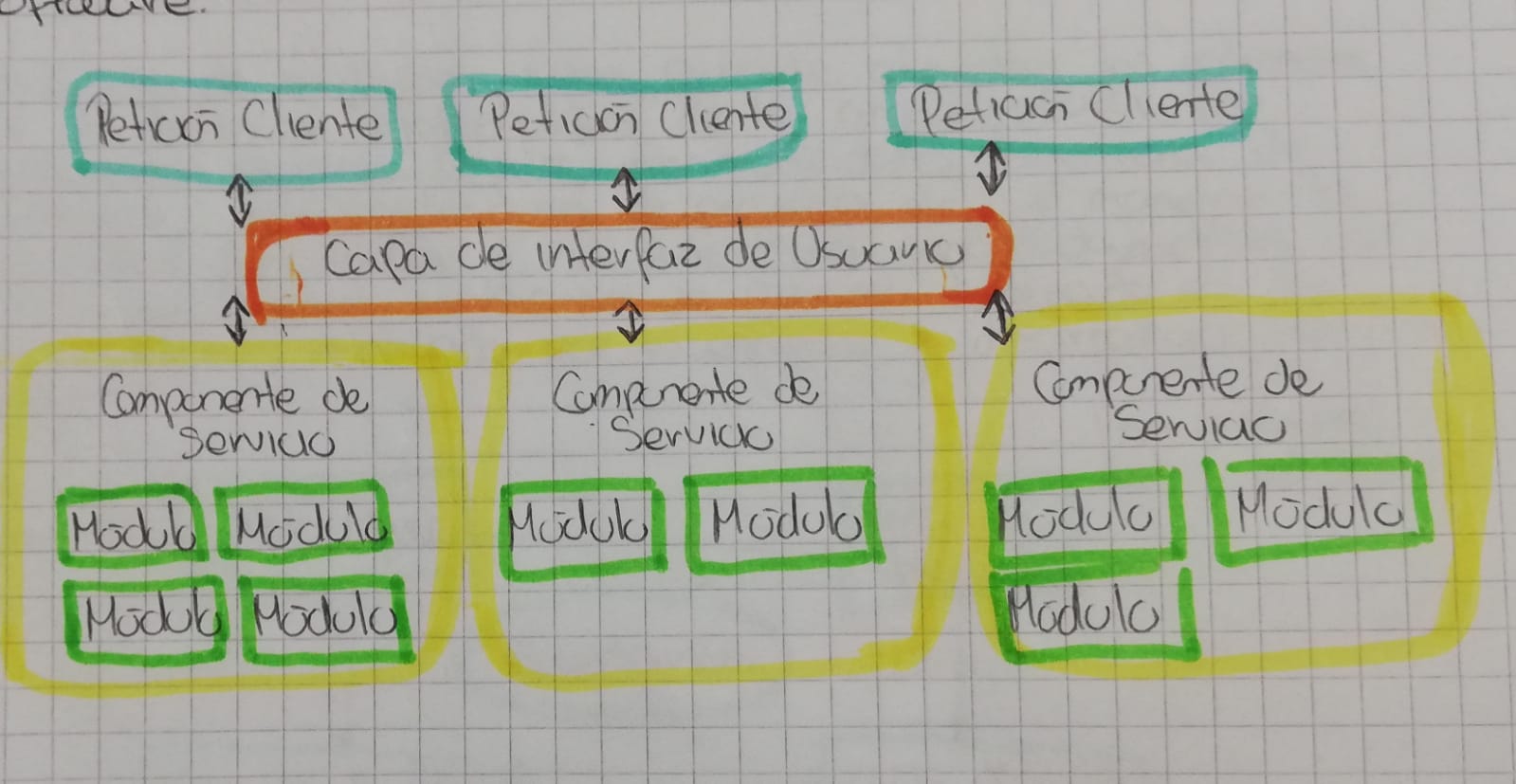
Este artículo me hizo reflexionar sobre la importancia de la arquitectura en el desarrollo de software. Me parece muy relevante que una buena arquitectura mejore la comprensión del sistema y la comunicación con los programadores, ya que permite mantener un orden que facilita que todos entiendan mejor cómo funciona el proyecto y cómo se relacionan sus componentes.

**Bibliografía**

* Fernández, L. F. (2006). Arquitectura de software. *Software Guru*, *2*(3), 40-45.

1. **Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web**

Según el artículo de desarrollo de software que realiza la Coordinación General de Tecnologías de la Información y Comunicación (CGTIC) de la Asamblea Nacional del Ecuador utiliza un enfoque tradicional o monolítico para el desarrollo de software. Este tipo de arquitectura agrupa toda la funcionalidad en una sola unidad ejecutable, lo que genera problemas en mantenimiento, escalabilidad y entregas. El estudio busca identificar las tecnologías, metodologías y arquitecturas usadas por la CGTIC para el desarrollo de aplicaciones web, y explorar opciones para implementar microservicios. Se utilizó un enfoque cualitativo con investigación descriptiva, aplicando grupos focales y revisiones bibliográficas. Se ha avanzado en el análisis del estado actual de los microservicios y en la identificación de necesidades para el desarrollo de aplicaciones web, con miras a satisfacerlas a través de una nueva arquitectura de software.



**Reflexión**

Este artículo me hizo reflexionar sobre la arquitectura utilizada por la CGTIC de la Asamblea Nacional del Ecuador. Actualmente, sus sistemas tienen un enfoque monolítico, lo que genera dificultades en el mantenimiento y su evolución. Me parece muy interesante que adopten una arquitectura basada en microservicios para mejorar la calidad de los sistemas, facilitando su mantenimiento y mejora.

**Bibliografía**

* López, D., & Maya, E. (2017). Arquitectura de software basada en microservicios para desarrollo de aplicaciones web.

1. **Planos Arquitectónicos: El Modelo de “4+1” Vistas de la Arquitectura del Software∗**

El modelo de 4+1 vistas fue desarrollado para remediar este problema. El modelo 4+1 describe la arquitectura del software usando cinco vistas concurrentes. Cada vista se

refiere a un conjunto de intereses de diferentes stakeholders del sistema.

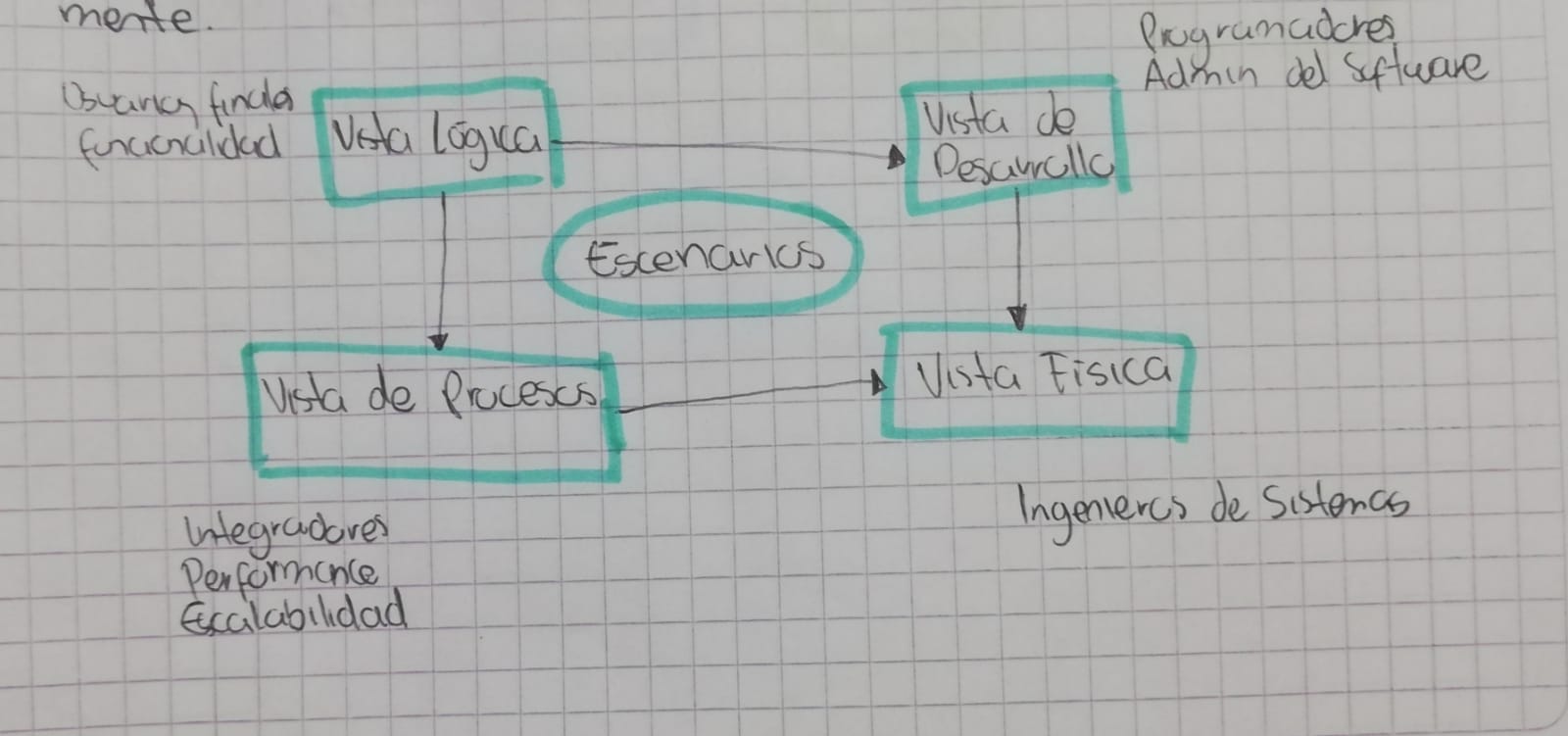
• La vista lógica describe el modelo de objetos del diseño cuando se usa un método de diseño orientado a objetos. Para diseñar una aplicación muy orientada a los datos, se puede usar un enfoque alternativo para desarrollar algún otro tipo de vista lógica.

• La vista de procesos describe los aspectos de concurrencia y sincronización del diseño.

• La vista física describe el mapeo del software en el hardware y refleja los aspectos de distribución.

• La vista de desarrollo describe la organización estática del software en su ambiente de desarrollo

Este uso de múltiples vistas permite abordar los intereses de los distintos “stakeholders” de la arquitectura por separado: usuarios finales, desarrolladores, ingenieros de sistemas, administradores de proyecto, etc., y manejar los requisitos funcionales y no funcionales separadamente. Las vistas se diseñan mediante un proceso centrado en la arquitectura, motivado por escenarios y desarrollado iterativamente.



**Reflexión**

Al leer este artículo me hizo reflexionar sobre el modelo 4+1 vistas en arquitectura de software. Permite describir un sistema desde cinco perspectivas complementarias, cada una dirigida a distintos stakeholders (usuarios, desarrolladores, ingenieros, gerentes), lo que facilita el diseño y la comunicación. Cada vista cumple un propósito, la lógica (diseño orientado a objetos y los requisitos funcionales), la de procesos (defines tareas y estilos), la de desarrollo (organiza el software en capas, facilitando el mantenimiento), la física (distribución de hardware) y la de escenarios (conecta todas las vistas, valida, ilustra y evoluciona). Este enfoque permite mitigar riesgos y construir sistemas de forma más organizada, abordando tanto requisitos funcionales como no funcionales.

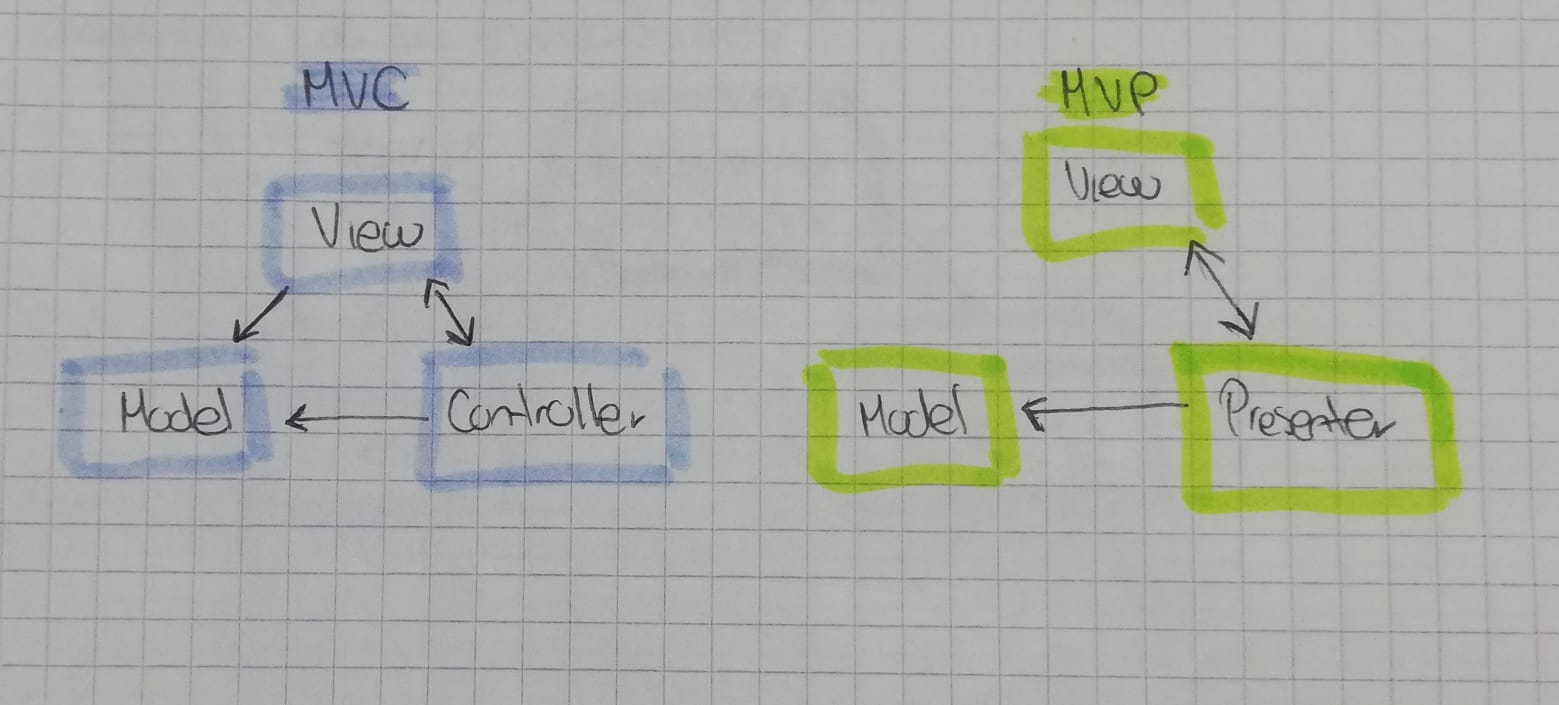
**Bibliografía**

* Kruchten, P. (1995). Planos Arquitectónicos: El Modelo de 4+ 1 Vistas de la Arquitectura del Software. *IEEE software*, *12*(6), 42-50.

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

Este artículo muestra el estado del arte en un área de la arquitectura de software llamada “Lenguajes de Patrones”, desde sus orígenes, los avances actuales y sus aplicaciones en la construcción de arquitecturas de software en diferentes dominios de aplicación. Este último aspecto es relevante ya que como se verá en este artículo, la extensibilidad y aplicabilidad de los lenguajes de patrones a diferentes dominios se convierte en una herramienta importante para diseñadores y desarrolladores de diversos tipos de sistemas de información.

Explora la evolución de los Lenguajes de patrones, su estructura y de cómo pueden aplicarse en distintos proyectos, ayudan a poder diseñar Arquitecturas de Software sólidas, organizadas y duraderas.



**Reflexión**

Este estudio me enseñó la importancia de los **lenguajes de patrones en arquitectura de software**, ya que no solo resuelven problemas complejos, sino que también aportan **calidad, eficiencia y adaptabilidad** en el desarrollo de sistemas. Considero necesario profundizar en estos lenguajes, pues ayudan a enfrentar mejor los retos actuales, diseñar nuevas soluciones y mantener un desarrollo más **organizado y efectivo.**

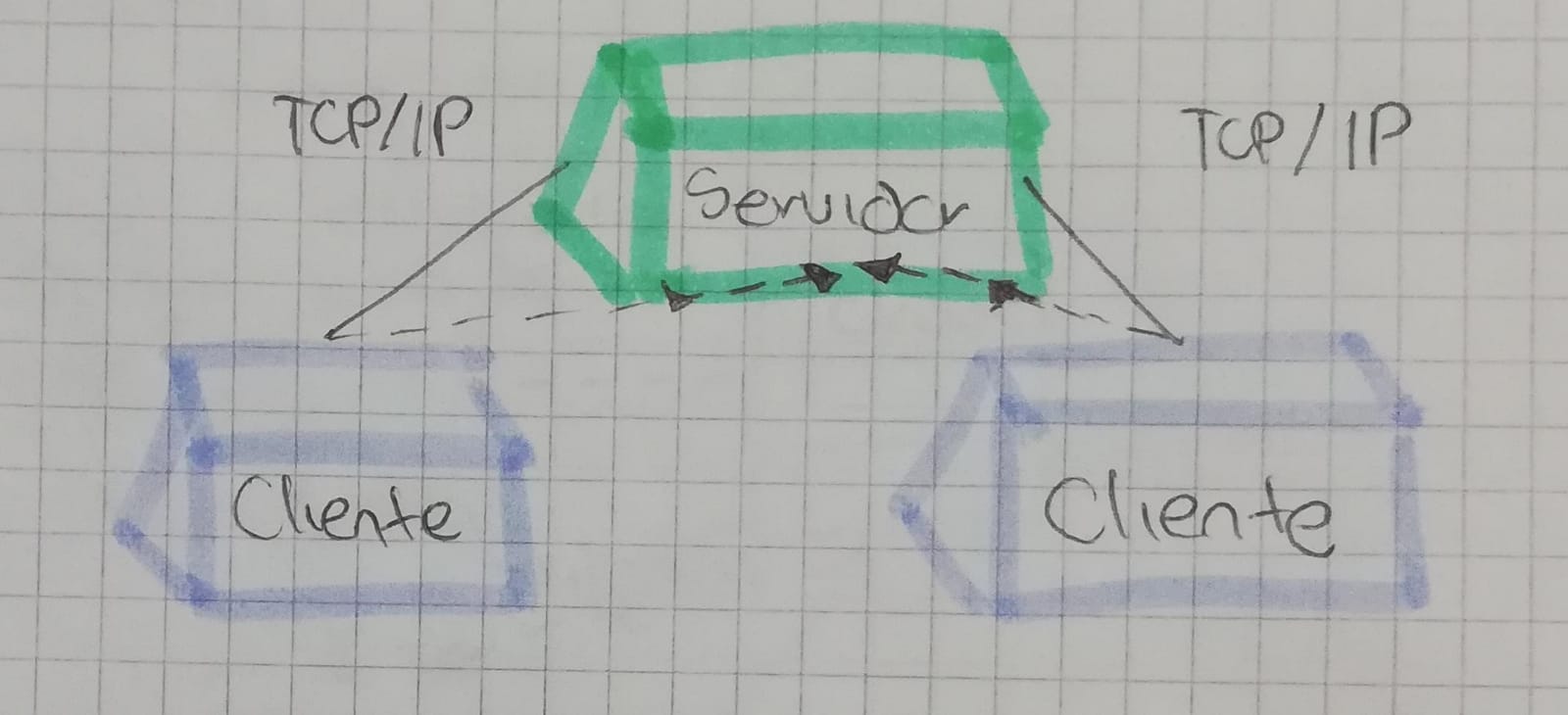
**Bibliografía**

* Jimenez-Torres, V. H., Tello-Borja, W., & Rios-Patiño, J. I. (2014). Lenguajes de patrones de arquitectura de software: una aproximación al estado del arte. *Scientia et technica*, *19*(4), 371-376.

1. **Introducción a la Arquitectura de Software**

En los primeros años de la construcción de software no existía el diseño del sistema como una etapa independiente de la programación. Pensar la descomposición del programa antes de programar se comienza a proponer, investigar y aplicar a principios de la década del 70 cuando se comienza a distinguir entre pequeños y grandes programas o sistemas.

Recién a principios de la década del 90 algunos investigadores, principalmente ligados a la Carnegie-Mellon University y al Software Engineering Institute, comienzan a ver la necesidad de investigar y desarrollar un nivel de abstracción superior al del diseño, al que llamaron Arquitectura de Software. Este trabajo se condensa principalmente en el libro pionero de MaryShaw y David Garlan y en el volumen más elaborado de Len Bass, Paul Clements, Rick Kazman. Precisamente la segunda edición de este último junto a otro volumen más especializado producido también en el SEI y al trabajo de un grupo de ingenieros de software de Siemens son tal vez las expresiones más modernas y acabadas de la especialidad.



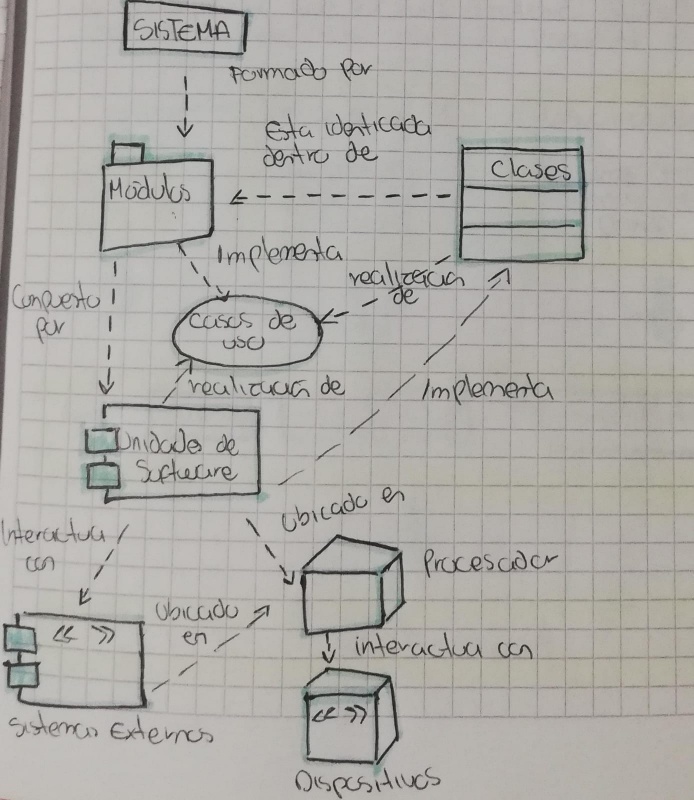
**Reflexión**

Este articulo me hizo reflexionar sobre la importancia de la arquitectura de software en el ciclo de vida de los sistemas. Comprendí que atender los requerimientos no funcionales es fundamental para evitar costos y rediseño. Considero que este enfoque aporta una visión más organizada, ya que la arquitectura de software no solo mejora la comunicación, también facilita el diseño de soluciones y brinda herramientas para enfrentar retos actuales.

**Bibliografía**

* Reynoso, C. B. (2004). Introducción a la Arquitectura de Software. *Universidad de Buenos Aires*, *33*, 11.

1. **Representación de la arquitectura de software usando UML**

El articulo propone una manera de representar la arquitectura de software usando UML, define diferentes tipos de diagramas UML, como el de casos de uso, clases, módulos, etc… En los últimos años se han llevado a cabo una gran cantidad de investigaciones en el área de arquitectura de software, buscando principalmente una forma de representación de un sistema que supere la informalidad de las líneas y cajas pero que a la vez sirva de medio de comunicación con los diferentes interesados en el proyecto, es decir, que no sea demasiado complejo. El desarrollo de lenguajes de descripción de Arquitecturas da a los ingenieros de sistemas una nueva herramienta para la acertada representación de la arquitectura de un sistema; sin embargo, los lenguajes desarrollados actualmente por lo general son muy complejos o solo se adaptan a un tipo particular de sistemas. En este contexto, usar UML es una alternativa más accesible y flexible, permitiendo ver el sistema en varias perspectivas.

**Reflexión**

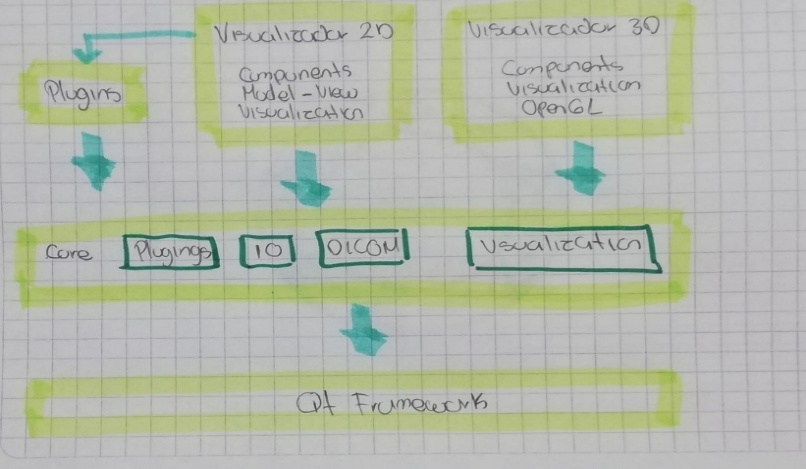
Este artículo me hace reflexionar sobre la importancia de los esquemas UML en la arquitectura de software, ya que permiten representar los sistemas de manera clara, organizada y estructurada, con el fin de facilitar la comunicación entre los distintos actores del desarrollo demostrando ser una herramienta viable, útil y adaptable. Además, resalta la necesidad de formalizar la documentación arquitectónica para optimizar la calidad del software y fortalecer la colaboración entre equipos.

**Bibliografía**

* Gil, S. V. H. (2003). Representación de la arquitectura de software usando UML. *Sistemas y Telemática*, *1*(1), 63-75.

1. **Arquitectura de software para el sistema de visualización médica Vismedic**

Una arquitectura de software basada en la integración de los estilos arquitectónicos: Arquitectura basada en componentes, para el sistema de visualización médica Vismedic, con el objetivo de reducir los problemas de extensibilidad, reusabilidad y dependencias que existían en la arquitectura anterior. Para realizar la propuesta se hizo necesario el estudio de los conceptos relacionados con la Arquitectura de Software, las características arquitectónicas de tres productos establecidos en el campo del procesamiento y visualización de imágenes. La arquitectura propuesta integra las principales características de las bibliotecas antes mencionadas e incorpora el empleo de plugins para extender las funcionalidades. La misma se validó a través de la Técnica de evaluación basada en prototipos y de la aplicación del Método de Análisis de Acuerdos de Arquitectura de Software (ATAM). La evaluación permitió identificar los riesgos presentes en la propuesta realizada y determinar que la arquitectura satisface los atributos de calidad definidos para la presente investigación.



**Reflexión**

Este artículo refleja la importancia de diseñar software que sea más flexible, escalable y fácil de mantener. Al reducir el acoplamiento, no solo mejora la extensibilidad del sistema, también se facilita la innovación y la incorporación de nuevas funcionalidades. El caso del visualizador médico desarrollado en la UCI es un buen ejemplo, su arquitectura anterior, con fuerte acoplamiento entre componentes y baja reusabilidad, dificultaba la incorporación de nuevas funciones y retrasaba el desarrollo. Esto demuestra cómo una buena arquitectura puede marcar la diferencia entre un software limitado y uno con capacidad de evolucionar, adaptarse y responder a las necesidades cambiantes de los usuarios y profesionales que lo utilizan.

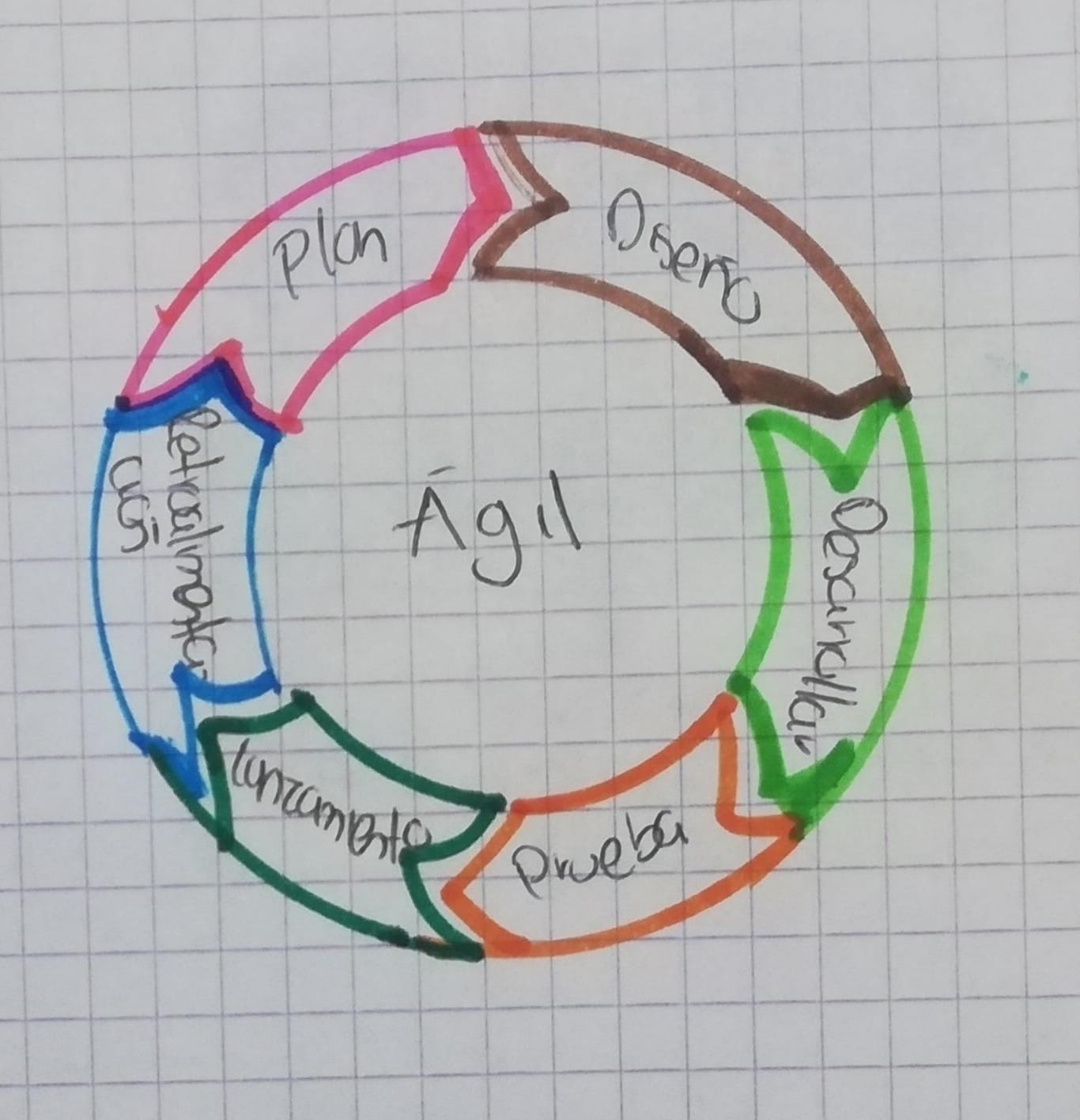
**Bibliografía**

* Rodríguez Peña, A. D., & Silva Rojas, L. G. (2016). Arquitectura de software para el sistema de visualización médica Vismedic. *Revista Cubana de Informática Médica*, *8*(1), 75-86.

1. **Integración de Arquitectura de Software en el Ciclo de Vida de las Metodologías Ágiles. Una Perspectiva Basada en Requisitos**

Las metodologías Ágiles se centran en el trabajo en equipo, la adaptabilidad y colaboración dentro del grupo de software y también entre los miembros del grupo y los usuarios finales. El uso de las Metodologías Ágiles, ha hecho que se use más al desarrollo de proyectos de software dado las necesidades cambiantes y la espera de beneficios en el menor tiempo posible. Pero también desde la perspectiva de los requisitos, esto hace que las Metodologías agiles típicamente eviten un trabajo inicial sustancial, suponiendo que los requisitos siempre cambian y continúan cambiando a lo largo del ciclo de vida del proyecto.

La Arquitectura de Software, es una manifestación de decisiones de etapas muy tempranas del diseño sobre un sistema. Este tratamiento con enfoques diferentes en las primeras etapas, ha sido uno de los factores que ha causado la sensación de que las Metodologías agiles y la Arquitectura de Software van en direcciones diferentes y no pueden coexistir juntas. Sin embargo, esta tendencia está cambiando, hasta el punto que ha surgido el concepto “Arquitectura Ágil”, se enfatiza fuertemente en el concepto de los “Requisitos Significantes para la Arquitectura”.



**Reflexión**

Este artículo me hizo reflexionar en la importancia de no ver la arquitectura y la agilidad como opuestas, sino como complementarias. La Arquitectura Ágil nos muestra que sí se puede construir software fuerte y bien diseñado sin perder la flexibilidad para adaptarse cuando las cosas cambian. Si desde el inicio se toman en cuenta los aspectos importantes de la arquitectura, el resultado no solo responde a lo que se necesita en el momento, sino que también asegura calidad, crecimiento y estabilidad con el paso del tiempo.

**Bibliografía**

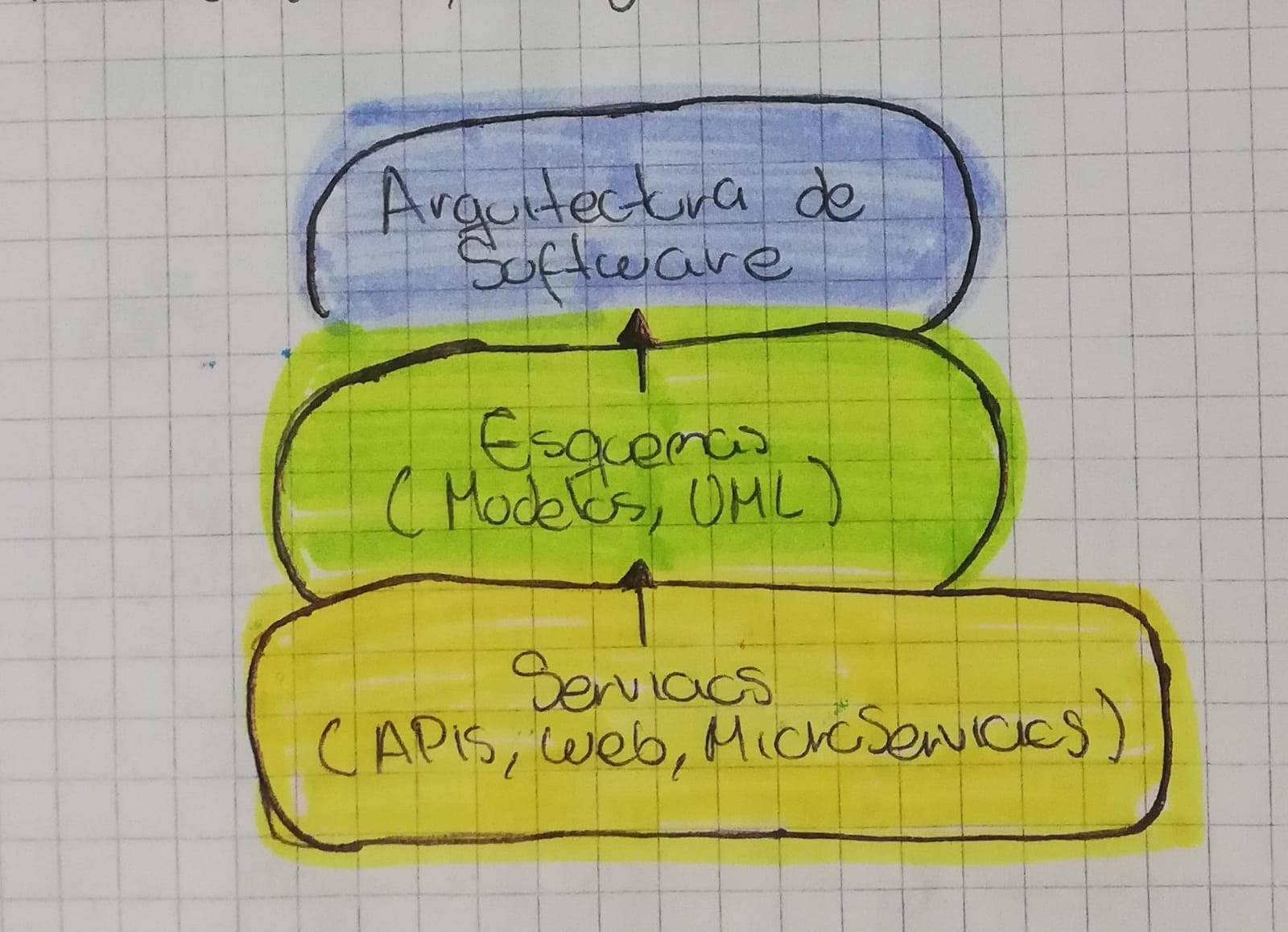
* Navarro, M. E., Moreno, M. P., Aranda, J., Parra, L., Rueda, J. R., & Pantano, J. C. (2017, September). Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles. In *XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires)*.

1. **ARQUITECTURA DE SOFTWARE, ESQUEMAS Y SERVICIOS**

El artículo se centra en la Arquitectura orientada a servicios (SOA), un enfoque en que las aplicaciones empresariales se construyen como un conjunto de servicios autónomos que se comunica hacia y desde el servicio, es realizada utilizando mensajes y no llamadas a métodos. Estos mensajes deben contener o referenciar toda la información necesaria para entenderlo.

La idea es que haya el mínimo posible de llamadas entre el cliente y el servicio. Un servicio es la evolución en complejidad de un componente distribuido, y se diferencian en:

* Mucho menos acoplados con sus aplicaciones cliente que los componentes.
* Menor granularidad que los componentes.
* Son controlados y administrados de manera independiente.
* Son transparentes de su localización en la red, de esta manera garantizan escalabilidad y tolerancia a fallos.
* Tienen sus propias políticas de escalabilidad, seguridad, tolerancia a fallos, manejo de excepciones, configuración.

****

**Reflexión**

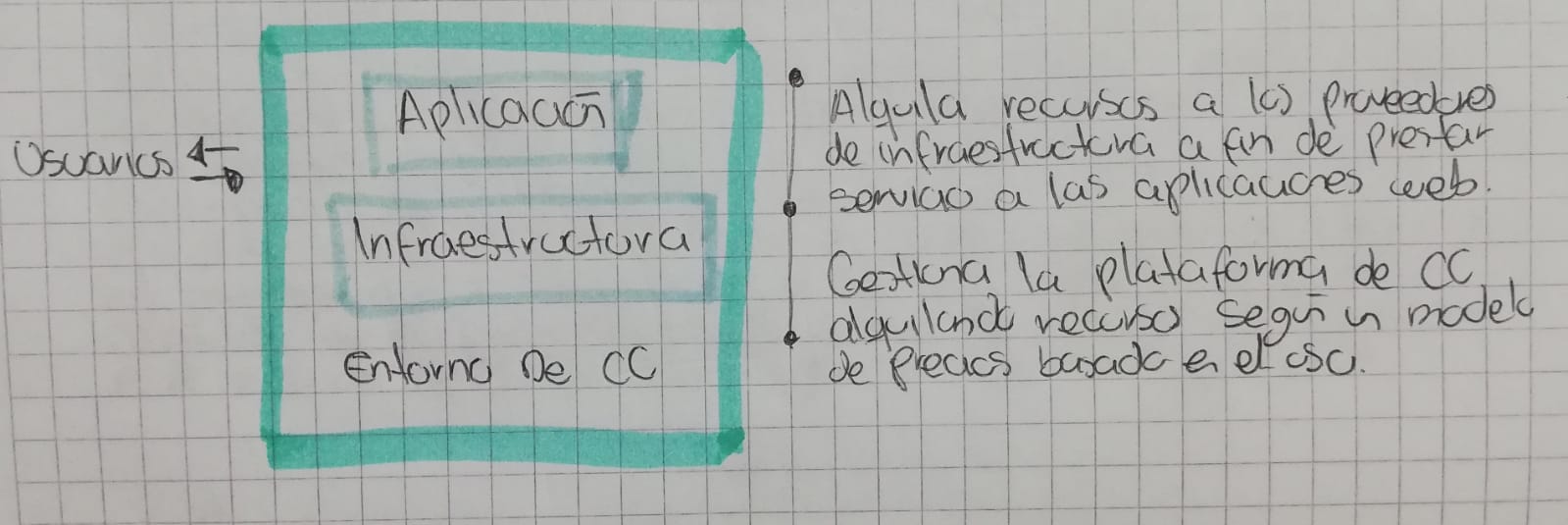
La arquitectura orientada a servicios (SOA) nos deja una reflexión de cómo el software también tiene que adaptarse a un mundo cada vez más conectado y en constante cambio. Antes, los sistemas eran grandes y rígidos, lo que hacía difícil modificarlos o crecer con ellos. Con SOA, se abre la puerta a un modelo más flexible, donde cada parte puede trabajar de manera independiente, pero al mismo tiempo unirse para lograr un objetivo común. Enseña que no se trata de crear software complicados, sino de diseñarlos de forma que puedan evolucionar, integrarse y seguir siendo útiles con el paso del tiempo.

**Bibliografía**

* Romero, P. Á. (2006). Arquitectura de software, esquemas y servicios. *Ingeniería Industrial*, *27*(1), 1

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

Este articulo presenta un entorno de diseño integral que permite formular diseños de arquitecturas de software destinadas a la representación de aplicaciones web. Este entorno abstrae los principales problemas identificados a nivel de diseño, planteando módulos que ayudan al arquitecto en la elaboración de diseños de calidad. Para esto, utiliza como base un metamodelo de componentes arquitectónicos que identifica un conjunto de elementos comúnmente utilizados en dichas arquitecturas. Sobre el modelo se construye una herramienta de instanciación gráfica que se complementa con la verificación de patrones de diseño a fin de garantizar su correcta aplicación.



**Reflexión**

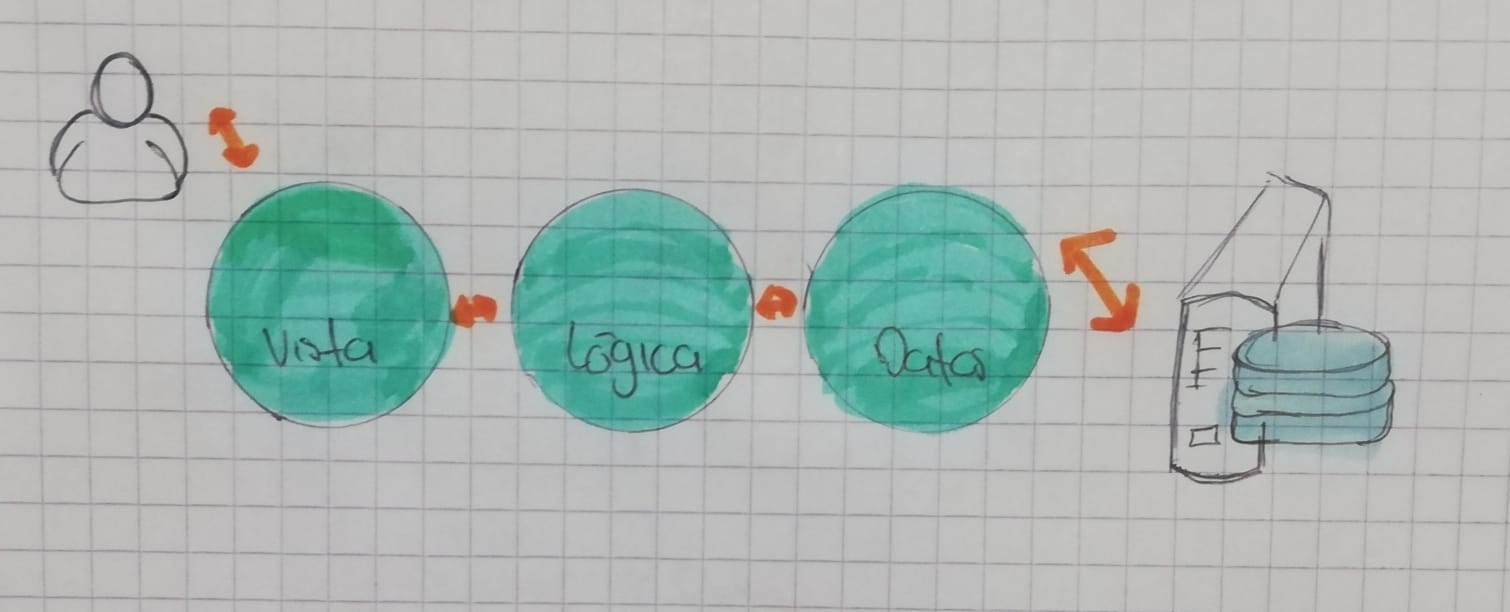
Esta idea de un ambiente integral para diseñar arquitecturas de software nos enseña que, en tecnología, no está la calidad solamente en escriHobir código, sino en pararse y pensar primero cómo queremos ensamblar lo que hacemos. Cuando usamos modelos claros, herramientas visuales y buenas prácticas, el trabajo no solo se vuelve más fácil para quienes desarrollan, sino que también permite que las aplicaciones sean más comprensibles, se mantengan mejor en el tiempo y puedan crecer sin perder su esencia. En realidad, es como hacer una casa, lo que importa no es colocar simplemente ladrillos, sino tener unos planos sólidos que aseguren que todo cuaje bien y que podamos ampliarla si la necesitamos.

**Bibliografía**

* Blas, M. J., Leone, H. P., & Gonnet, S. M. (2019). Modelado y Verificación de Patrones de Diseño de Arquitectura de Software para Entornos de Computación en la Nube.

1. **Arquitectura de software académica para la comprensión del desarrollo de software en capas**

El desarrollo de software implica considerar una cantidad variada de aspectos tecnológicos. Entre los más destacados podemos mencionar los relacionados con: el acceso a datos, las interfaces, los procesos funcionales, el control de las transacciones, la accesibilidad y la seguridad. Lograr un diseño coherente con los requerimientos planteados, niveles aceptables de flexibilidad, extensibilidad y usabilidad, así como facilitar las actividades de mantenimiento (preventivo, adaptativo, correctivo, evolutivo y perfectivo) lleva a pensar la concepción del software sustentado en una arquitectura. Una de las formas arquitectónicas más conocidas propone dividir al software en capas, donde cada una representa un agrupamiento lógico que se corresponde con un elemento físico del sistema deseado. Las capas planteadas en el diseño arquitectónico deben tener la suficiente independencia para tratarlas como unidades funcionales independientes y las interfaces de sus componentes permitirán que interactúen de manera significativa y eficiente entre sí. Con el diseño en capas se intenta lograr que los cambios que se realizan en el sistema impacten de forma acotada en el lugar afectado y que los efectos no deseados en el resto del software disminuyan a su mínima cantidad.



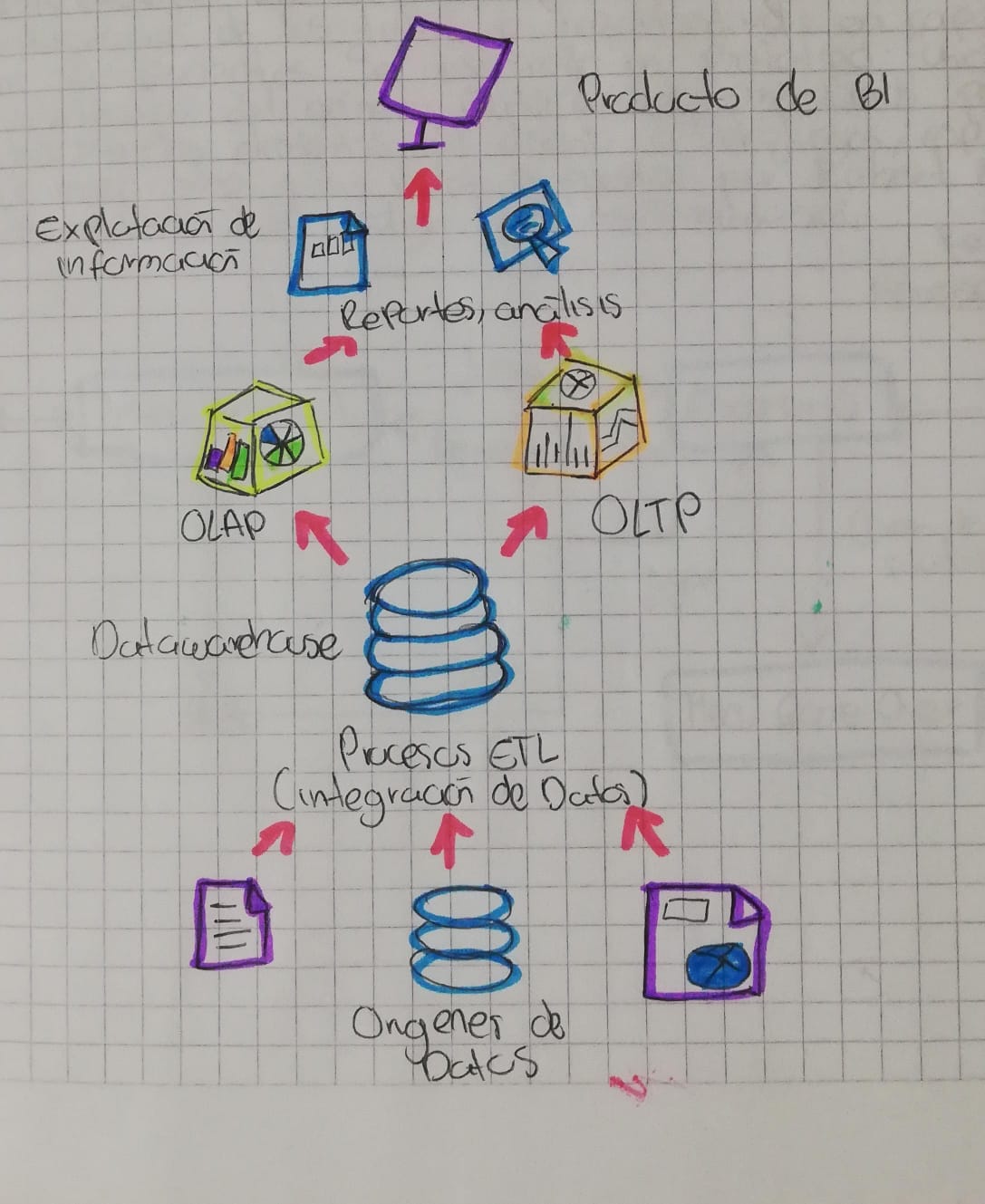
**Reflexión**

Este artículo me hizo reflexionar que la arquitectura en capas nos muestra que desarrollar software no es solo programar, sino también planear, organizar y dar forma a lo que queremos construir. Al dividir responsabilidades y aplicar buenas prácticas de diseño, no solo conseguimos aplicaciones más claras y fáciles de mantener, sino que también ayudamos a que los desarrolladores, desarrollen un pensamiento crítico, preparándolos para crear sistemas sólidos y duraderos en un mundo tecnológico que cambia constantemente.

**Bibliografía**

* Cardacci, D. G. (2015). *Arquitectura de software académica para la comprensión del desarrollo de software en capas* (No. 574). Serie Documentos de Trabajo.

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

Cuando se desconocen las herramientas y metodologías existentes para el uso de las organizaciones que ayudan en la toma de decisiones, se presentan problemas al valerse de información engañosa o no confiable. A la hora de superar estas limitaciones, la Inteligencia de Negocios se apoya en un conjunto de herramientas que facilitan la extracción, depuración y análisis de los datos generados en una organización que, sin importar su tamaño, se puede beneficiar con la implementación de una de estas herramientas como puede ser el Cuadro de Mando Integral (CMI), el cual contribuye a la revisión permanente de la estrategia y asiste a la empresa en el desarrollo de procesos de gestión decisivos.

**Reflexión**

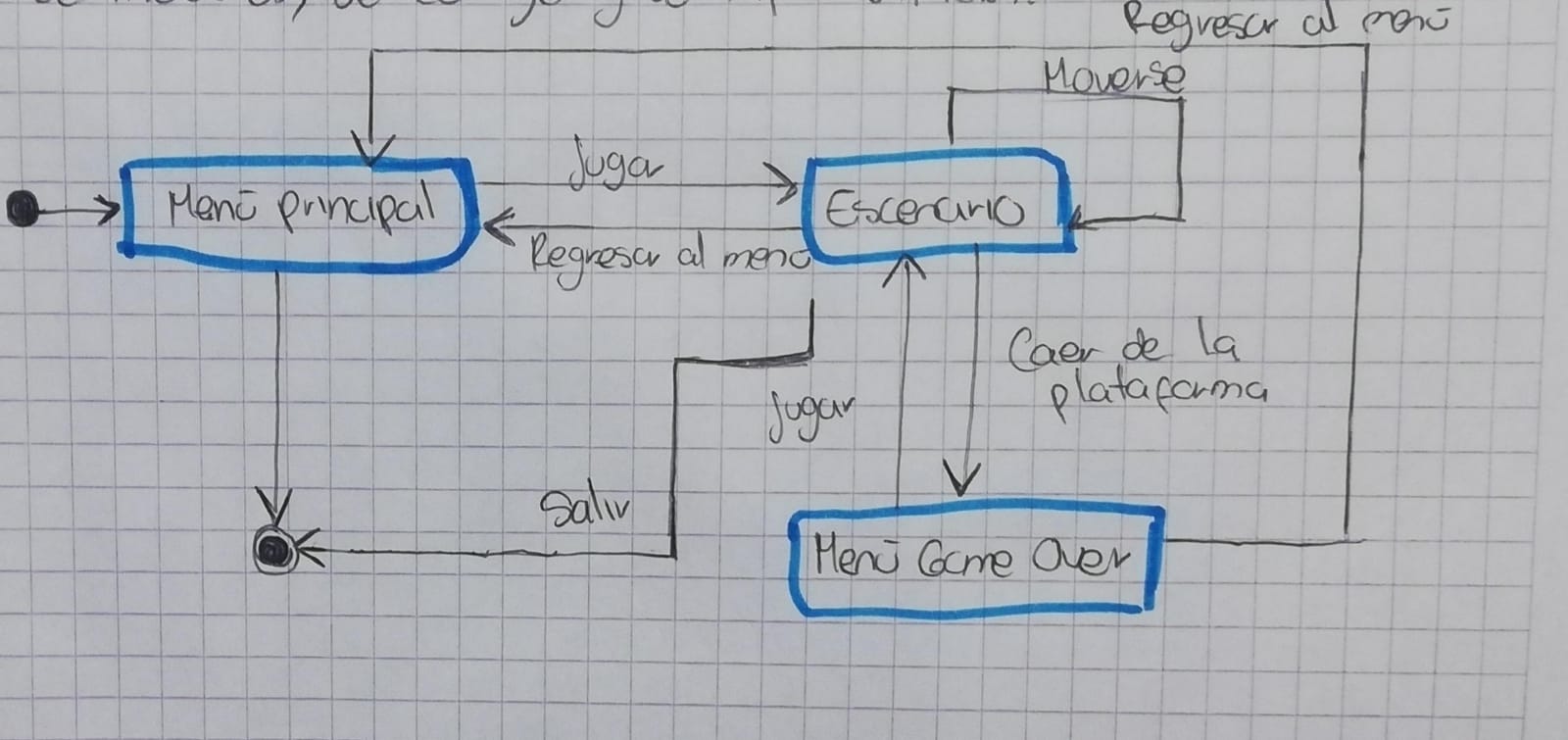
un sistema de Cuadro de Mando Integral muestra cómo la tecnología puede convertirse en un verdadero puente entre la estrategia y la acción dentro de una organización. Un buen diseño de arquitectura de software no se limita a lo técnico, sino que se convierte en un aliado estratégico, permite que la información se transforme en conocimiento y que ese conocimiento guíe decisiones más acertadas, fortaleciendo la competitividad y sostenibilidad de la organización.

**Bibliografía**

* Cabrera, G. A. H. (2017). Arquitectura de software para la construcción de un sistema de cuadro de mando integral como herramienta de inteligencia de negocios. *Tecnología Investigación y Academia*, *5*(2), 143-152.

1. **Arquitectura de software para el desarrollo de videojuegos sobre el motor de juego Unity 3D**

La arquitectura de software se ha consolidado como una disciplina que intenta contrarrestar los efectos negativos que pueden surgir durante el desarrollo de un producto informático, ocupando un rol significativo en la estrategia de negocio de una organización que basa sus operaciones en el software, volviéndose necesaria para todo tipo de desarrollo, incluyendo los videojuegos. La presente investigación tiene como objetivo desarrollar una arquitectura de software para videojuegos desarrollados sobre el motor de juego Unity 3D, que permita organizar y estructurar sus características funcionales básicas. A partir del estudio de arquitecturas usadas en videojuegos se agruparon las clases candidatas de la solución propuesta, identificándose los paquetes principales, dependencias entre ellos, patrones de diseño y buenas prácticas empleadas, concretando una arquitectura de software basada en la integración de los tipos de arquitectura: en capas y basada en componentes. Se desarrolla un prototipo funcional de un videojuego del género plataformas, empleando para describir los elementos del diseño de videojuegos, especificación de mecanismos y las vistas propuestas por Robert Nord: conceptual, de módulos, de código y de implementación.



**Reflexión**

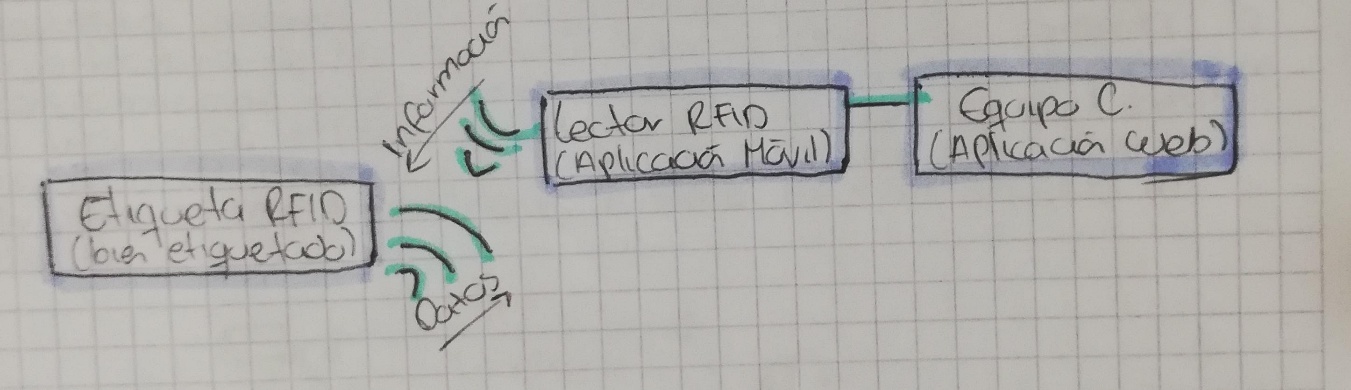
Este artículo muestra que no solo se trata de videojuegos de creatividad, sino también de emplear principios de ingeniería de software que aseguren orden, calidad y crecimiento. Utilizar una arquitectura organizada, basada en patrones de diseño y principios sólidos, estimula la colaboración, el aprendizaje y garantiza un desarrollo más eficiente y sostenible.

**Bibliografía**

* Paez, A. H., Falcón, J. A. D., & Cruz, A. A. P. (2018). Arquitectura de software para el desarrollo de videojuegos sobre el motor de juego Unity 3D. *I+ D Tecnológico*, *14*(1), 54-65

1. **Arquitectura de software de una aplicación móvil para desarrollar un sistema de identificación por radiofrecuencia**

Un sistema RFID (Radio Frequency Identification, dentificación por Radiofrecuencia) ayuda a identificar toda clase de objetos por medio de etiquetas o tags RFID, un sistema de este tipo tiene como funciones principales saber cuántos bienes se tienen y donde se encuentran ubicados, controlando estos elementos, el apoyo al inventario aumenta porque lo mantiene actualizado con información real. El presente trabajo propone una arquitectura de software para desarrollar una aplicación móvil que es parte de un sistema RFID, que sirve de apoyo al inventario del Instituto Tecnológico de Orizaba y así también se presentan los resultados (aplicaciones) que se generaron a partir de las arquitecturas propuestas.

****

**Reflexión**

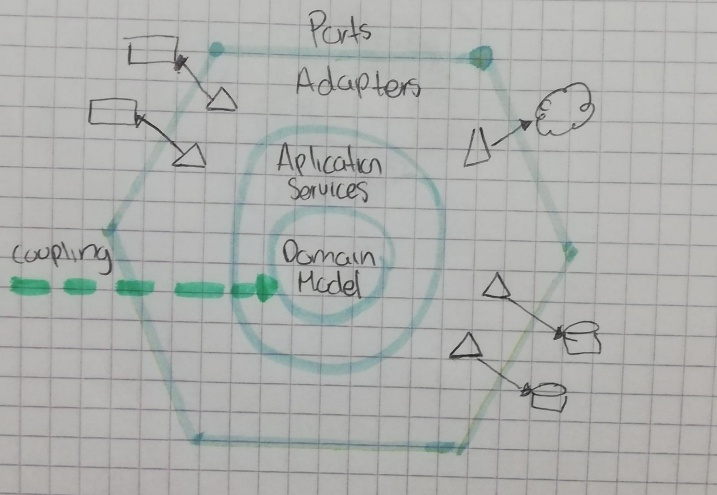
Este artículo muestra cómo la tecnología RFID, combinada con una arquitectura de software bien estructurada, permitiendo identificar objetos o personas en tiempo real. Además, demuestra que el orden arquitectónico en aplicaciones móviles garantiza eficiencia, escalabilidad y confiabilidad. Este tipo de enfoque permite evolucionar hacia procesos más inteligentes, seguros y adaptables.

**Bibliografía**

* García, G. L., Martínez, I. L., Camarena, G. S. P., Figueroa, M. A. A., & Zepahua, B. A. O. (2015). Arquitectura de software de una aplicación móvil para desarrollar un sistema de identificación por radiofrecuencia-Software architecture of a mobile application for developing a system identification by radiofrequency. *ReCIBE, Revista electrónica de Computación, Informática, Biomédica y Electrónica*, *4*(1), IV-IV.

1. **Implementación de una Arquitectura de Software guiada por el Dominio**

El diseño de software bajo un enfoque sólido, sistemático, y completo cuenta con un conjunto de herramientas y técnicas, que permiten separar la complejidad del negocio, dejando como pieza central del mismo, el dominio. El enfoque diseño dirigido por el dominio ofrece la posibilidad de contar con principios, patrones y actividades para construir un modelo de dominio, que es el artefacto principal. Además, ayuda a garantizar que la arquitectura de software permanezca centrada en las funcionalidades del negocio. En este trabajo proponemos un desarrollo de software y la aplicación de una arquitectura orientada al dominio. La contribución del mismo es mostrar la viabilidad sobre la adopción del enfoque, como valor estratégico, el cual proporciona mapear la idea del dominio del negocio para el desarrollo de los artefactos del software. El trabajo propuesto presenta la transformación y adaptación de una arquitectura de software de tres capas típicas a una arquitectura centrada en el dominio específico del negocio y la selección de tecnologías que permiten su implementación, se valida mediante un caso de estudio.

****

**Reflexión**

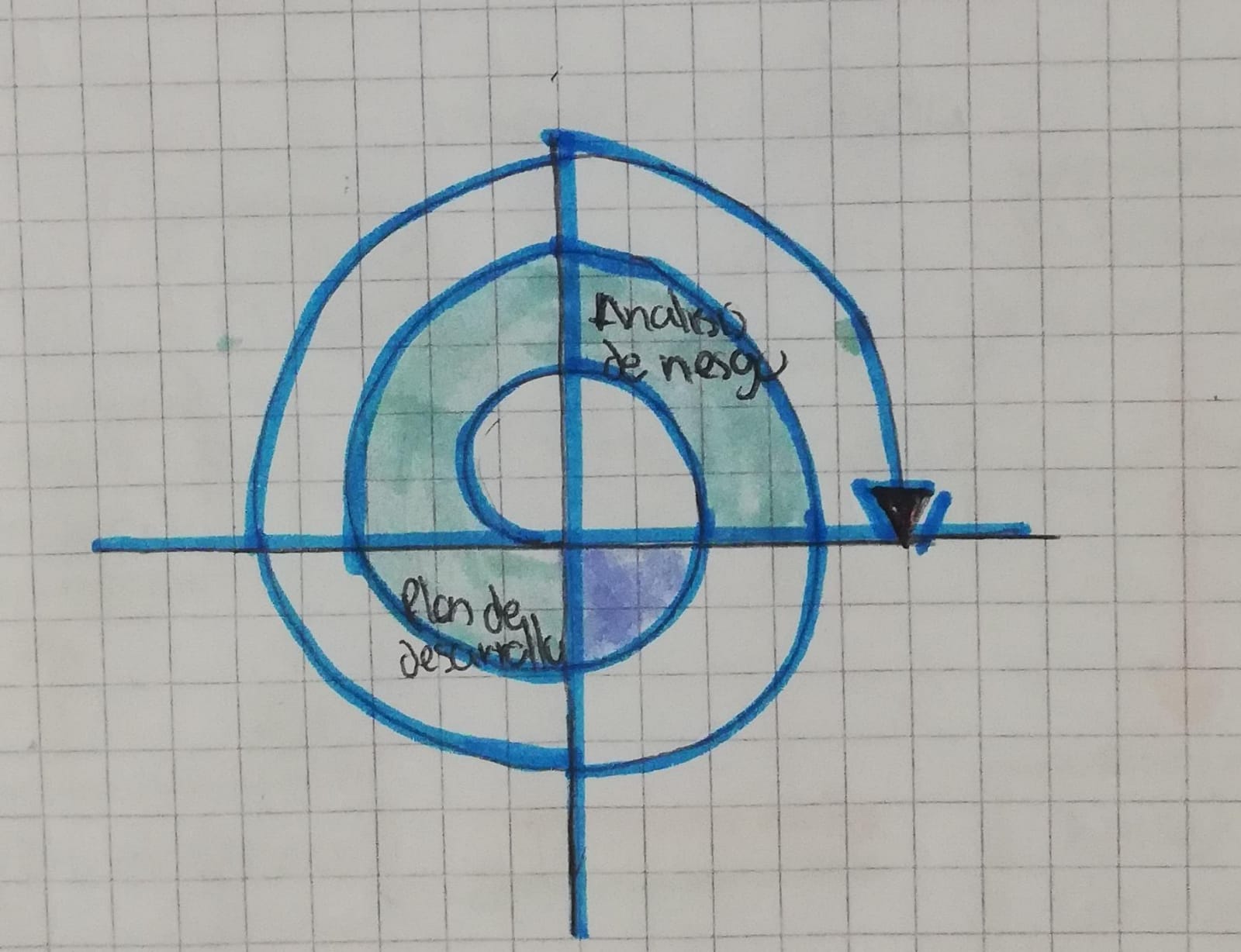
Este artículo presenta una propuesta de transformar una arquitectura en capas hacia una arquitectura hexagonal apoyada en Domain-Driven Design (DDD), Adoptar DDD y la arquitectura hexagonal no solo mejora la mantenibilidad y escalabilidad, también fomenta una buena comunicación entre expertos del negocio y desarrolladores. Esta propuesta no sólo soluciona problemas técnicos, sino que también cambia la forma en que pensamos en el software, de ser un conjunto de capas a volverse un reflejo vivo y adaptable del dominio que quiere modelar.

**Bibliografía**

* Cambarieri, M., Difabio, F., & García Martínez, N. (2020). Implementación de una Arquitectura de Software guiada por el Dominio. In *XXI Simposio Argentino de Ingeniería de Software (ASSE 2020)-JAIIO 49 (Modalidad virtual)*.

1. **La Arquitectura de Software en el Proceso de Desarrollo: Integrando MDA al Ciclo de Vida en Espiral**

El artículo habla de cómo el desarrollo de software debe enfrentarse a problemas como la rapidez con que debe estar disponible, el alto grado de evolución de los sistemas y su complejidad. Contar con un modelo de la arquitectura en etapas tempranas es evidente, pues anticiparnos a la especificación detallada permite un modelo de alto nivel de la solución a los requerimientos, que en sucesivos refinamientos conducirá al producto final. Ante estos problemas, la propuesta de la OMG **arquitectura dirigida por modelos** plantea la construcción de software como la transformación sucesiva de modelos, desde un alto nivel de abstracción hasta la implementación en una plataforma concreta. El objetivo del trabajo es aplicar MDA dentro del ciclo de vida en espiral de Bohem para establecer el impacto de la arquitectura de software en sus etapas y cómo MDA minimiza el riesgo.

****

**Reflexión**

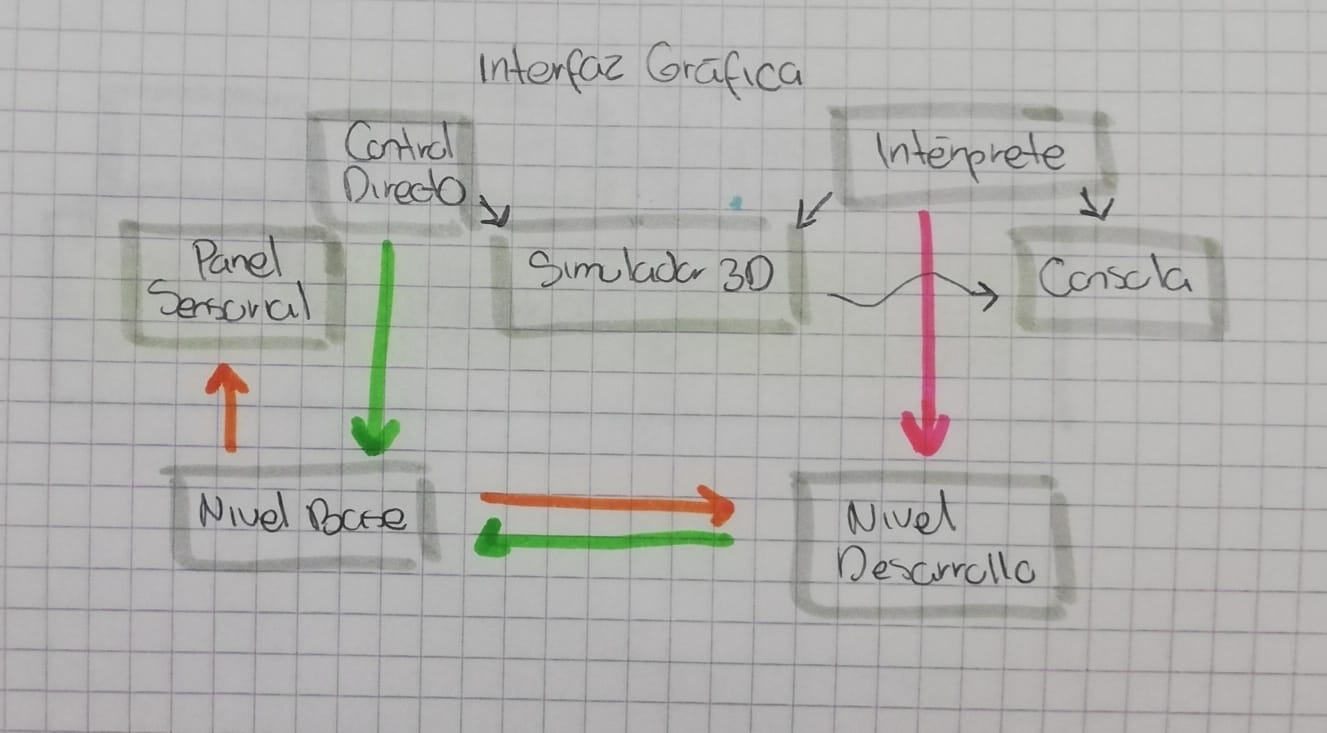
La integración de MDA en el ciclo de vida en espiral no solo mejora la productividad o la portabilidad, también mejora la trazabilidad, facilita la adaptación a cambios y minimiza riegos haciendo el proceso más efectivo. El combinar el carácter iterativo y de gestión de riesgos del ciclo en espiral con la trazabilidad y sistematización de MDA, se obtiene un proceso que no solo construye software, sino que construye confianza, en que lo que se desarrolla responde realmente a las necesidades del cliente, es flexible ante el cambio y sostenible a largo plazo.

**Bibliografía**

* Meaurio, V. S., & Schmieder, E. (2013). La arquitectura de software en el proceso de desarrollo: integrando MDA al ciclo de vida en espiral. *Archivo de la Revista Latinoamericana de Ingeniería de Software*, *1*(4), 142-1

1. **Desarrollo de una arquitectura de software para el robot móvil Lázaroto**

Una arquitectura de software para el robot móvil Lázaro, desarrollado en el Laboratorio de Prototipos de la Universidad Nacional Experimental del Táchira. La arquitectura propuesta posee tres niveles, un nivel base que gestiona el uso de los distintos elementos del robot a bajo nivel. Un segundo nivel de desarrollo formado por un conjunto de librerías que permiten generar aplicaciones para el control del robot en conjunto con el nivel base. Por último, un nivel de interfaz que permite el control y programación de Lázaro a alto nivel. En esta capa se le brinda al usuario una interfaz que consta de un panel de control directo y un simulador 3D, donde se pueden observar las acciones que ejecutan los actuadores, además se puede monitorear la información sensorial y se pueden programar las instrucciones a ser ejecutadas por el robot móvil. Finalmente, se presentan varios casos de prueba para verificar las funcionalidades y capacidades de esta arquitectura desarrollada en C#.



**Reflexión**

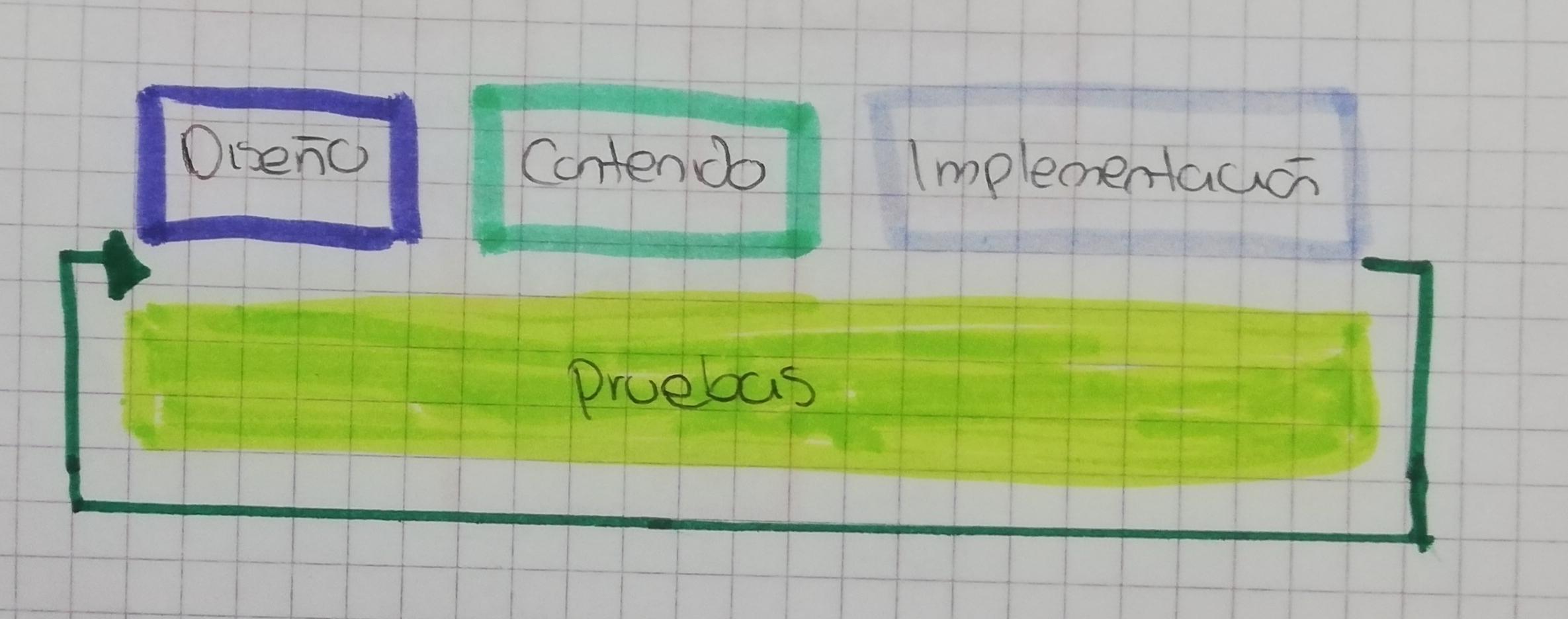
El desarrollo de la arquitectura de software para el robot móvil Lázaro demuestra la importancia de diseñar sistemas modulares, escalables y flexibles para aplicaciones robóticas complejas. Esta arquitectura permite separar responsabilidades, facilitando la integración de sensores, actuadores y rutinas personalizadas sin comprometer la estabilidad del sistema. La implementación del patrón MVC y la inclusión de un simulador 3D no solo mejoran la trazabilidad y el control de las acciones del robot, sino que también reducen riesgos al permitir pruebas virtuales antes de ejecutar comportamientos en el robot físico. Las pruebas realizadas confirman que la arquitectura combina eficiencia, precisión y facilidad de uso, evidenciando un diseño sólido y bien estructurado.

**Bibliografía**

* García, J. M., Gil, Á. E., & Sánchez, E. A. (2018). Desarrollo de una arquitectura de software para el robot móvil Lázaro. *Ingeniare. Revista chilena de ingeniería*, *26*(3), 376-390.

1. **Propuesta de una arquitectura de software para el desarrollo de sitios web accesibles**

En esta investigación se realiza una arquitectura de software para el desarrollo de sitios web accesibles. En los resultados se analizan los patrones de prácticas de accesibilidad en entornos de trabajo de JavaScript, se abstraen los mismos y se alinean según las directrices de la WCAG. Luego se propone la arquitectura de Software considerando los hallazgos desde el diseño, la generación de contenido, implementación y pruebas de manera iterativa. Esto permite un ahorro de tiempo y recursos al considerar la accesibilidad desde el inicio del desarrollo de un sitio web y no su verificación al final como lista de verificación. Para la validación de la arquitectura se desarrolló un sitio web y se corroboro la accesibilidad haciendo la evaluación con distintas herramientas de evaluación automática de la accesibilidad web. La arquitectura de software propuesta permite la implementación de la accesibilidad web desde el inicio del desarrollo de un sitio web.



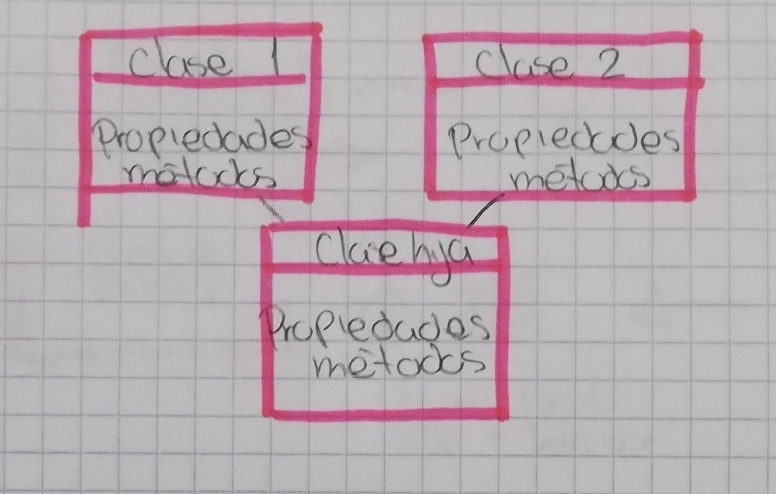
**Reflexión**

La propuesta de esta arquitectura de software demuestra lo importante que es pensar en la accesibilidad desde el inicio al diseñar un sitio web. Al organizar el sistema en capas, se facilita que cada parte cumpla su función, la interfaz se adapta a los usuarios, la lógica asegura que la información se guarde de forma organizada y eficiente. Las pruebas con usuarios mostraron que esta arquitectura no solo mejora la navegación y comprensión del contenido, sino que también fomenta un desarrollo más inclusivo y responsable.

**Bibliografía**

* Palomeque-Zambrano, E. E., & Campoverde-Molina, M. (2023). Propuesta de una arquitectura de software para el desarrollo de sitios web accesibles. *MQRInvestigar*, *7*(3), 1458-1474.

1. **Arquitectura de software con programación orientada a objeto**

Se ha encontrado que La Programación Orientada al Objeto (POO), agrupa un conjunto de técnicas que permiten desarrollar y mantener mucho más fácilmente programas de una gran complejidad. Es un paradigma de programación que define los programas en términos de “clases de objetos”, Las características de orientación a objetos han sido agregadas a muchos lenguajes existentes como Python, Java, PHP, C++, Objective C, TypeScript, Smalltalk, entre otros, de este modo, C++ y Java son los dos lenguajes de programación orientada a objetos más usados. Se puede concluir que, la evolución de la Arquitectura de Software escala cada vez más posiciones superiores, que permiten a los profesionales del dicho campo al desarrollo de software que permiten resolver diversos problemas de arquitectura con mejor calidad y en un menor tiempo para la satisfacción del usuario.

**Reflexión**

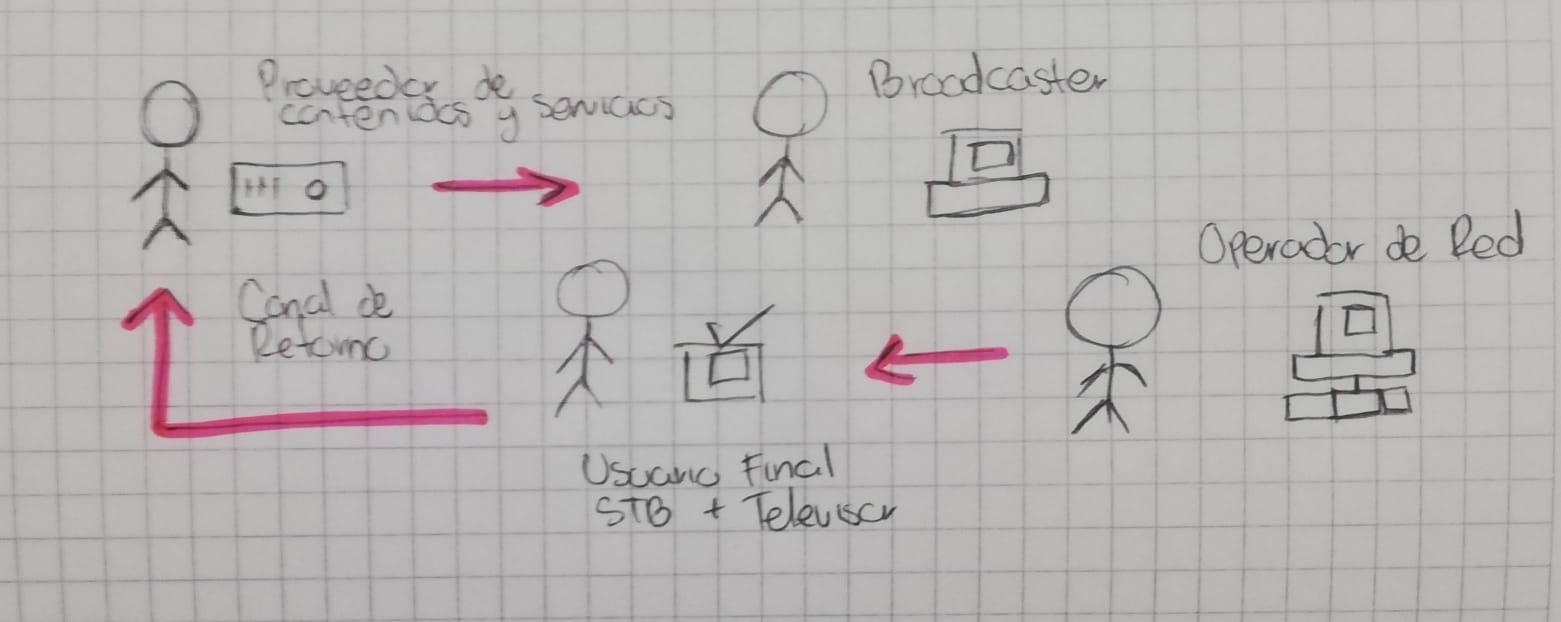
El articulo muestra cómo la programación orientada a objetos y una buena arquitectura de software son clave para crear sistemas complejos que funcionen bien. Al organizar el software en clases y objetos, se facilita entenderlo, mantenerlo y reutilizarlo. Conceptos como herencia y encapsulamiento ayudan a reflejar situaciones del mundo real dentro del programa. Este articulo me invita a reflexionar que aplicar POO y patrones de diseño permite desarrollar software de calidad, más rápido y adaptable, haciendo que los proyectos sean confiables y sostenibles a largo plazo.

**Bibliografía**

* Vera, J. B. V. (2023). Arquitectura de software con programación orientada a objeto. *Polo del Conocimiento: Revista científico-profesional*, *8*(12), 1497-15

1. **Arquitectura de Software para el Soporte de Comunidades Académicas Virtuales en Ambientes de Televisión Digital Interactiva**

En este artículo se propone y analiza una arquitectura de software para el soporte de comunidades académicas virtuales (CAV), en el contexto de la televisión digital interactiva. La arquitectura propuesta extiende los servicios hasta ahora definidos dentro del ambiente de e-learning, introduciendo conceptos y aplicaciones de la Web 2.0 mediante un esquema de consumo de servicios REST-JSON. Se describen los escenarios soportados por la arquitectura, los componentes y su interoperabilidad que permiten dotar de interactividad a los diferentes servicios en un entorno de televisión digital, y los avances de su implementación en el laboratorio. Se trata de una solución innovadora que permite combinar aprendizaje electrónico y medios tradicionales como la televisión, con comunicación en tiempo real.

****

**Reflexión**

La propuesta de esta arquitectura de software demuestra cómo la tecnología puede ampliar y enriquecer el aprendizaje, llevando las comunidades académicas virtuales a entornos de televisión digital interactiva. Al integrar herramientas de la web como foros, chats, wikis, se crea un entorno flexible y accesible para los usuarios. Esta arquitectura evidencia que, con un diseño bien pensado, la educación puede adaptarse a nuevas tecnologías y contextos, abriendo oportunidades.

**Bibliografía**

* Campo, W. Y., Chanchí, G. E., & Arciniegas, J. L. (2013). Arquitectura de software para el soporte de comunidades académicas virtuales en ambientes de televisión digital interactiva. *Formación universitaria*, *6*(2), 03-14