**Artículos sobre la arquitectura y patrones de software**

**Mayra Alejandra Tamayo Perdomo**

**Jesús Ariel**

**Servicio Nacional de Aprendizaje SENA**

**Centro de la industria y los servicios**

**Análisis y Desarrollo de software 2694667**

**2024**

Contenido

[1. Introducción 5](#_Toc182631252)

[2. Modelado y verificación de patrones de diseño de arquitectura de software para entornos de la computación en la nube 6](#_Toc182631253)

[3. Desarrollo de un herramienta para el aprendizaje de patrones de diseño software 7](#_Toc182631254)

[4. Implementación de un framework para el desarrollo de aplicaciones web utilizando patrones de diseño y arquitectura MVC/REST 8](#_Toc182631255)

[5. Desarrollo de una arquitectura de software para el robot móvil lazaro 9](#_Toc182631256)

[6. Arquitectura de software, esquemas y servicio 10](#_Toc182631257)

[7. Arquitectura de software para el desarrollo de herramientas tecnológicas de costo, presupuestos y programas de obra 11](#_Toc182631258)

[8. Lenguajes de patrones de arquitectura de software: una aproximación al estado de arte 12](#_Toc182631259)

[9. Arquitectura de software basado en micro servicios para el desarrollo 13](#_Toc182631260)

[10. Patrones de diseño Gof en el contexto de procesos de desarrollo de aplicaciones orientadas a la web 14](#_Toc182631261)

[11. Patrones de usabilidad: mejora la usabilidad del software desde el momento arquitectónico 15](#_Toc182631262)

[12. Arquitectura del software para el sistema de visualización medica vismedic 16](#_Toc182631263)

[13. Arquitectura de software reutilizable basada en patrones de interacción, para el desarrollo rápido de aplicaciones web 17](#_Toc182631264)

[14. Revisión sistemática sobre generadores de código fuente y patrones arquitectónicos 18](#_Toc182631265)

[15. Perfiles UML para definición de patrones de diseño 19](#_Toc182631266)

[16. Herramientas para reusó de código JavaScript orientado a patrones de interacción 20](#_Toc182631267)

[17. Introducción a la arquitectura de software 21](#_Toc182631268)

[18. Aplicación de patrones de diseño para garantizar alta flexibilidad en el software 21](#_Toc182631269)

[19. Arquitectura de software académico para la comprensión del desarrollo de software en capas 23](#_Toc182631270)

[20. Introducción a los patrones de diseño 24](#_Toc182631271)

[21. Modulo de recomendación de patrones de diseño EGPat 25](#_Toc182631272)

[22. Arquitectura de software para el soporte de comunidades académicas virtuales en ambientes de televisión digital interactiva 26](#_Toc182631273)

[23. Desarrollo de sistemas de software con patrones de diseño orientado a objetos 27](#_Toc182631274)

[24. Atributos de calidad y arquitectura de software 28](#_Toc182631275)

[25. Arquitectura de software en el proceso de desarrollo ágil: una perspectiva basada en requisitos significantes para la arquitectura 29](#_Toc182631276)

[26. Análisis comparativos de patrones de diseño de software 30](#_Toc182631277)

[27. Identificación y clasificación de patrones de diseño de servicios web para mejorar Qos 31](#_Toc182631278)

[28. Especificación del patrón de diseño memento atreves de un perfil UML 32](#_Toc182631279)

[29. Documentación y análisis de los principales frameworks de arquitectura de software en aplicaciones empresariales 33](#_Toc182631280)

[30. Implementación de una arquitectura de software guiada por el dominio 34](#_Toc182631281)

[Referencias 35](#_Toc182631282)

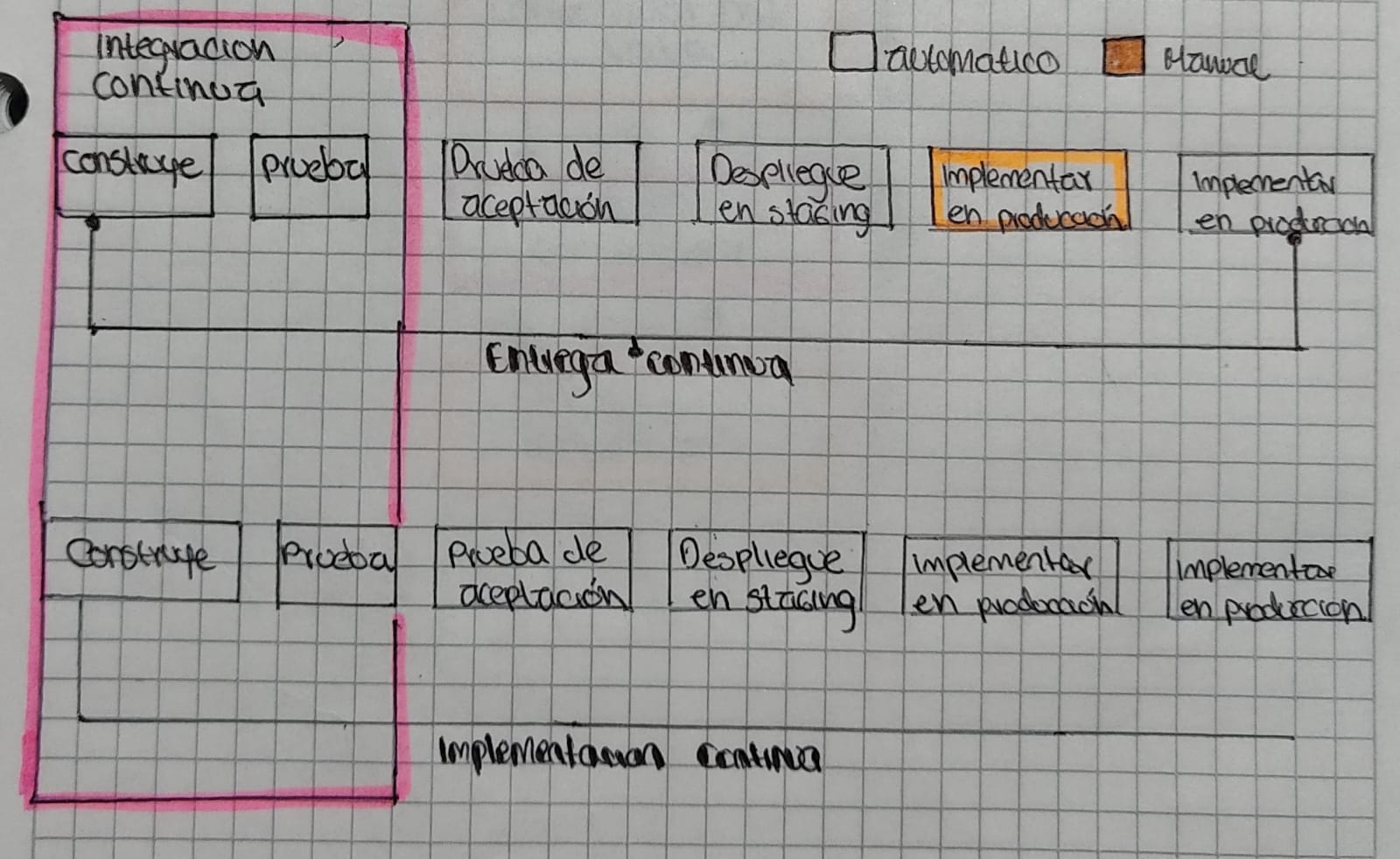
[Apéndices 36](#_Toc182631283)

# 1. Introducción

En el ámbito del desarrollo de software, la arquitectura y los patrones de diseño son esenciales para crear sistemas que sean robustos, escalables y fáciles de mantener. Este documento reúne varios artículos que analizan conceptos importantes, mejores prácticas relacionados con la arquitectura y los patrones de software.

# 2. Modelado y verificación de patrones de diseño de arquitectura de software para entornos de la computación en la nube

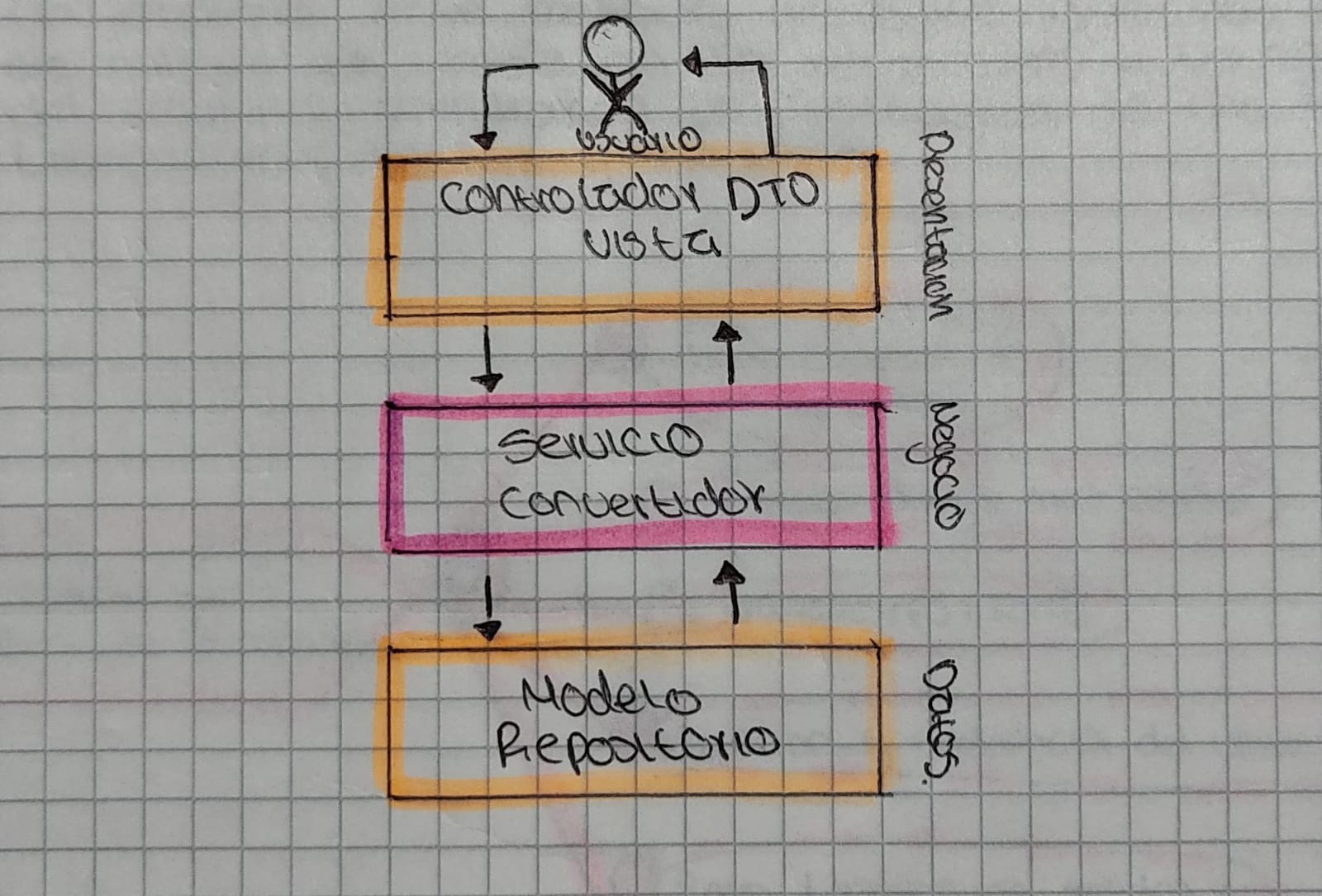
El documento aborda desafíos de garantizan la calidad en la producción de software, especialmente en entornos de informática en la nube(computación en la nube ). Los desafíos incluyen la falta de modelos de arquitectura maduros, documentación y un lenguaje deficiente para comunicar exigencias de calidad. Para tratar el problema se propone un instrumento basados en un matamodelo de patrones que facilitan el modelado y verificación de manera automática su uso mejorando la eficiencia y precisión del diseño, el propósito del ambiente integrado es giar a los arquitectos mediante validaciones y sugerencias inteligentes para potenciar la calidad de las soluciones en la computación en la nube.



# 3. Desarrollo de un herramienta para el aprendizaje de patrones de diseño software

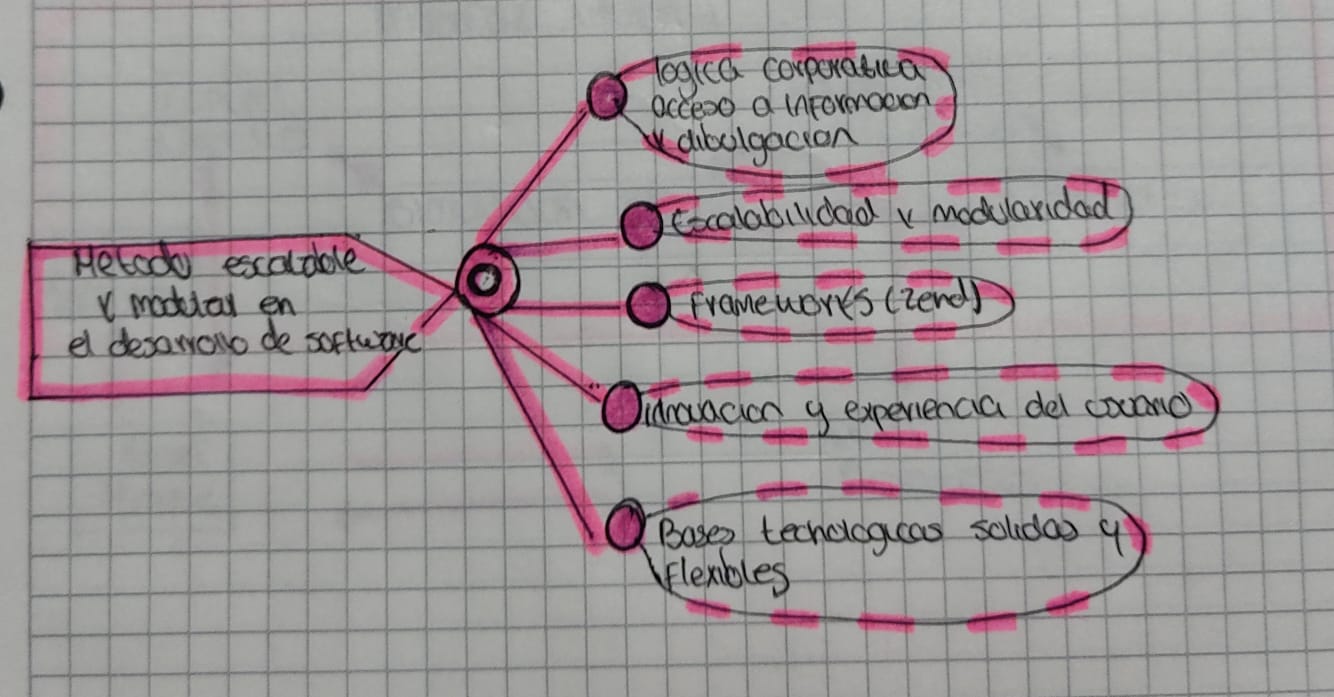
En este artículo se destaca y se detalla un proyecto enfocado en el diseño y desarrollo de un instrumento para simplificar el aprendizaje del diseño en programación teniendo en cuenta su relevancia en la creación del software.

Por las limitaciones del covid-19, el proyecto se llevó acabo de forma remota; el estudiante adquirió experiencias en las tecnologías spring y mejoro las habilidades de desarrollo web en frontend y backend unque no se logró alcanzar todas las características esperadas, el proyecto da como resultado un prototipo funcional que evoluciona en el futuro



# 4. Implementación de un framework para el desarrollo de aplicaciones web utilizando patrones de diseño y arquitectura MVC/REST

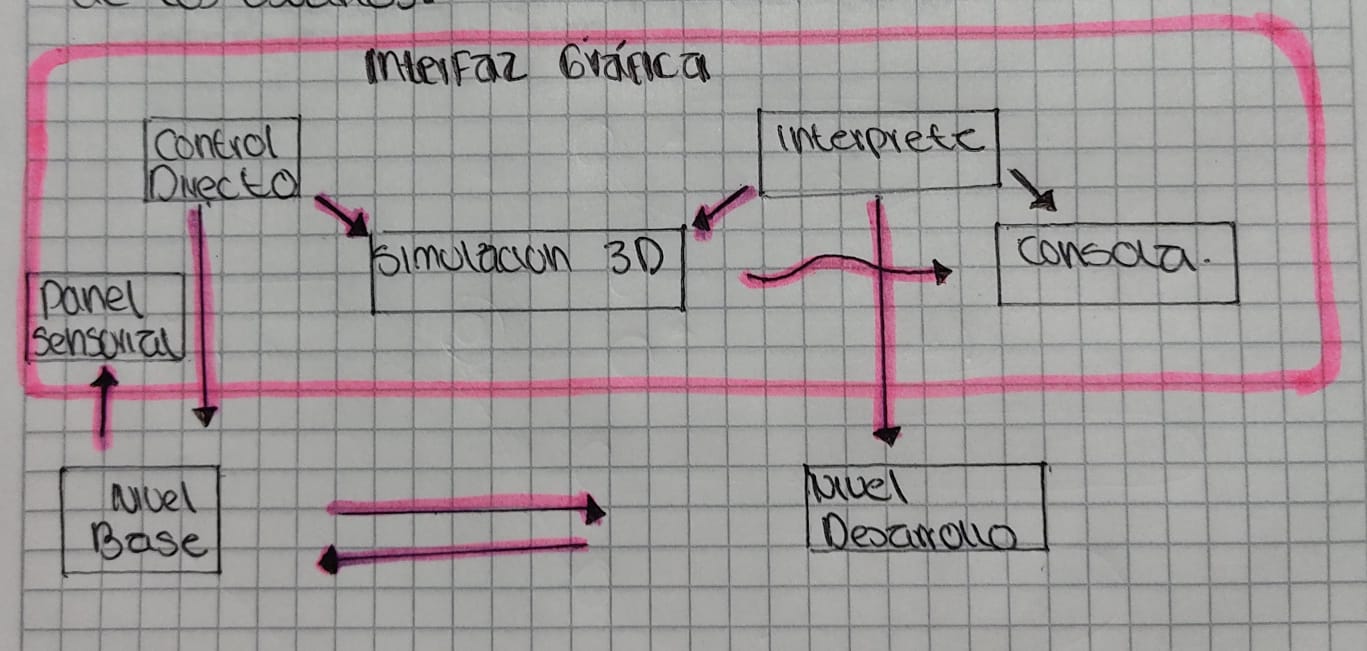
El artículo se centra en la importancia de utilizar un método escalable y modular en la elaboración de software, al distinguir la lógica corporativa, el acceso a la información y la divulgación, esta no solo facilita la gestión de cambios y amplificaciones sino que además proporciona flexibilidad y sostenibilidad, el uso de frameworks verificados y flexible, como zend, resalta la manera en que las herramientas ya existentes puedan ser mejoradas para satisfacer, las necesidades de las empresas, este método posibilita que las empresas se concentren más en la innovación y la experiencia del usuario, asegurando la protección de una base tecnológica salida y flexible.



# 5. Desarrollo de una arquitectura de software para el robot móvil lazaro

El texto aborda diferentes tipos de arquitectura de software aplicadas a robots, centrándose en modelos deliberativo, reactivo e hibrido.

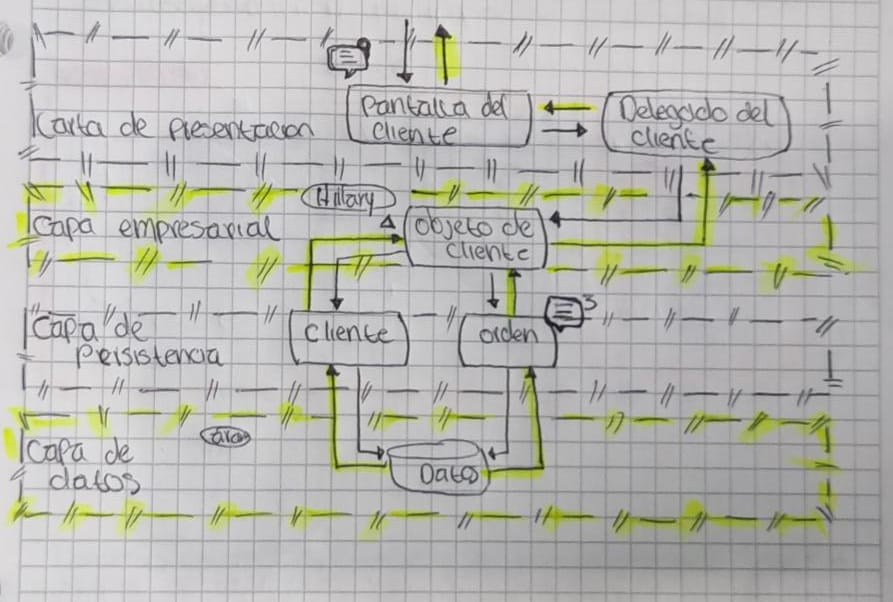
Las arquitecturas hibridas fusionan ambos enfoques con el objetivo de optimizar la capacidad de respuestas de los robots antes situaciones inesperadas y entornos complicados, un ejemplo es la arquitectura del robot móvil lázaro, que combina el control jerárquico y reactivo, empleando c# para la gestión de actuadores y sensores, lo que proporciona flexibilidad y escalabilidad, la creación de arquitecturas de software para robots resalta la relevancia de la flexibilidad, escalabilidad y adaptabilidad para cumplir con las necesidades de los usuarios.



# 6. Arquitectura de software, esquemas y servicio

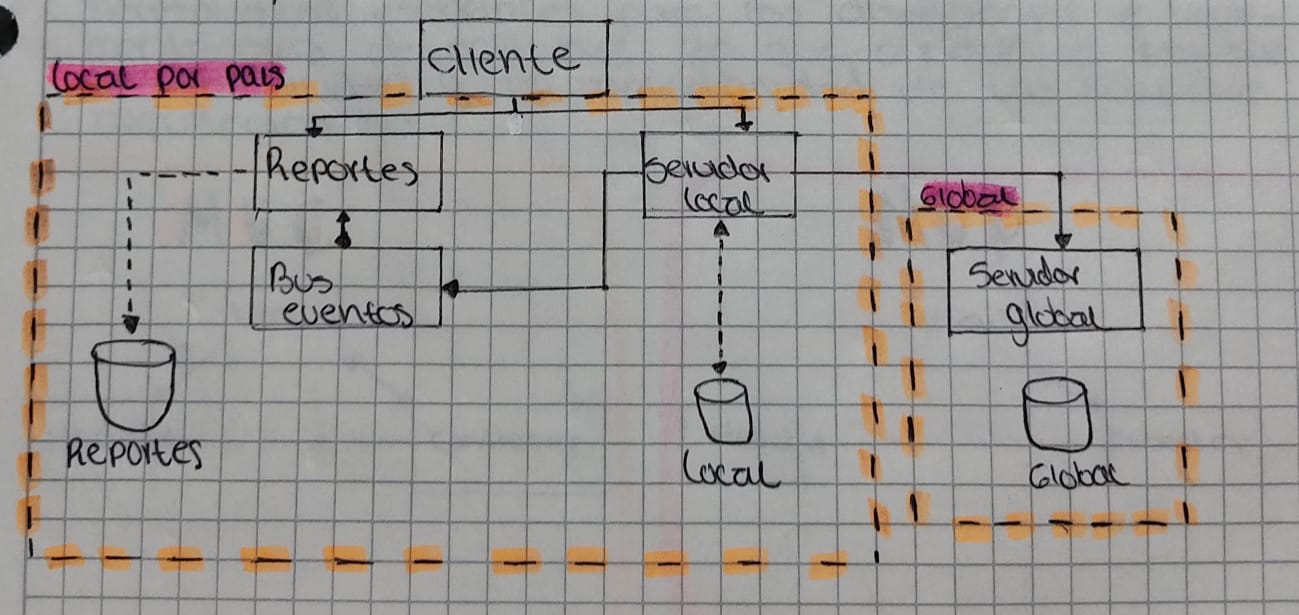
El texto trata sobre cómo cada vez más empresas necesitan aplicaciones que tengan funciones avanzadas que antes no eran tan comunes. Estas funciones incluyen la posibilidad de funcionar sin depender de un sistema operativo o base de datos especifico, poder acceder a ellas desde diferentes lugares, integrarse con otros sistemas y manejar muchas interacciones al mismo tiempo.

Se menciona que utilizar arquitecturas orientadas a servicios (SOA) es importante por que ayuda a que los diferentes componentes de una aplicación este menos conectados entre sí, lo que mejora ña reutilización y hace que sea más fácil adaptar las aplicaciones sin importar donde estén o que plataforma se use.



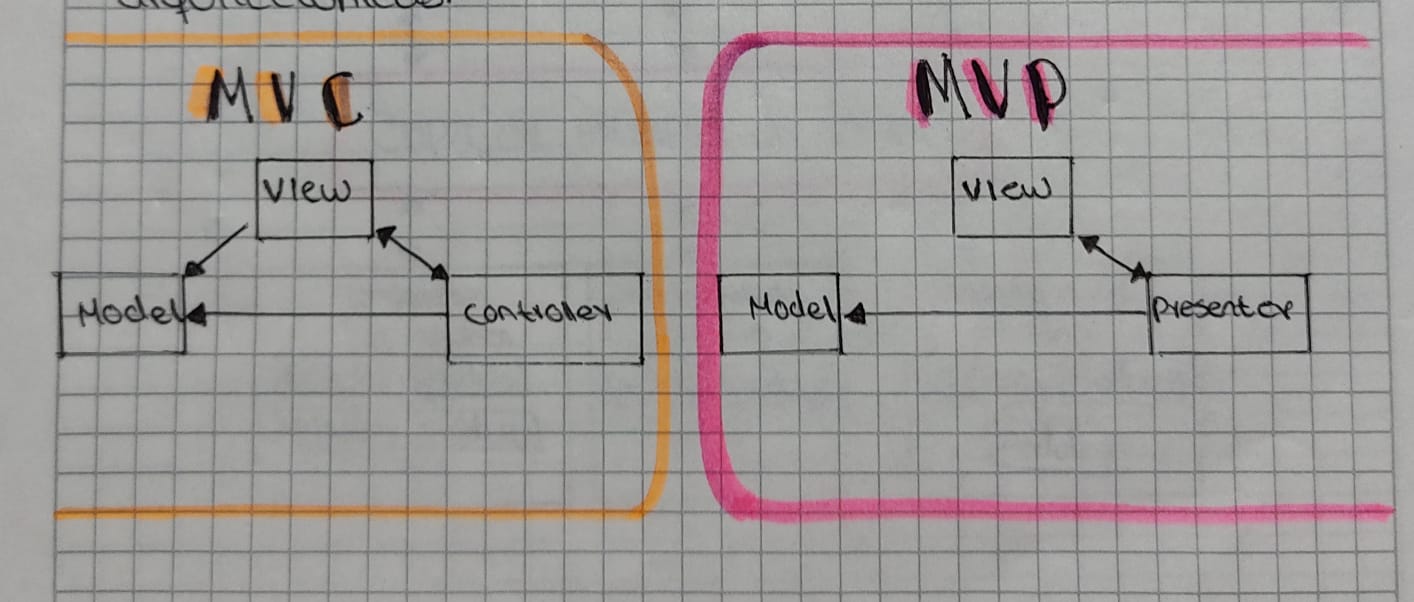
# 7. Arquitectura de software para el desarrollo de herramientas tecnológicas de costo, presupuestos y programas de obra

El texto destaca la importancia de las Tecnologías de la Información y las Comunicaciones (TIC) como pilares de la economía, la educación y la sociedad en general, que afectan de manera directa a la educación superior al requerir nuevas herramientas y métodos de enseñanza que se adapten al avance tecnológico. En particular, el campo de la Ingeniería Civil enfrenta desafíos debido al aumento en la construcción de proyectos complejos que requieren una gestión precisa de costos y presupuestos. Para abordar estas necesidades, se propone desarrollar un software educativo en la Universidad Francisco de Paula Santander, enfocado en la administración de costos, presupuestos y planificación de trabajos. La implementación de herramientas específicas en los programas educativos, como el software propuesto para Ingeniería Civil, contribuye a la formación de profesionales capacitados para enfrentar estos retos.



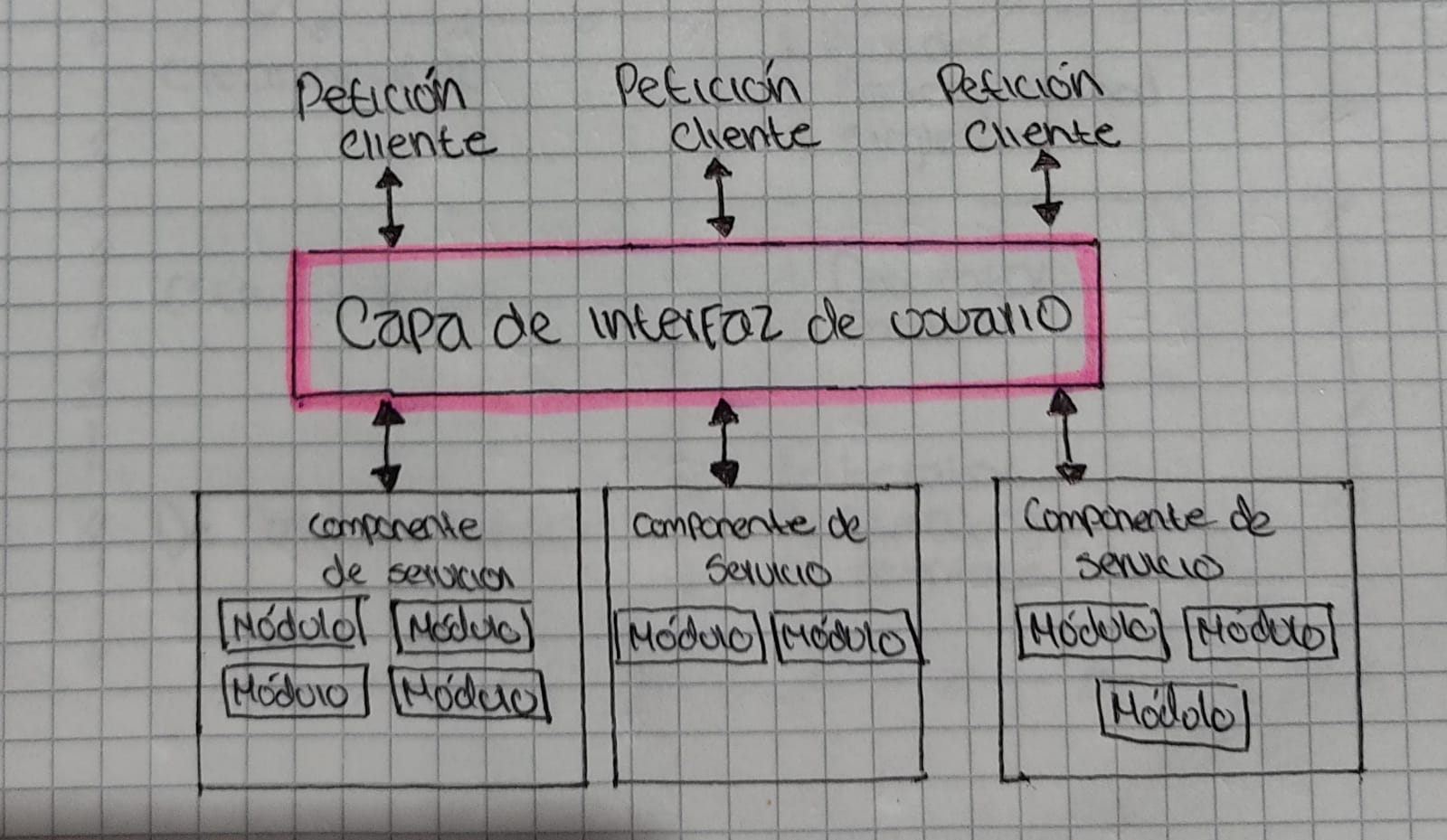
# 8. Lenguajes de patrones de arquitectura de software: una aproximación al estado de arte

Este artículo muestra cómo la evolución de la arquitectura de software ha ayudado mucho a los desarrolladores a enfrentar los problemas complicados de los sistemas de información. Con el avance de la tecnología, es importante seguir mejorando estas herramientas y crear procesos más sólidos que permitan a los diseñadores de software construir arquitecturas que no solo solucionen problemas actuales, sino que también se adapten y crezcan para enfrentar los desafíos del futuro. El objetivo principal del artículo es dar una visión general sobre el estado actual de la Arquitectura de Software, especialmente en el uso de Lenguajes de Patrones. La investigación cubre desde los inicios de esta arquitectura hasta sus últimos avances, resaltando su uso en la creación de sistemas de información en diferentes áreas. Los Lenguajes de Patrones son presentados como herramientas esenciales para los diseñadores e implementadores de software, ya que ayudan a mejorar la flexibilidad y la aplicabilidad de las soluciones arquitectónicas.



# 9. Arquitectura de software basado en micro servicios para el desarrollo

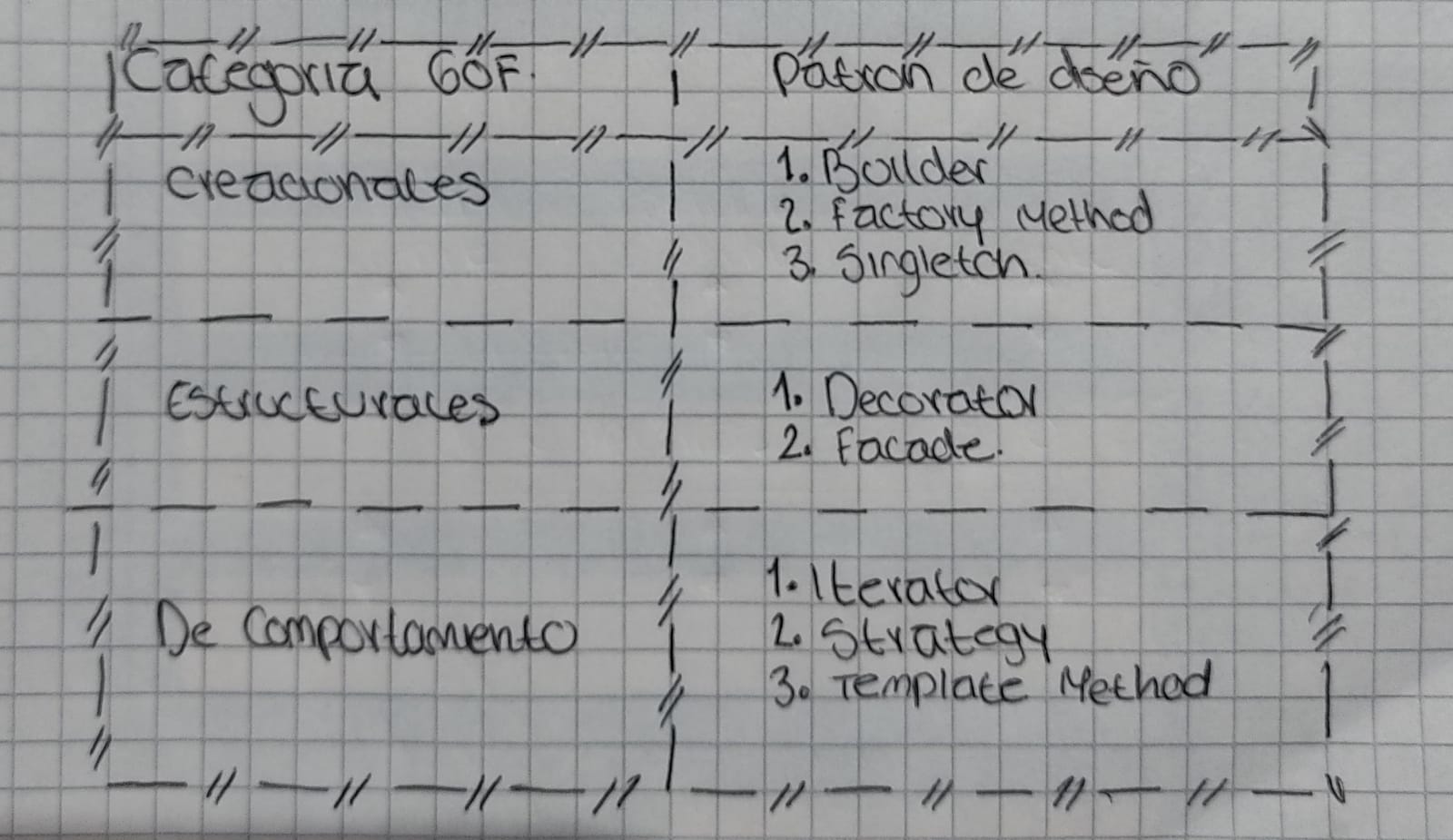
Hoy en día, muchas compañías, tanto del gobierno como del sector privado, crean programas para hacer más eficientes sus procesos internos, usando arquitecturas tradicionales y monolíticas. No obstante, el uso de una arquitectura monolítica ha traído consigo problemas como el mantenimiento complicado, la falta de escalabilidad y dificultades al agregar nuevas funciones. Actualmente, se está investigando para encontrar una nueva arquitectura de software que se ajuste mejor a las tecnologías actuales y que ayude a solucionar estos inconvenientes. La idea de cambiar a una arquitectura más moderna es una respuesta sensata a estos retos.



# 10. Patrones de diseño Gof en el contexto de procesos de desarrollo de aplicaciones orientadas a la web

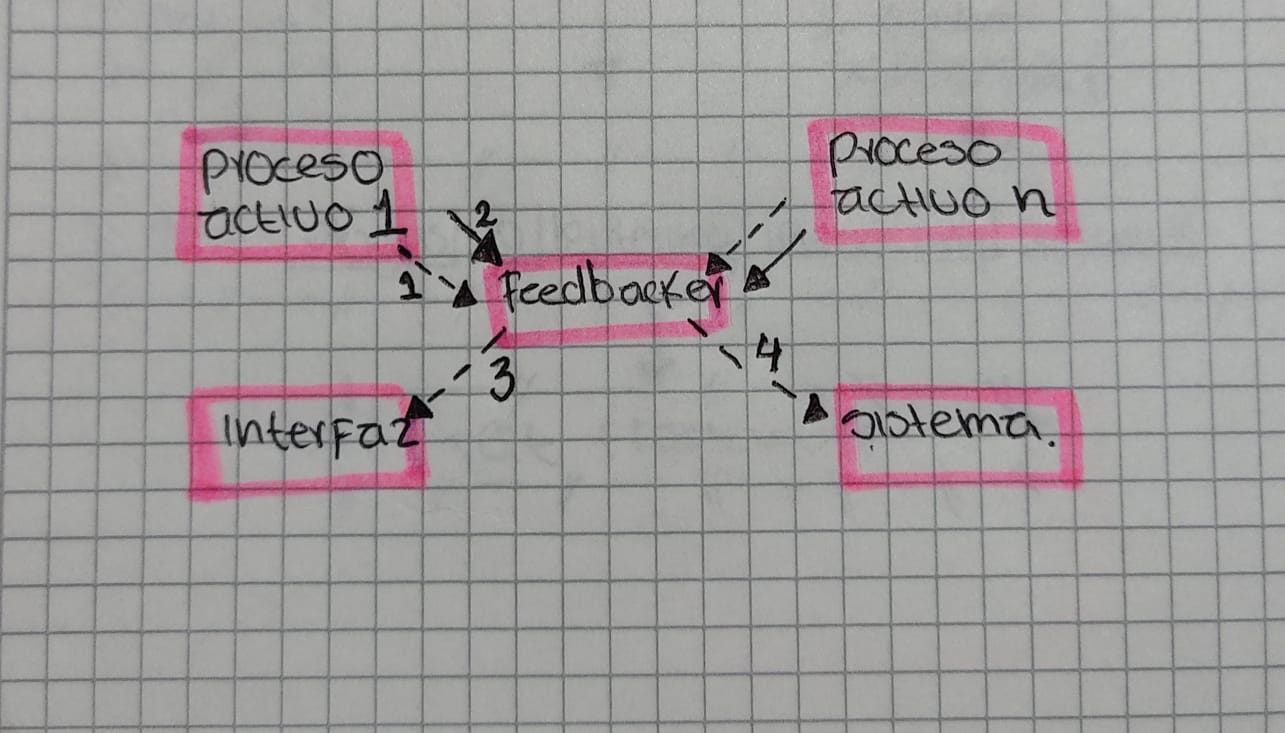
Este artículo aborda la investigación sobre los Patrones de Diseño GOF (Gang of Four) en el contexto de la ingeniería del software, específicamente en la industria de software de Colombia.

El GRIIS identificó los patrones más utilizados en el desarrollo de software en Colombia, destacando que su implementación mejora la calidad y facilita el mantenimiento. Se advierte sobre el uso cuidadoso de patrones, ya que no todos los problemas los requieren, evitando complicaciones en el diseño.



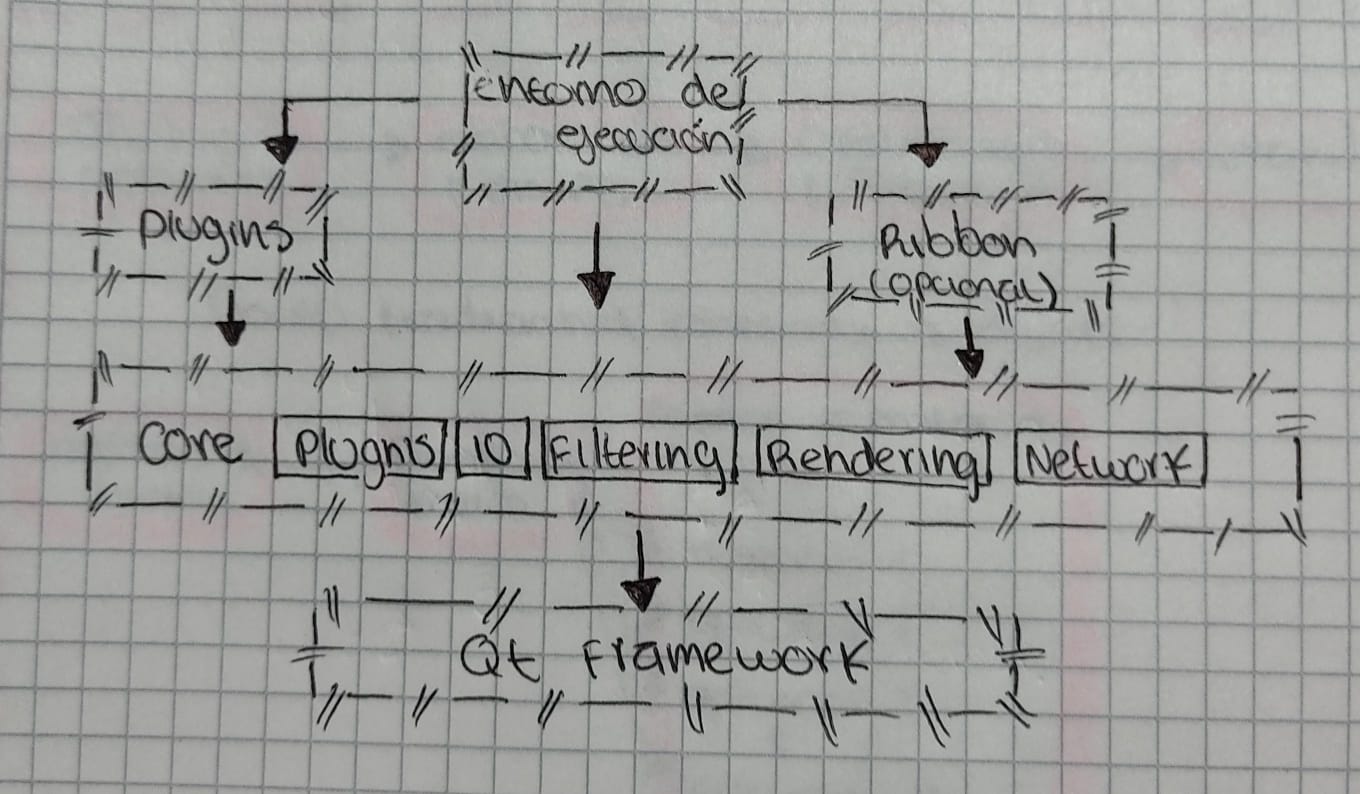
# 11. Patrones de usabilidad: mejora la usabilidad del software desde el momento arquitectónico

El artículo se centra en la importancia de seleccionar y aplicar patrones arquitectónicos desde el principio para optimizar la usabilidad de un sistema, permitiendo que la arquitectura se diseñe para cumplir con requisitos específicos. Aunque aún no hay resultados empíricos disponibles, se espera que este enfoque mejore la usabilidad del software final. El proyecto STATUS propone mejorar la usabilidad desde las primeras etapas del desarrollo de software mediante la integración de evaluaciones arquitectónicas y el uso de patrones específicos.

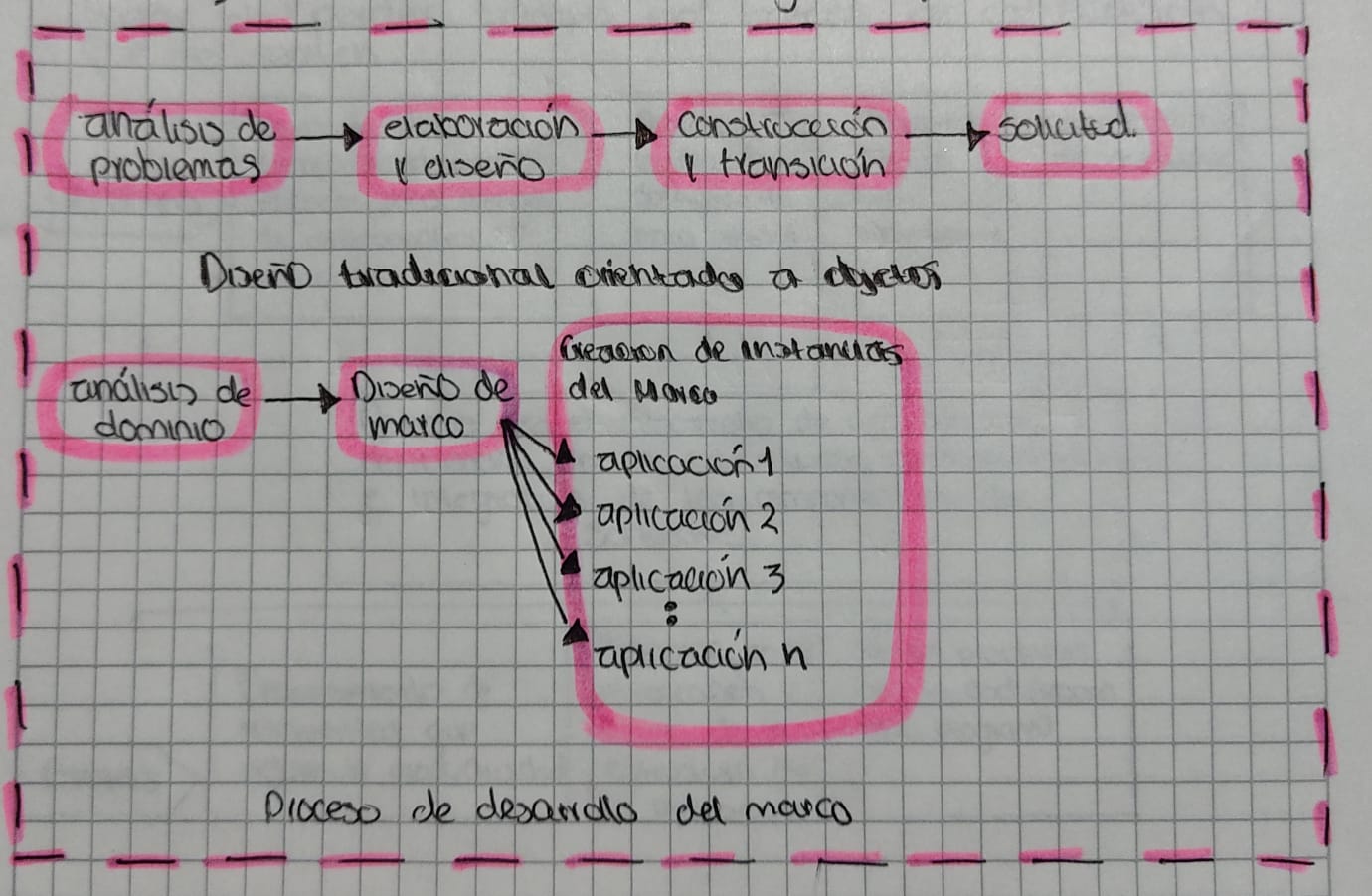


# 12. Arquitectura del software para el sistema de visualización medica vismedic

El diseño de la arquitectura de software es esencial para satisfacer las necesidades de las organizaciones. La reutilización de conceptos de otros proyectos, junto con una red de flujo de datos dinámica, mejora la eficiencia del sistema y fomenta la innovación. La integración de tecnologías existentes y configuraciones dinámicas no solo resuelve problemas técnicos, sino que también mejora la experiencia del usuario al permitir soluciones personalizadas sin conocimientos técnicos profundos.

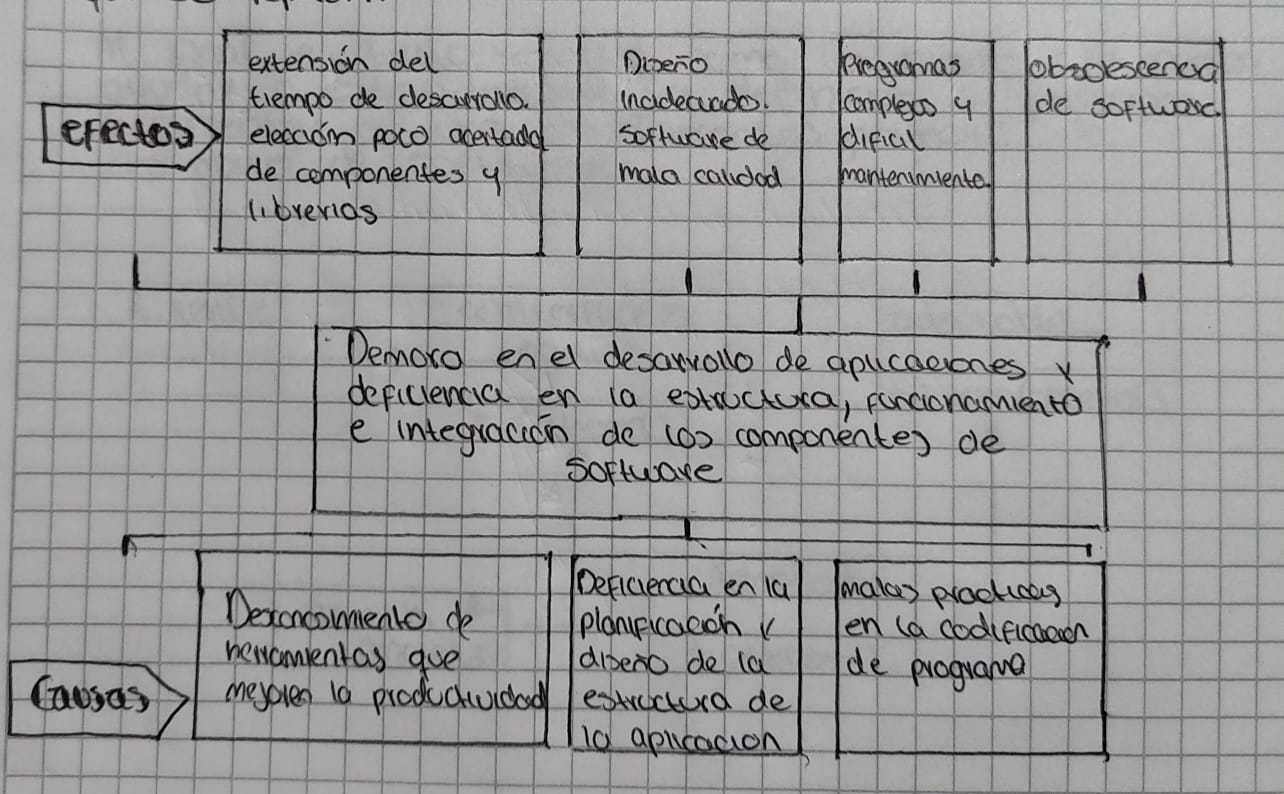


# 13. Arquitectura de software reutilizable basada en patrones de interacción, para el desarrollo rápido de aplicaciones web

Este proyecto muestra cómo se ha creado un marco reutilizable que permite hacer aplicaciones web de manera rápida y eficiente, especialmente para manejar contenidos. Se basa en patrones de diseño, como el patrón MVC (Modelo-Vista-Controlador) y otros patrones para la interfaz de usuario. El objetivo de este marco es proporcionar una solución común a problemas que suelen aparecer en el desarrollo de software, lo que ayuda a que las aplicaciones sean más fáciles de escalar, mantener y usar.

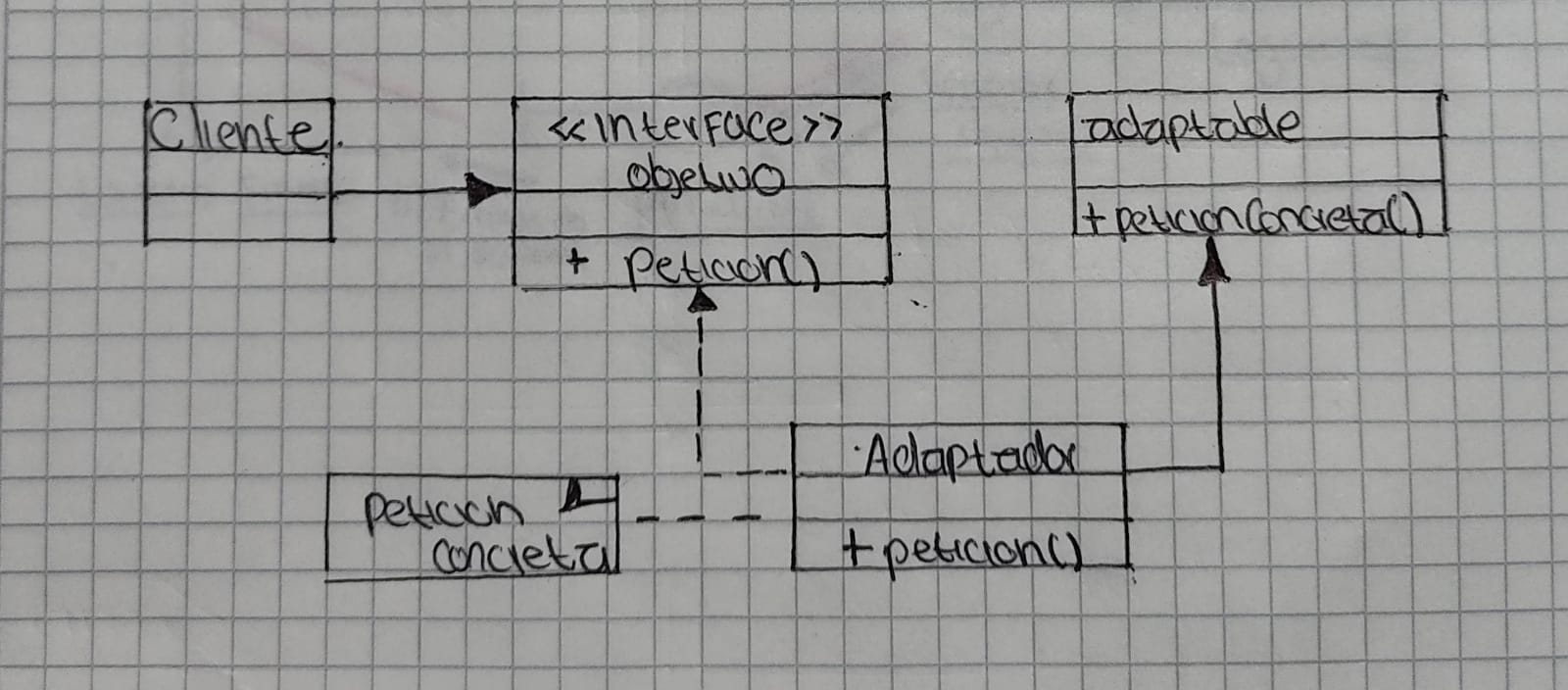
# 14. Revisión sistemática sobre generadores de código fuente y patrones arquitectónicos

Este artículo trata sobre los generadores de código fuente, que son herramientas que ayudan a automatizar gran parte de la programación. Estas herramientas crean código usando plantillas y reglas ya establecidas. Esto hace que el desarrollo de software sea más rápido, disminuye los errores y mejora la uniformidad del código. Al seguir patrones de arquitectura, estos generadores garantizan que el software se desarrolle de forma ordenada y que pueda crecer fácilmente. En resumen, los generadores de código permiten a los programadores enfocarse en la lógica del negocio y en la creatividad, en lugar de perder tiempo en tareas de codificación que se repiten.

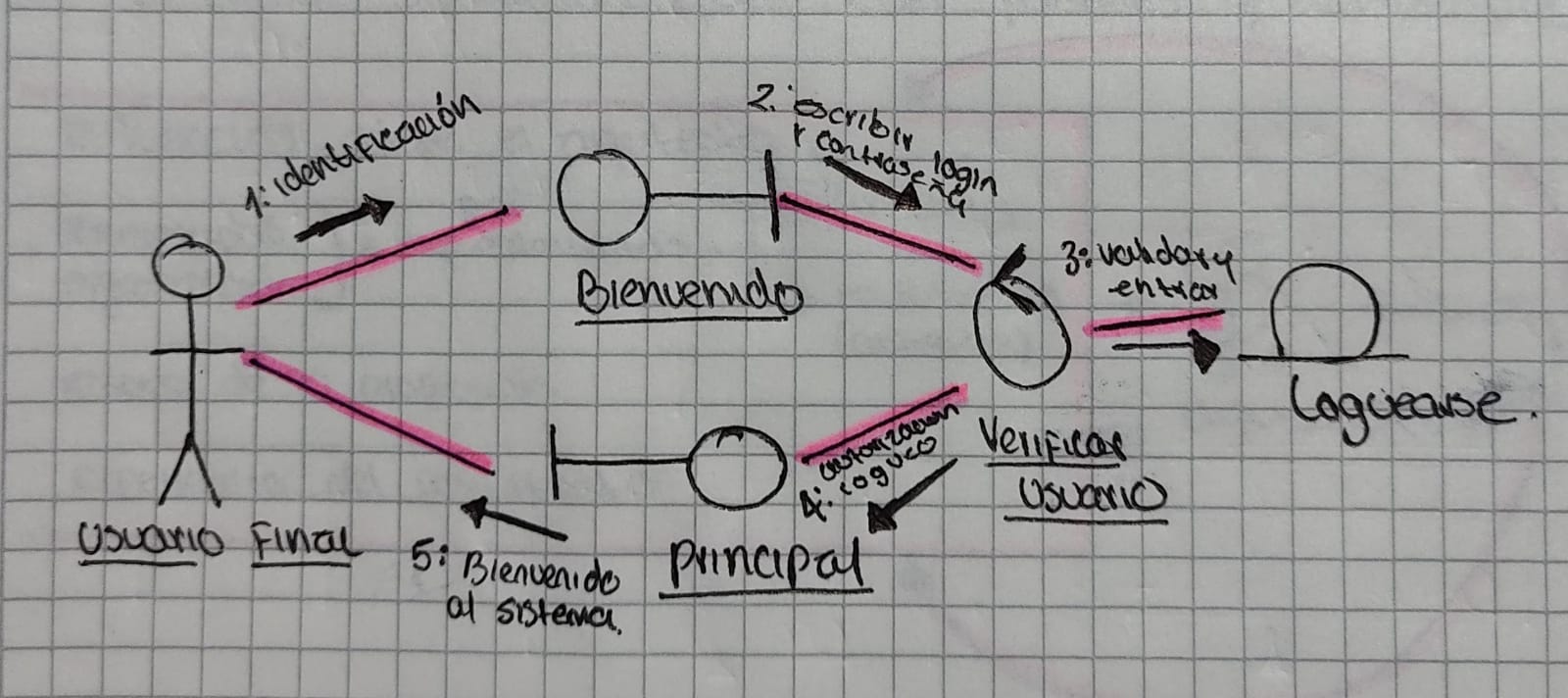


# 15. Perfiles UML para definición de patrones de diseño

El texto habla sobre cómo los perfiles UML son útiles para definir y documentar patrones de diseño en software. Aunque UML ya es una herramienta muy buena para modelar sistemas, los perfiles UML ayudan a ampliar su uso para representar ideas más específicas, como los patrones de diseño. Al hacer perfiles personalizados, los programadores pueden crear un vocabulario claro para explicar los componentes y las relaciones de un patrón. Esto hace que sea más fácil comunicarse, documentar y reutilizar los patrones. Además, al usar perfiles UML, se pueden aprovechar las herramientas UML que ya existen, sin tener que crear nuevas herramientas para cada patrón. Al final, el texto resalta cómo los perfiles UML pueden ayudar a estandarizar la definición de patrones y fomentar su uso en diferentes proyectos de desarrollo de software.

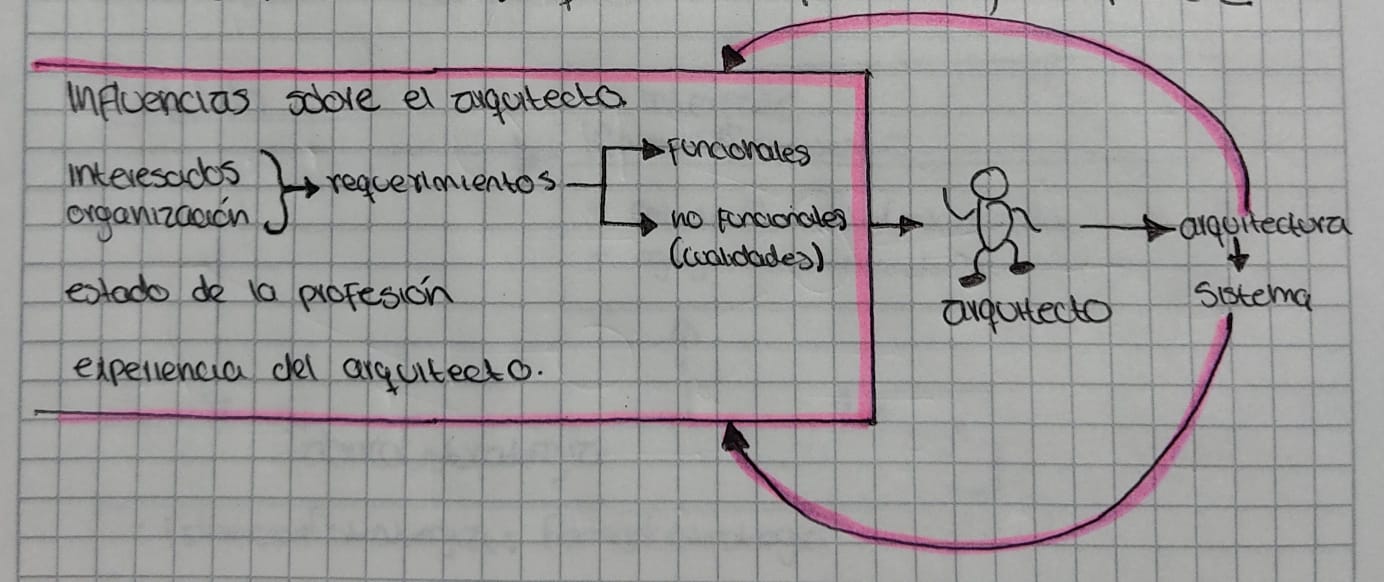


# 16. Herramientas para reusó de código JavaScript orientado a patrones de interacción

Hoy en día, hay muchas más páginas web disponibles en internet. Este artículo nos habla sobre la importancia de usar patrones de interacción, ya que estos ayudan a los diseñadores a crear de manera más efectiva. Al reimaginar soluciones que ya existen, se pueden aplicar a diferentes entornos o a problemas que surgen. Los desarrolladores de sitios web y aplicaciones están siempre buscando nuevas formas de diseñar páginas que ofrezcan soluciones efectivas a problemas que siguen apareciendo, utilizando métodos que ya han demostrado ser útiles.

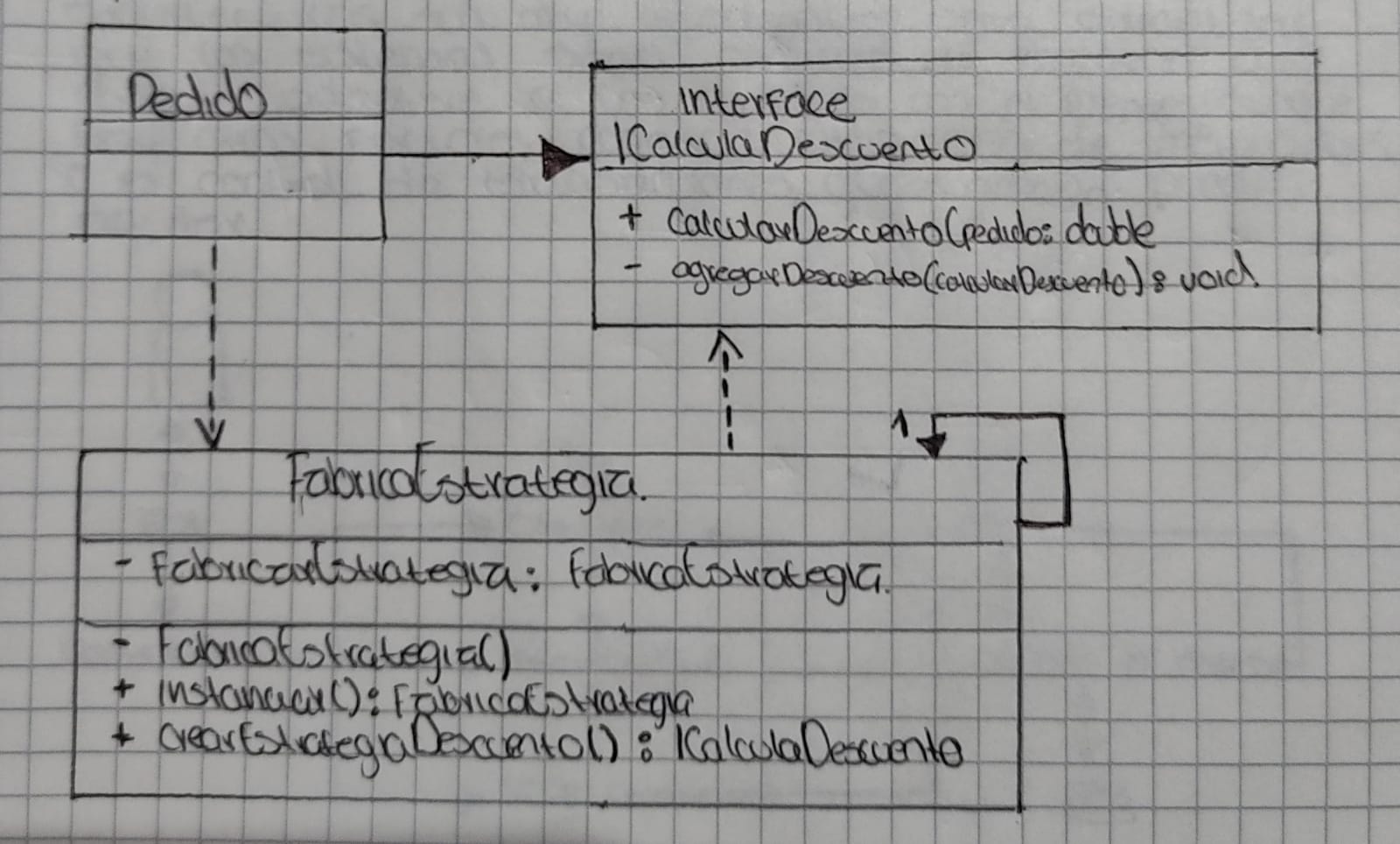
# 17. Introducción a la arquitectura de software

Este texto habla sobre por qué es importante ver la arquitectura de software como una etapa diferente y más general en el ciclo de vida del software. También menciona que es crucial tener en cuenta tanto los requisitos funcionales como los no funcionales al definir la arquitectura. Es esencial entender la diferencia entre arquitectura y diseño, ya que la arquitectura se ocupa del sistema de manera más general, mientras que el diseño se enfoca en los detalles de la implementación. Al centrarse en estilos arquitectónicos y conectores complejos, se puede formalizar esta idea y ayudar a crear sistemas que sean fuertes y adaptables.



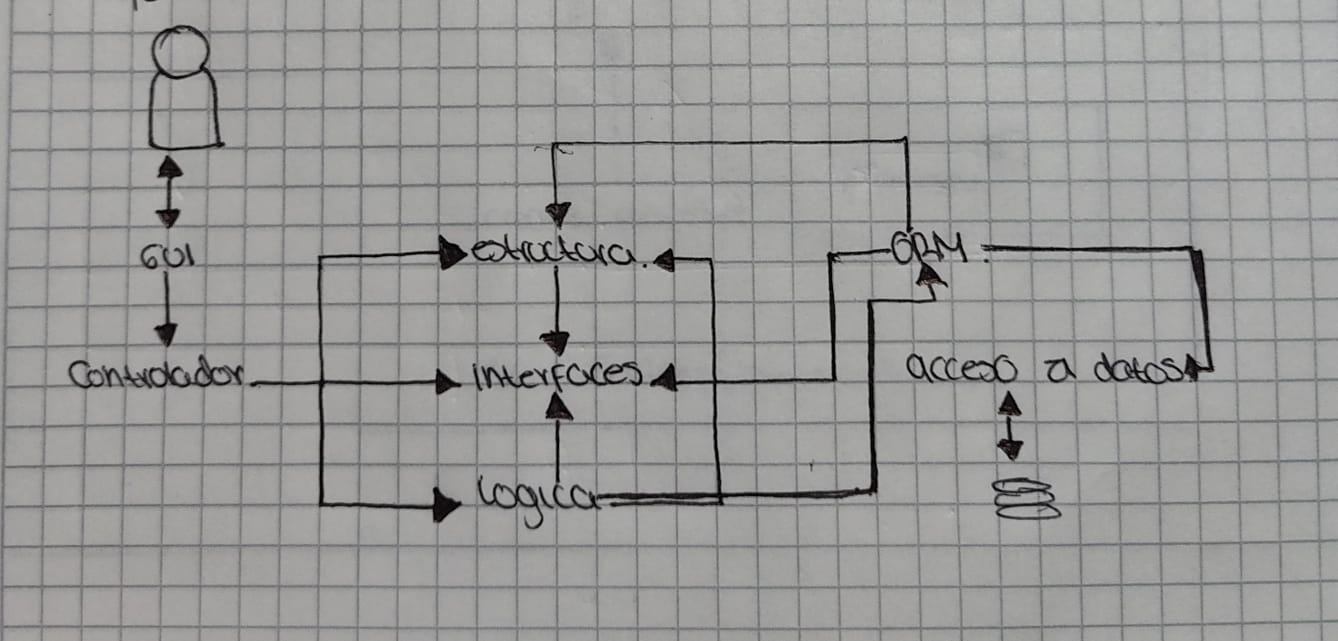
# 18. Aplicación de patrones de diseño para garantizar alta flexibilidad en el software

Para conseguir esta flexibilidad, se sugiere utilizar patrones de diseño como estrategias, como el patrón composite y el patrón de fábrica. Estos patrones ayudan a crear aplicaciones que se adaptan de forma eficiente a nuevos requerimientos sin necesitar mucho tiempo o recursos. En pocas palabras, al aplicar estos principios de diseño, se puede desarrollar software muy flexible, especialmente en sistemas que tienen muchas reglas de negocio que cambian con el tiempo. Esto permite que las empresas se mantengan competitivas en un entorno que está en constante cambio.



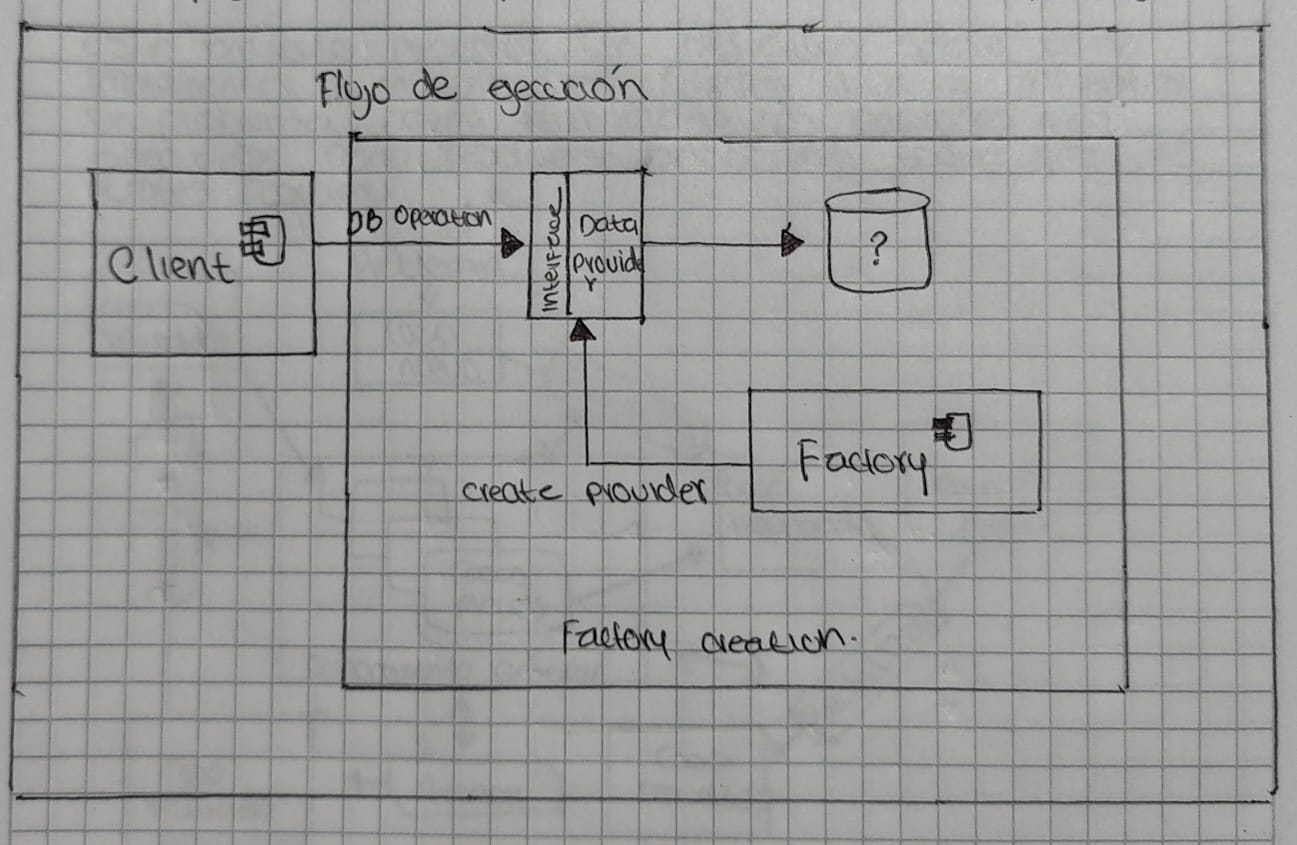
# 19. Arquitectura de software académico para la comprensión del desarrollo de software en capas

Se destaca que las arquitecturas bien estructuradas, como la arquitectura cliente-servidor, orientada a objetos, basada en componentes o en capas, ayudan a separar las funciones. Esto significa que si se hacen cambios en una parte del sistema, no afectarán a las demás. Esto hace que sea más fácil mantener y escalar el software, porque cualquier ajuste, ya sea en el negocio, en el control o en la presentación de las vistas, se puede hacer sin causar problemas en otras partes. La flexibilidad y escalabilidad que brindan estas arquitecturas son muy importantes para garantizar que los sistemas sean sencillos de mantener y se puedan adaptar a los cambios con el tiempo, aunque hay otros factores, como la automatización de mapeadores o el control de transacciones, que también pueden ayudar.

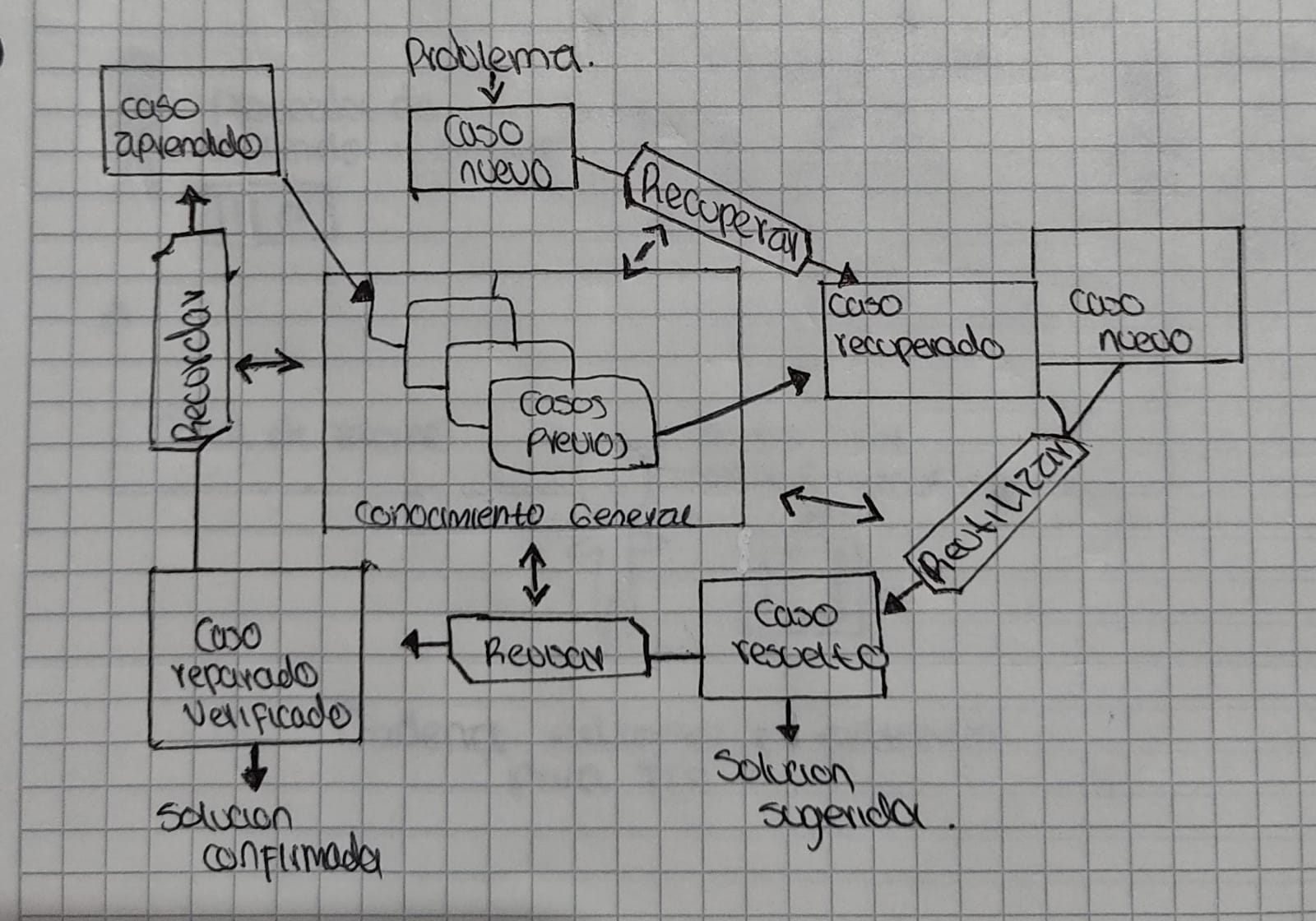


# 20. Introducción a los patrones de diseño

Los patrones de diseño tienen su origen en la arquitectura, tomando como referencia el trabajo de Christopher Alexander. Él decía que la arquitectura moderna había disminuido en calidad y describió los patrones como respuestas a problemas que se repiten en el entorno, que se pueden reutilizar y aplicar en diferentes situaciones. Alexander y su equipo crearon una colección de patrones arquitectónicos en \*A Pattern Language\*, que ofrecen soluciones útiles a problemas de diseño que son comunes en contextos particulares. Este mismo concepto se utiliza en los patrones de diseño de software.

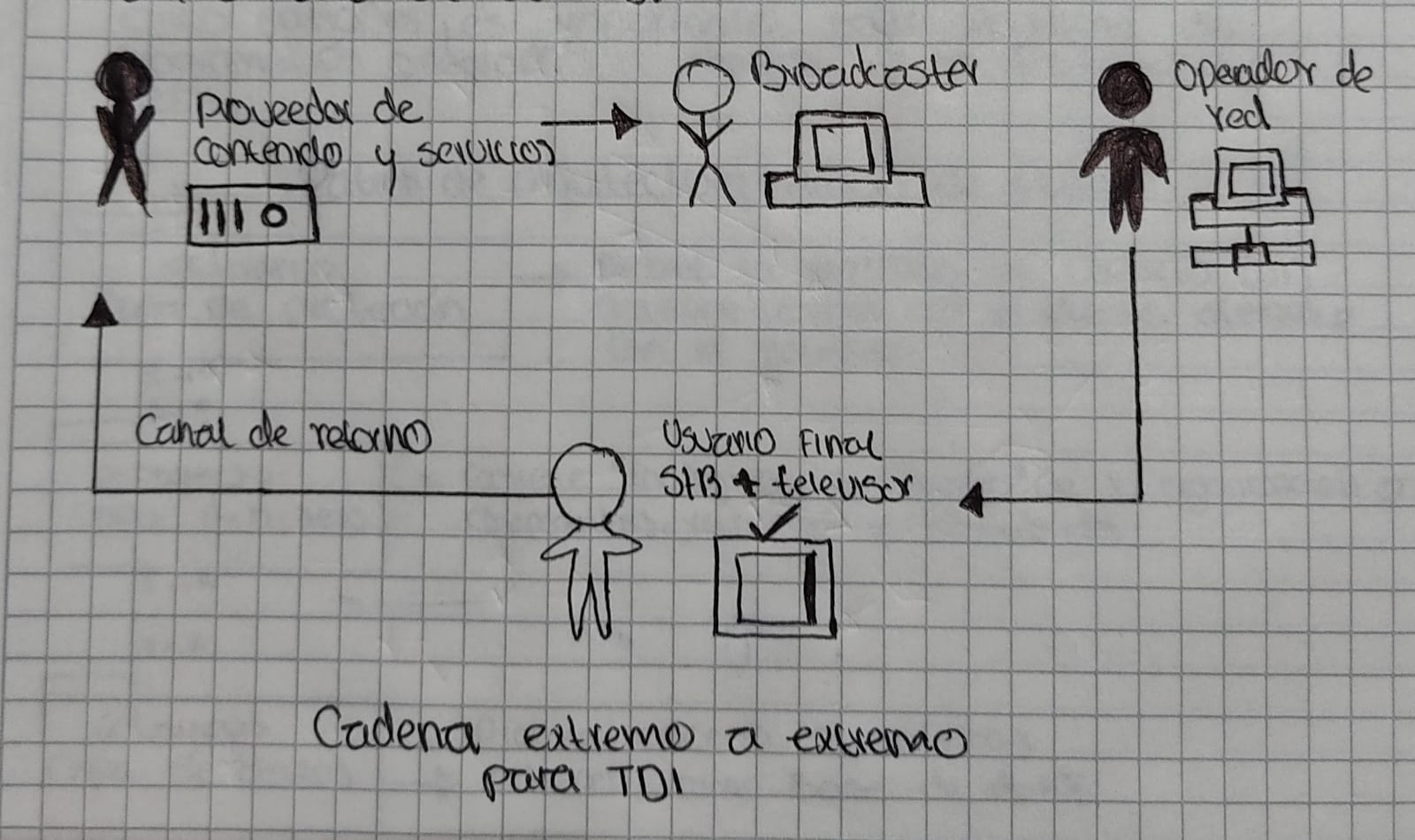


# 21. Modulo de recomendación de patrones de diseño EGPat

Se busca crear un camino que facilite la revisión de recursos educativos, así como la gestión y evaluación de patrones de diseño, y que también permita consultar estos recursos de manera más sencilla. Sin embargo, todavía hay problemas relacionados con el tiempo y la complejidad de las consultas. Aunque hay patrones disponibles en una fuente local, encontrar cuál o cuáles son los más apropiados para el recurso educativo que se está creando sigue siendo un proceso complicado. Este artículo presenta los detalles sobre cómo se implementó un módulo para EGPat que, al enfrentar un problema, puede recomendar los patrones más adecuados para resolverlo, analizando todas las fuentes posibles.

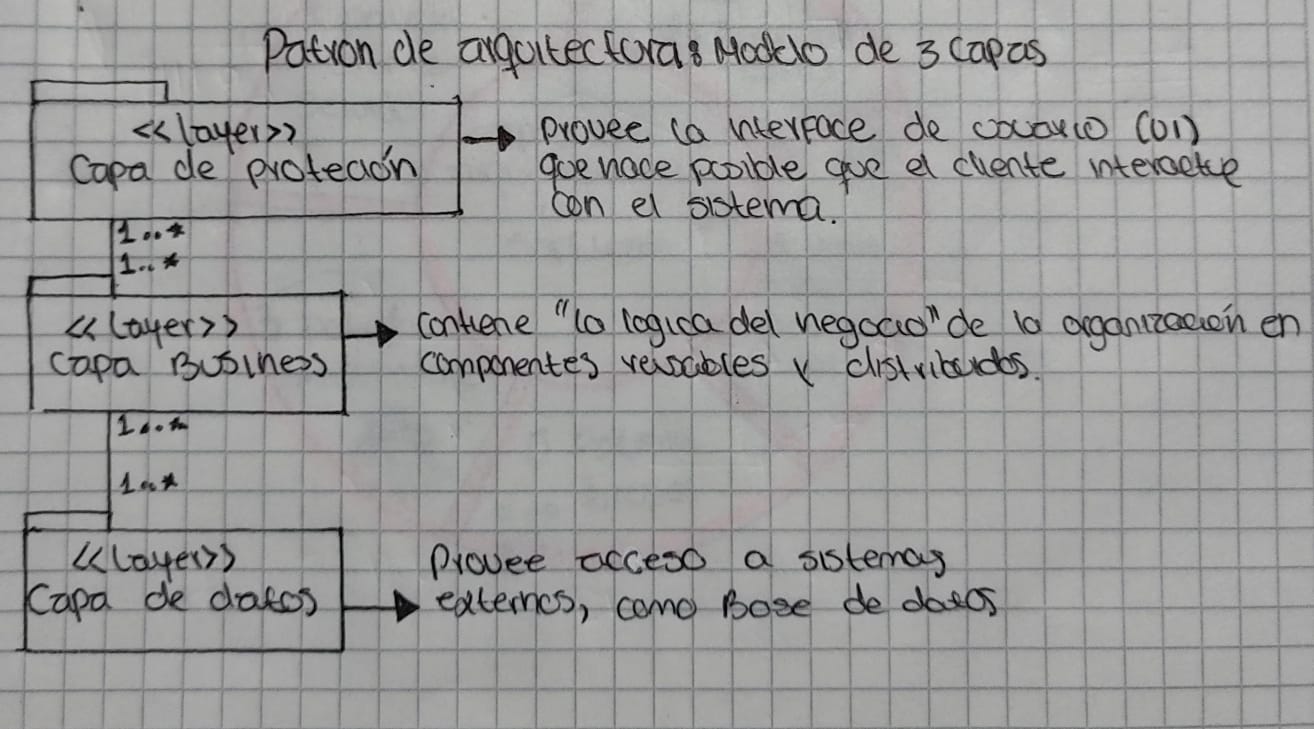
# 22. Arquitectura de software para el soporte de comunidades académicas virtuales en ambientes de televisión digital interactiva

El avance de la tecnología ha transformado Internet y la televisión digital interactiva (TDi), ofreciendo nuevas formas de aprender y colaborar gracias a herramientas como foros y redes sociales. Se sugiere una estructura que utiliza servicios REST-JSON para apoyar aplicaciones en TDi, especialmente en la educación, aprovechando las ventajas de la Web 2.0 y la interactividad. Iniciativas como EDiTV y T-Maestro investigan cómo usar tecnologías como IPTV y TDi en el aprendizaje, mejorando la educación y superando las limitaciones del e-learning tradicional. Este estudio presenta una solución que aplica minería de texto y razonamiento basado en casos para sugerir patrones de diseño en aplicaciones interactivas dentro del ámbito de la TDi.



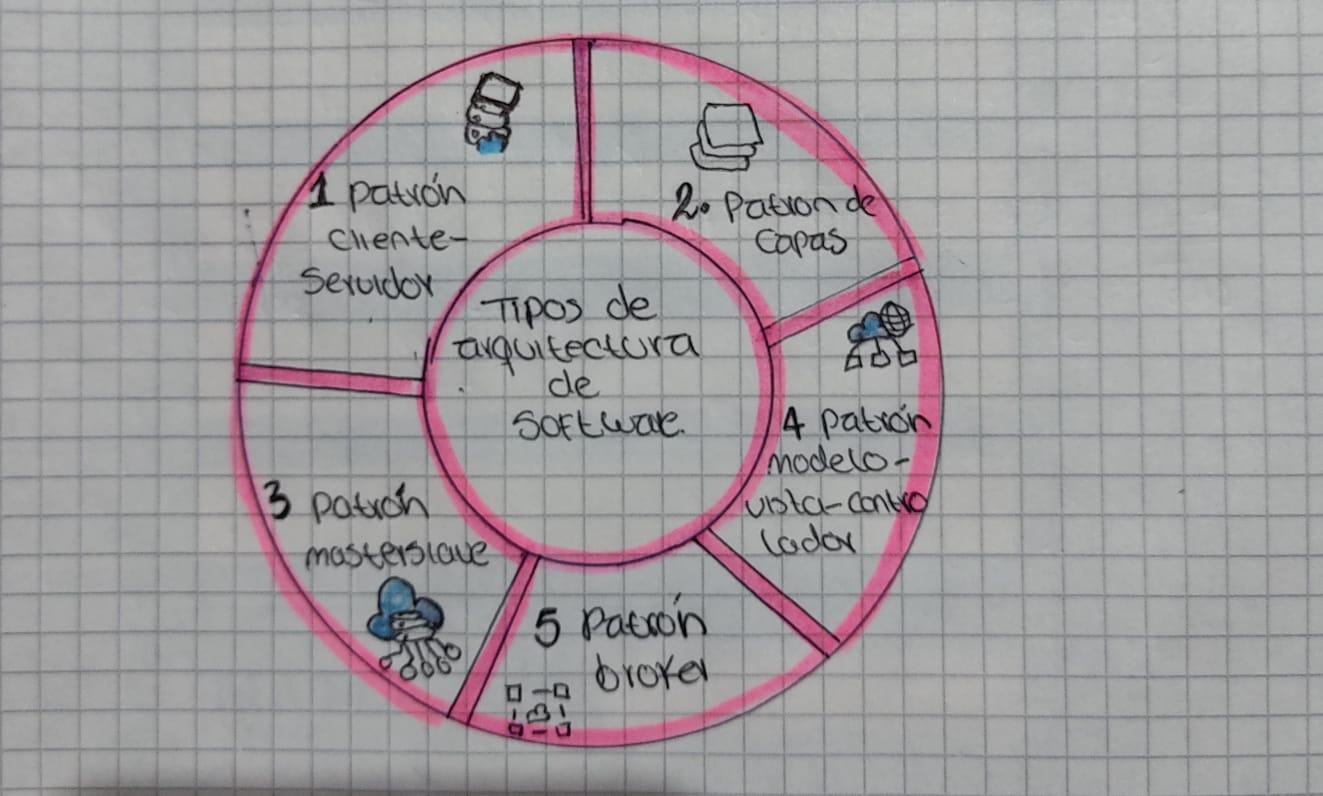
# 23. Desarrollo de sistemas de software con patrones de diseño orientado a objetos

El uso de patrones de diseño en el desarrollo de software ayuda a reutilizar estructuras lógicas y aumenta la productividad. La programación orientada a objetos (POO) es fundamental para esta reutilización, ya que organiza el diseño en torno a componentes que se pueden usar varias veces. Los patrones de diseño hacen que sea más fácil construir sistemas, disminuyen los costos y mejoran la calidad del software. Ejemplos como los patrones \*Informador\*, \*Sensor\* y \*de 3 capas\* permiten que el sistema sea más modular y flexible, además de ayudar a manejar la complejidad en partes más simples. Estos patrones también ayudan a que el sistema sea escalable y mejoran la comunicación entre los desarrolladores y los analistas. Para implementar estos patrones, es importante usar lenguajes de programación orientados a objetos como .NET, Java o Delphi.



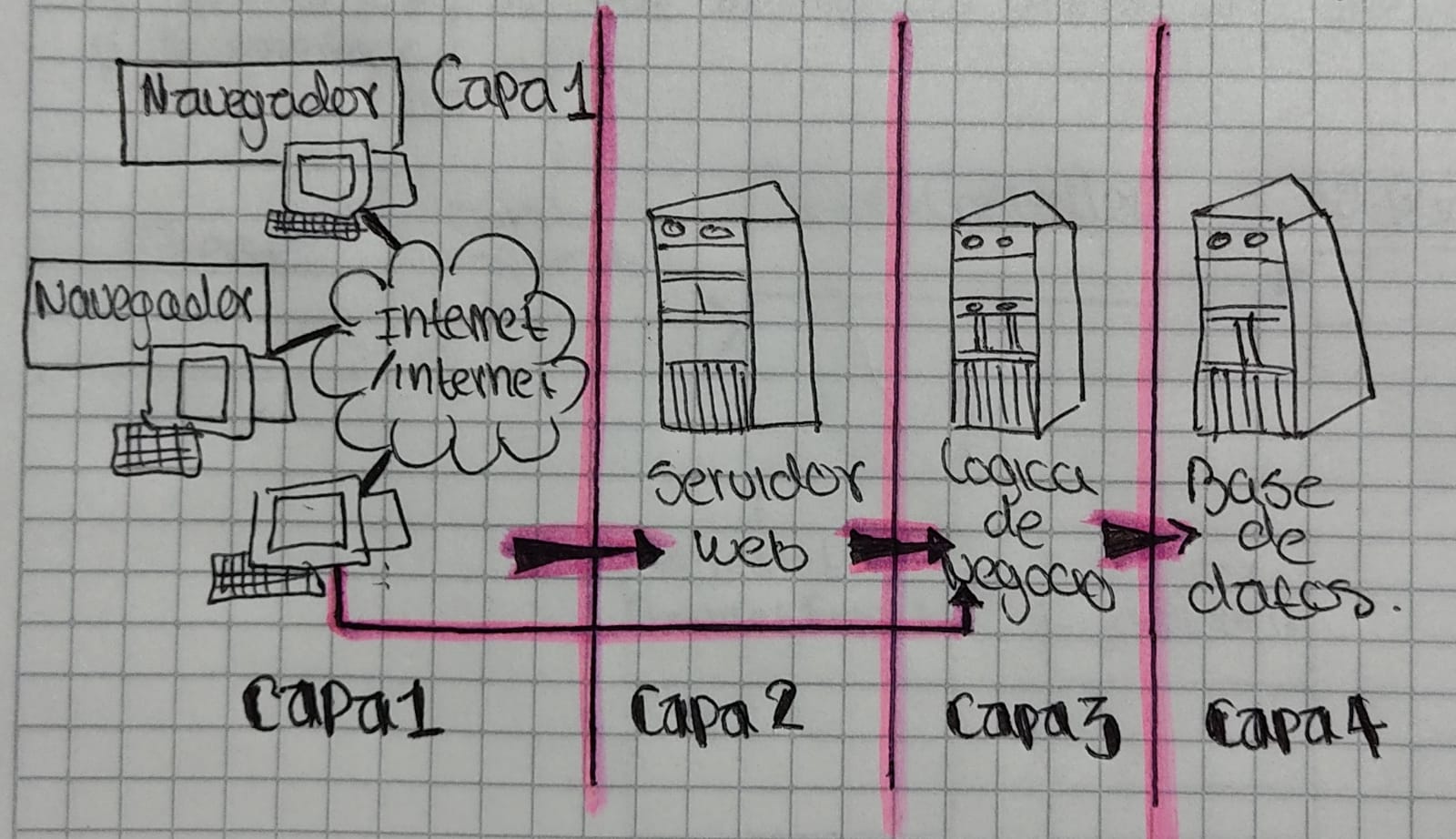
# 24. Atributos de calidad y arquitectura de software

En este minicurso de 4 horas, se explorarán los conceptos básicos de la arquitectura de software, los atributos de calidad y cómo crear un modelo de arquitectura que satisfaga estos requisitos. Se enfatizará la importancia de definir la arquitectura desde diferentes ángulos para resaltar las distintas cualidades del sistema y tomar decisiones de diseño que sean equilibradas, asegurando que se logre una cualidad sin comprometer demasiado a las demás. Diseñar una arquitectura que cumpla con los requisitos tanto funcionales como de calidad es un desafío complicado, ya que las decisiones que se tomen en esta etapa afectarán de manera significativa las características del sistema final. Se presentará un método que se basa en analizar y comprobar que la arquitectura propuesta cumpla con las cualidades deseadas.



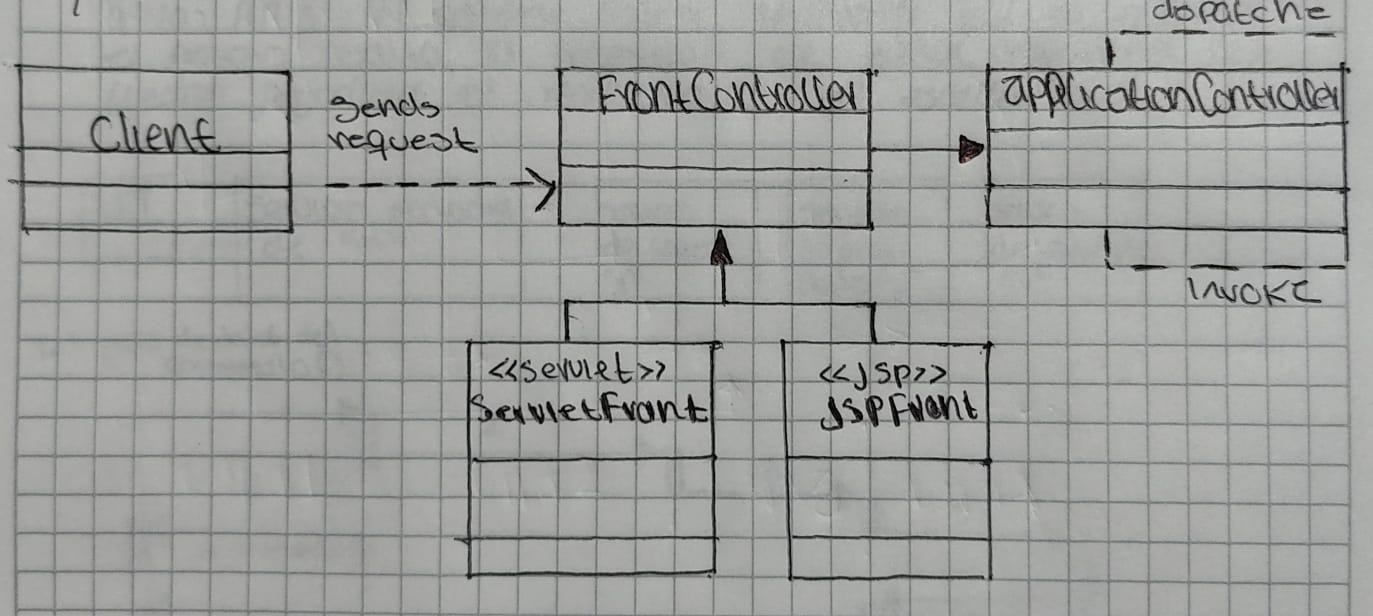
# 25. Arquitectura de software en el proceso de desarrollo ágil: una perspectiva basada en requisitos significantes para la arquitectura

Las Metodologías Ágiles son conocidas por ser flexibles y fomentar la colaboración, pero a veces pueden complicar la definición precisa de los requisitos arquitectónicos al comienzo de un proyecto. Por otro lado, la Arquitectura de Software se enfoca en requisitos no funcionales que son muy importantes. Para solucionar este problema, se creó el concepto de "Arquitectura Ágil", que intenta combinar los requisitos arquitectónicos con el enfoque ágil, destacando los "Requisitos Importantes para la Arquitectura".



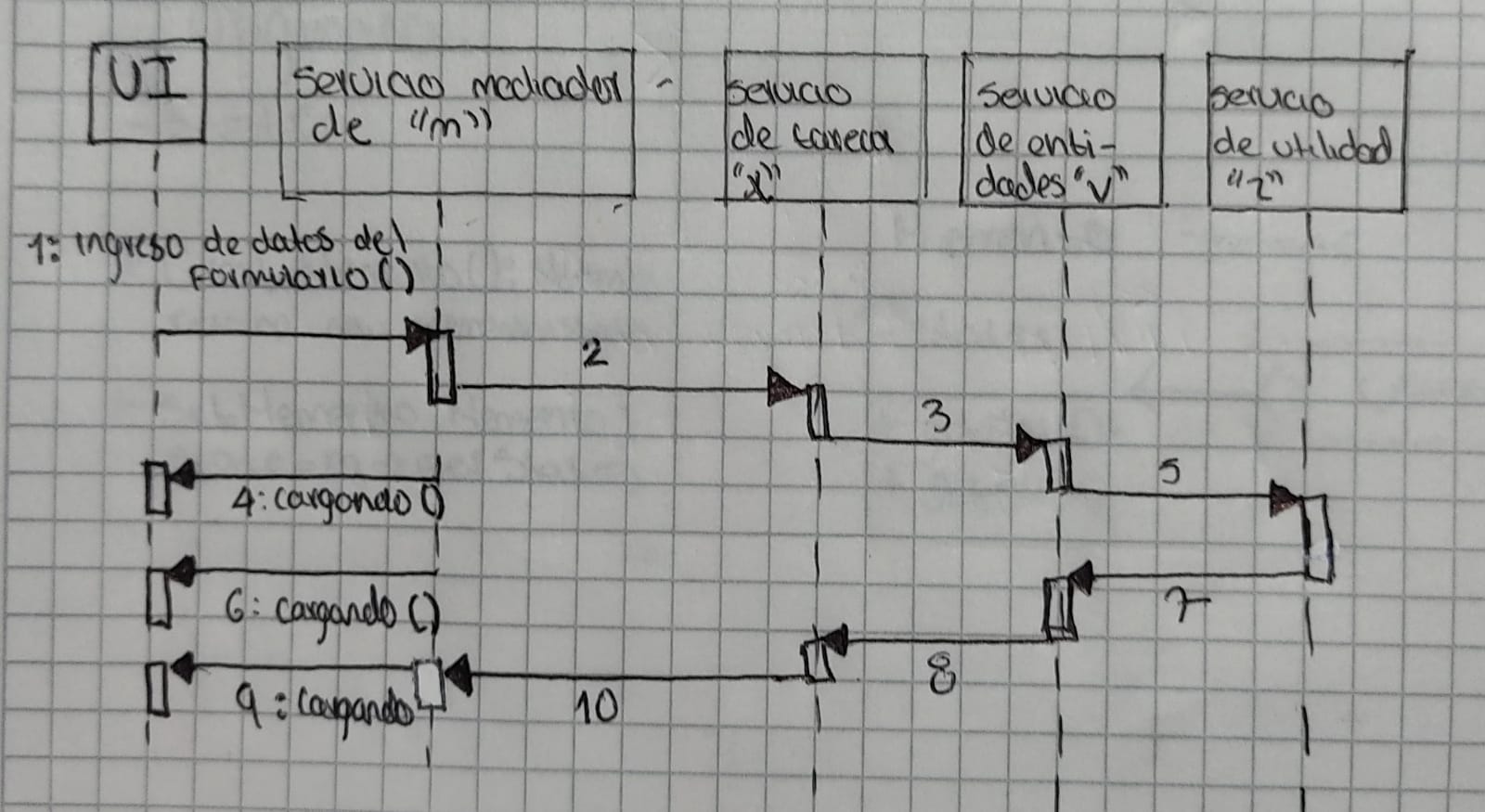
# 26. Análisis comparativos de patrones de diseño de software

Los patrones de diseño son métodos que se utilizan a menudo para solucionar problemas que aparecen frecuentemente en la creación de software. Algunos ejemplos son el Método Plantilla, MVC, MVP, Controlador Frontal y MVVM. Estos patrones hacen que sea más fácil organizar las aplicaciones y usar el código en diferentes partes. También ayudan a mantener separada la lógica de la interfaz, lo que puede hacer que el desarrollo sea más eficiente, aunque a veces pueden complicar las cosas. Además, los patrones garantizan que el código sea válido y reutilizable, lo que mejora la forma en que se documenta y se mantiene.



# 27. Identificación y clasificación de patrones de diseño de servicios web para mejorar Qos

Los servicios web son muy importantes para conectar diferentes aplicaciones en las organizaciones de hoy en día. La arquitectura orientada a servicios (SOA) es un modelo que se adapta bien y se utiliza a menudo con servicios web. Esta tesis sugiere que se puede mejorar la calidad del servicio (QoS) de estos servicios usando patrones de diseño que son bien conocidos. Se presenta una fórmula para medir cómo estos patrones afectan la QoS, tanto de manera individual como en conjunto. La investigación también analiza ontologías de servicios web, principios de diseño y antipatrones. Además, se aplica esta metodología a las APIs del sistema SABRE, mostrando que es efectiva. El trabajo resalta la necesidad de evitar antipatrones que puedan perjudicar el rendimiento y la escalabilidad.

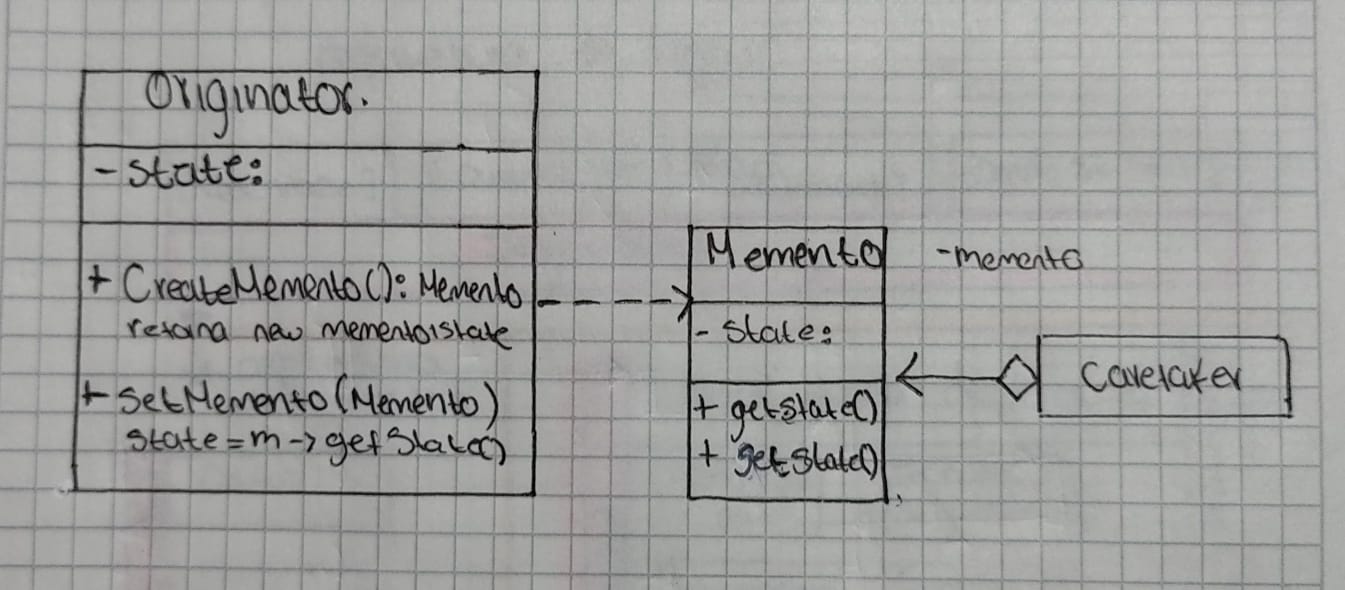


# 28. Especificación del patrón de diseño memento atreves de un perfil UML

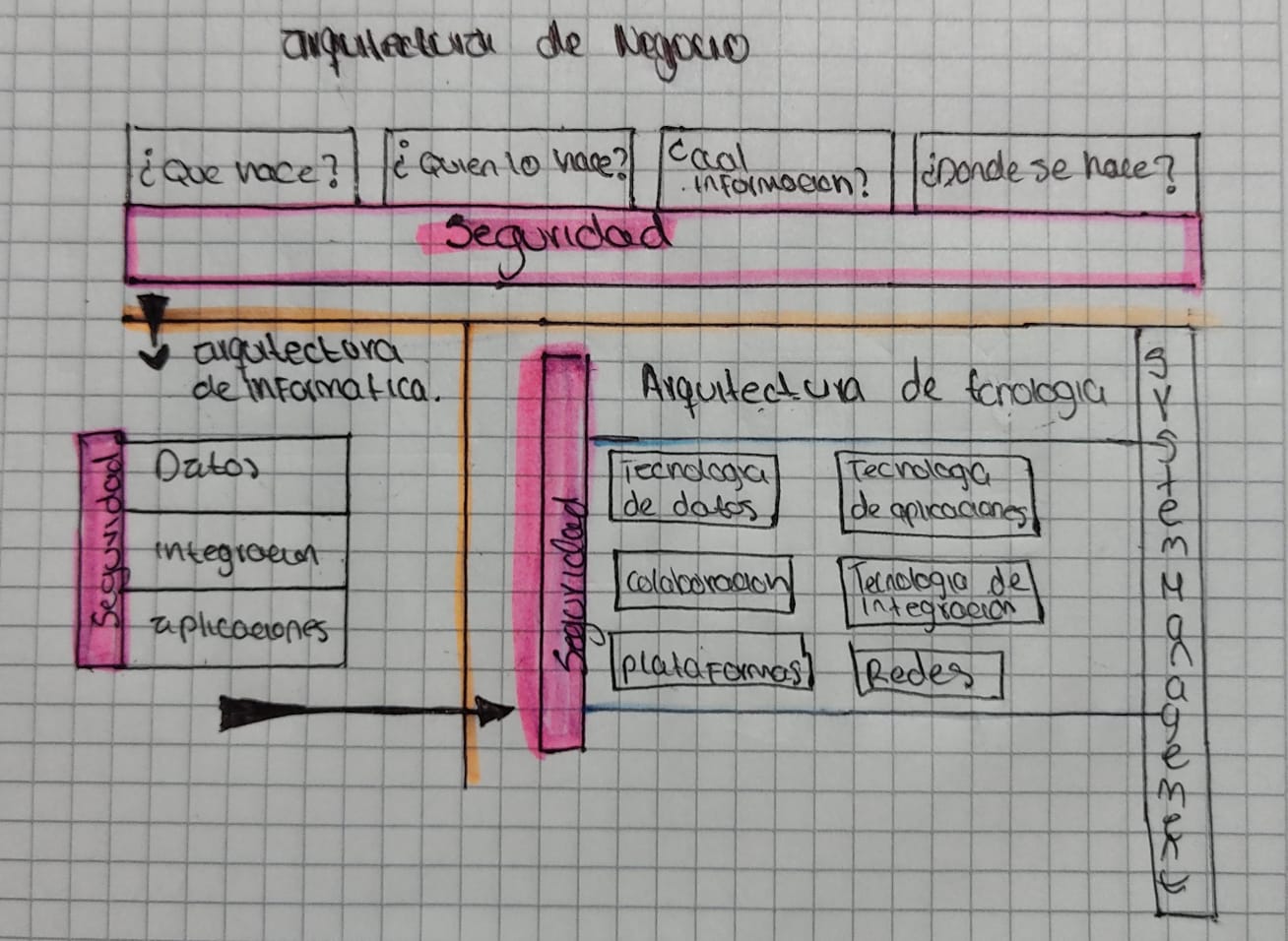
Los patrones de diseño son estrategias basadas en la experiencia de expertos que ayudan a documentar, compartir conocimientos y crear una terminología común en el desarrollo de software.

El catálogo más conocido es el “Gang of Four” (GoF), que clasifica los patrones en tres tipos: Creacionales, Estructurales y de Comportamiento.

Para especificar y validar estos patrones, se propone la Arquitectura de Perfiles UML de Patrones de Diseño (APPD), que organiza los patrones en tres niveles, asegurando coherencia entre diagramas estáticos y dinámicos.



# 29. Documentación y análisis de los principales frameworks de arquitectura de software en aplicaciones empresariales

El avance tecnológico ha impulsado nuevas metodologías y herramientas en empresas, destacando la importancia de la arquitectura de software. Se analizan los frameworks más utilizados en el desarrollo de aplicaciones empresariales, resaltando su aplicabilidad. Una buena arquitectura alinea la estrategia de negocio con la infraestructura tecnológica. La elección correcta del framework es clave. Es fundamental combinar correctamente los componentes de la arquitectura, contar con una buena gestión y mantener una comunicación constante para el éxito en la implementación.

# 30. Implementación de una arquitectura de software guiada por el dominio

La idea principal es dividir la lógica del negocio de los aspectos tecnológicos para que el software sea más flexible, fácil de mantener y pueda evolucionar mejor. DDD proporciona un método organizado para crear modelos de dominio que reflejen la realidad, mientras que la arquitectura hexagonal ayuda a separar el núcleo del negocio de la tecnología, lo que permite hacer cambios en las herramientas sin afectar el dominio. Este artículo sugiere un enfoque innovador para el desarrollo de software, destacando la necesidad de separar la lógica del negocio de las tecnologías que se utilizan. Al aplicar DDD y la arquitectura hexagonal, se pretende construir sistemas más fuertes y sencillos de mantener, que se adapten a los cambios del mercado y la tecnología sin dañar el núcleo del negocio.

