

## Arquitectura de Software basada en microservicio para desarrollo de aplicaciones web

### Resumen

El desarrollo de software en la coordinación general de tecnologías de la información y la comunicación (CGTIC) de la Asamblea Nacional del Ecuador (ANE). Se ha basado en una arquitectura monolítica. Este enfoque empaquetó toda la funcionalidad en una única unidad ejecutable, siguiendo las tendencias impuestas por el lenguaje de programación utilizado y la experiencia del área de desarrollo. Si bien este modelo ha sido funcional, ha generado desafíos significativos en términos de funcionamiento, escalabilidad y entrega de software, especialmente a medida que las necesidades y los sistemas han evolucionado.

### Reflexión

El uso de arquitectura monolíticas presenta problemas de sistema complejos, dificultando mantenimiento y escalabilidad. Esto resalta la necesidad de adoptar arquitecturas modernas, como los microservicios, que dividen aplicaciones en componentes independientes.

Esta transición mejora el mantenimiento, como la escalabilidad, permitiendo un trabajo más ágil y eficiente para los equipos de desarrollo. Las investigaciones actuales pueden transformar la forma en que la CGTIC aborda el desarrollo.

## Modelado y verificación de entornos de diseño de arquitectura de Software para entornos de computación en la nube.

### Resumen

Presenta un enfoque integral para el diseño de arquitectura de software, especialmente para aplicaciones web. Se destaca la importancia de utilizar arquitecturas genéricas que proporcionen una estructura común para resolver problemas específicos, lo que ayuda a los arquitectos a evitar conflictos derivados de la falta de experiencia. El trabajo introduce un metamodelo de componentes arquitectónicos que identifica elementos clave en el diseño, y se complementa con una herramienta gráfica para la instalación de estos componentes.

### Reflexión

El enfoque presentado en el documento es altamente relevante en el contexto de la computación en la nube, donde la complejidad de las aplicaciones web está en constante aumento. La propuesta de un entorno de diseño integral que combina un metamodelo con herramientas de verificación es una contribución valiosa, ya que no facilita el trabajo de los arquitectos de software, sino que también promueve la creación de aplicaciones más robustas y de calidad. Además, la atención a la experiencia del arquitecto resalta la necesidad de formación continua en un campo que evoluciona rápidamente. En general, este trabajo puede servir como una guía útil para profesionales y académicos interesados en mejorar sus prácticas de diseño en entornos de computación en la nube.

# Análisis Comparativo de patrones de diseño de Software

## Resumen

En resumen, la evolución de las prácticas de desarrollo de software ha llevado a la implementación de pruebas durante el desarrollo para identificar problemas tempranamente y reducir costos de mantenimiento. Los patrones de diseño son herramientas esenciales que ayudan a estructurar y a organizar aplicaciones, resolviendo problemas comunes con soluciones probadas. No son implementación directa de código, sino modelos conceptuales que guían a los desarrolladores en la resolución de problemas específicos, mejorando la calidad del software mediante el diseño orientado a objetos.

## Reflexión

Los patrones de diseño son esenciales para el desarrollo de software moderno, ya que permiten la reutilización de soluciones probadas, reduciendo errores y mejorando la eficiencia. Sin embargo, al seleccionar el patrón correcto puede ser un desafío para desarrolladores menos experimentados debido a la amplia variedad disponible. Patrones como MVC, MVVM y MUVM ofrecen enfoques específicos para la interacción entre componentes, lo que hace que el software sea más modular y mantenable. Es crucial evaluar cada patrón en función de las necesidades del proyecto, y la documentación es fundamental para educar desarrolladores y garantizar un uso efectivo de estas herramientas.

## Patrones de diseño GOF (The gang of four) en el contexto de patrones y procesos de desarrollo de aplicaciones orientadas a la web

### Resumen

El estudio analiza los patrones de diseño Gang of Four (GOF) presentados en el libro Design Patterns: Elements of Reusable Object-Oriented Software. Estos patrones, clasificados en creacionales, estructurales y de comportamientos, brindan soluciones a problemas comunes en el diseño de software orientado a objetos. Se resalta la importancia de estos patrones como una buena práctica para mejorar la calidad de software, aunque también se mencionan desafíos como la falta de ejemplos didácticos en el catálogo original.

### Reflexión

Los patrones de diseño GOF han sido clave en el desarrollo de software al proporcionar soluciones reutilizables para problemas comunes, pero su alta abstracción dificulta su implementación, especialmente para desarrolladores novatos. Este estudio destaca la importancia de documentación accesible y ejemplos prácticos alineados con los modelos originales es esenciales para poder seleccionar los patrones con cuidado según el problema y contexto, evitando su uso innecesario para no introducir complejidad adicional.

# Arquitectura de software esquemas y servicios.

D - M - A

## Resumen.

Las aplicaciones empresariales actuales deben cumplir con requisitos que antes eran poco comunes, como la independencia, como sistemas operativos y base de datos, el acceso desde ubicaciones distante, la interacción con otros sistemas existentes y el manejo de grandes volúmenes de interacciones simultáneas. Esto ha generado una transición de arquitecturas monolíticas y centralizadas a entornos distribuidos y heterogéneos, lo que ha incrementado la complejidad estructural de las aplicaciones.

## Preflexion.

La evolución hacia arquitecturas distribuidas y heterogéneas refleja el cambio de las necesidades del mercado empresarial que exige aplicaciones flexibles escalables y accesibles. Esta complejidad, aunque desafiante, ofrece oportunidades para crear soluciones innovadoras que aprovechan tecnologías emergentes. Sin embargo, el desafío para los arquitectos de software radica en gestionar esta complejidad sin sacrificar la eficiencia de desarrollo, especialmente cuando los recursos con el tiempo son limitados.

## Arquitectura de software para el desarrollo de videojuegos sobre el motor de juego Unity 3d.

### Resumen.

El uso de tecnologías de información y comunicación (TIC) ha impulsado el conocimiento y crecimiento de la industria en los videojuegos. Entre las disciplinas involucradas en su desarrollo se encuentra la programación, diseño y marketing. Los motores de videojuegos como Unity 3d simplifican y automatizan tareas complejas. Sin embargo en la UCI se detectaron problemas como la falta de organización y reutilización limitada de componentes de videojuegos creados con Unity 3d. Para resolver esto, se diseñó una arquitectura de software que organiza las características fundamentales y funcionales básicas de los videojuegos.

### Reflexión

El desarrollo de videojuegos refleja cómo la tecnología y la creatividad pueden converger para generar soluciones innovadoras. Sin embargo, los retos asociados a la organización y estructuración del proceso evidencian la importancia de contar con herramientas y metodologías sólidas que optimicen los recursos y promueven la calidad del producto final. El diseño de arquitecturas de software específicas para videojuegos no solo facilita la reutilización de componentes y la eficiencia en el desarrollo, sino que también fomenta la sostenibilidad de proyectos a largo plazo.

Arquitectura de software orientada a la creación de micromundos para la enseñanza y el aprendizaje.

### Resumen.

Los procesos de enseñanza-aprendizaje enfrentan diversos desafíos que dificultan el manejo y acceso de las tecnologías de información limitan tanto a estudiantes como docentes mientras que los currículos rígidos restriñen la creatividad e impiden una integración efectiva de las tics. Muchas instituciones educativas no cuentan con recursos tecnológicos suficientes o no lo utilizan adecuadamente, desaprovechando el potencial de los estudiantes y afectando su formación.

### Reflexión:

La educación enfrenta desafíos importantes en un mundo cada vez más digital. La falta de la integración de las tics en los procesos de enseñanza limita la capacidad de los estudiantes para adaptarse a las demandas actuales. Es necesario reestructurar los modelos educativos para fomentar habilidades tecnológicas, creatividad y autogestión, asegurando que todos los factores tengan acceso equitativo a los recursos. Solo a través de cambios sistemáticos, que priorice la inclusión tecnológica, podremos fomentar generaciones preparadas para enfrentar los retos del Siglo XXI.

# Uso de patrones de software: Un caso práctico.

D M E A

## Resumen

Los patrones de diseño son herramientas fundamentales en la ingeniería de software, ya que brindan soluciones comprobadas a problemas comunes y facilitan el desarrollo eficiente y estructurado en aplicaciones. Entre sus principales beneficios destacan la reutilización de código, la reducción de la complejidad, el decoupling de componentes y la mejora del mantenimiento de software. Los resultados también nos demuestran como la implementación de patrones puede mejorar significativamente la calidad del software aunque debe considerarse su impacto en la complejidad del código y de las pruebas.

## P reflexión:

Los patrones de diseño son un pilar esencial en la formación de ingenieros de software y científicos de la computación. Incorporar su enseñanza desde etapas tempranas en las carreras universitarias es crucial, ya que promueve el desarrollo de las habilidades prácticas que trascienden los métodos tradicionales de enseñanza.

Aunque el uso de patrones puede aumentar la complejidad inicial del diseño, los beneficios a largo plazo, como la facilidad de mantenimiento, escalabilidad y flexibilidad, justifican ampliamente su adopción.

Desarrollo de una herramienta para el aprendizaje de patrones de diseño de Software.

## Resumen.

Los patrones de diseño ofrecen soluciones probadas y eficaces a problemas recurrentes en programación, facilitando la creación de código limpio y fácil de mantener. Estos patrones no sólo resuelven problemas técnicos, sino que también mejoran la comunicación entre los desarrolladores, ya que el conocimiento de un patrón permite comprender rápidamente su estructura y propósito.

El prototipo desarrollado puede ser la base para futuras mejoras y evoluciones por parte de otros estudiantes.

## Reflexión.

El desarrollo de una herramienta para aprender patrones de diseño en un paso significativo en la formación de futuros programadores, ya que permite que los principiantes comprendan la importancia de un diseño estructurado desde el inicio. El aprendizaje de tecnologías como Spring y el desarrollo front-end, inicialmente desconocidos para el autor muestra cómo la resolución de problemas a través de la práctica y la investigación puede superar barreras y expandir habilidades. Este prototipo tiene un gran potencial para evolucionar y servir como base para futuras interacciones que mejoren su funcionalidad y la experiencia del usuario.

## Resumen

La arquitectura de software se enfoca en diseñar y organizar sistemas, incluyendo componentes y sus relaciones entre ellos. Las arquitecturas de referencia (AR) son marcos estandares que mejoran la reutilización y la adaptabilidad, integrando conceptos clave para prácticas más eficientes en la industria del software. Estas integran conceptos como configuraciones, restricciones, estilos arquitectónicos y servicios, promoviendo prácticas más eficientes en la industria del software.

## Reflexión

La arquitectura de software es esencial para conectar las necesidades específicas del proyecto y utilizar arquitecturas de referencia para reducir costos y tiempos de desarrollo. Sin embargo, se enfrentan desafíos como la falta de mecanismo para la evolución dinámica de los sistemas y la reutilización efectiva en algunos lenguajes. Este enfoque fomenta la colaboración interdisciplinaria, lo que resulta vital para alcanzar los objetivos del software y abordar sus desafíos en un entorno constante de cambio.

## Atributos de calidad y arquitectura de software.

### Resumen.

En este artículo logramos construir una arquitectura de software afectando la cual requiere balancear múltiples atributos de calidad (como seguridad, mantenibilidad, portabilidad y rendimiento) sin comprometer desmedidamente a otros. las decisiones tomadas durante el diseño tienen un impacto en el resultado final.

Aplicar una metodología que considere de diferentes perspectivas, documente de manera integral y priorice las necesidades de los stakeholders lo cual permite crear arquitecturas que sean tanto funcionales como la calidad, aunque no garantiza estos atributos automáticamente.

### Reflexión

la arquitectura de Software no es un solo conjunto de decisiones técnicas si no un proceso estratégico que impacta directamente al éxito y la calidad de un sistema. Enfrenta el desafío de equilibrar como rendimiento, seguridad y mantenibilidad que a menudo entran al conflicto entre si.

El uso de vistas arquitectónicas y estilos bien definidos no solo facilita la comunicación con los stakeholders, si no que tambien actuan como una guia para tomar decisiones informadas y mantener la coherencia en el desarrollo.

# Herramienta para reuso de código Java Script orientado a patrones de interacción

## Resumen.

El desarrollo de sistemas informáticos enfrenta diversos problemas desde el análisis hasta la implementación. Herramientas como eBuilder, Visual Basic y Delphi han emergido para facilitar la creación de interfaces de usuario. La usabilidad es fundamental para la satisfacción del usuario. La reutilización de patrones acelera el desarrollo y mejora la calidad del software. Resmu es un ejemplo de herramienta que genera componentes web reutilizables en Java Scripts, optimizando la creación de interfaces gráficas personalizadas.

## Reflexión

El uso de patrones de interacciones en el desarrollo de interfaces y aplicaciones web mejora la eficiencia y calidad de los productos. Al adoptar patrones reutilizables, los diseñadores evitan reinventar soluciones y aprovechan mejores prácticas. Herramientas como Heus me facilita la personalización de componentes manteniendo la usabilidad. La reutilización de código y patrones son esenciales para optimizar el desarrollo y crear productos más robustos y accesibles a la entrega final.

## Implementación de una arquitectura guiada por el dominio.

### Resumen.

Presento un enfoque de implementación de una arquitectura de software guiadas por el dominio, enfatizando el diseño dirigido por el dominio (DDN). Se propone transformar una arquitectura de software típica en una arquitectura hexagonal, centrada en el dominio de negocio. Esto permite separar la complejidad de negocio de la lógica técnica, promoviendo una mejor comunicación entre expertos en dominios y desarrolladores. La investigación concluye que esta transformando y mejorando la mantenibilidad y escalabilidad del software.

### Reflexión

El enfoque presentado en este artículo es muy relevante en el contexto actual del desarrollo de software, donde la complejidad y la rapidez son cambios constantes. La adopción de una arquitectura guiada por el dominio no solo facilita una mejor alineación con los objetivos de negocio, si no que también promueve una colaboración más efectiva entre diferentes partes interesadas. La transformación hacia una arquitectura hexagonal, que favorece la independencia de tecnologías, es una estrategia inteligente para asegurar la adaptabilidad de futuros cambios y mejorar la calidad de software.

## Una teoría para el diseño de software.

### Resumen.

El diseño de software es esencial porque, al dividir un sistema en elementos que interactúan entre sí, se facilita su desarrollo y mantenimiento. Este enfoque permite asignar funciones a cada componente, establecer sus relaciones y describir su estructura lo que reduce un costo y la complejidad a lo largo del ciclo de vida del software. El diseño de software se vuelve crucial para reducir los costos de mantenimiento, ya que anticipa y facilita la incorporación de cambios sin comprometer la integridad del sistema.

### Reflexión

La reflexión sobre el diseño de software nos muestra la importancia de planificar y estructurar adecuadamente un sistema desde el principio. Aunque pueda parecer tentador lanzarse directamente a la programación, sin un diseño previo, los costos y problemas surgen a lo largo del ciclo de vida del software puede ser mucho mayor. En definitiva, el diseño adecuado no solo es cuestión técnica, sino estratégica económica que puede hacer diferencia entre un sistema costoso de mantener y uno que se quede y adapte con facilidad a las necesidades futuras.

# D.M.A

## Desarrollo de una arquitectura de software para el robot móvil Lázaro.

### Resumen.

Este artículo presenta una innovadora estructura de software diseñada para optimizar el control y la operación del robot Lázaro. La arquitectura se compone de tres niveles que permiten la gestión eficiente de los actuadores y monitoreo de sensores, utilizando librerías implementadas en C#. Se incorpora diversos sensores, como sonares y un telémetro láser, para la detección de obstáculos y la medición de fuerzas. La comunicación entre el robot y un computador remoto a través de módulos XBee®, permitiendo un control tanto autónomo como teleoperado.

### Reflexión

La propuesta de una arquitectura de software para el robot Lázaro es un avance significativo en el campo de la robótica móvil. La integración de múltiples sensores y la capacidad de control remoto ofrecen un enfoque versátil y adaptable a diferentes entornos. Además, la combinación de arquitecturas deliberativas y reactivas junto con el uso de inteligencia artificial, demuestra un entendimiento profundo de los desafíos que enfrentan los robots en situaciones dinámicas. En general, este trabajo, no solo contribuye al desarrollo de robot Lázaro, sino que también sienta las bases para futuras investigaciones en arquitecturas de software para robots móviles.

## Artículo de software académico para la compresión del desarrollo de software en capas.

### Resumen

Este artículo expone un modelo académico de arquitectura de software basado en un enfoque de desarrollo en capas, destacando su relevancia para prevenir problemas comunes a largo plazo. Esto impone una estructura de tres capas (Vista, Lógica y Datos) y una más detallada de 7 capas (Interfaz gráfica del usuario, Controlador, Interfaces, estructura, lógica, Maneo, Objeto relacional y acceso a datos). Esto nos facilita el desarrollo y también optimizar los sistemas. El artículo concluye que la adopción de arquitecturas de software es clave para desarrollar sistemas flexibles y sostenibles, surgiendo mejoras como la automatización de manejadores y el control de transacciones.

### Reflexión

El texto resalta la importancia de las arquitecturas de software en capas, utilizando el patrón modelo-vista-controlador (MVC) para mejorar y organizar y mejorar la reutilización y el mantenimiento de código. Presenta ejemplos prácticos, como las clases de cliente y teléfono.datos, para ilustrar sus ideas a la articulación.

Aunque es un enfoque sólido, se sugiere profundizar la automatización de manejadores y el control de transacciones para mejorar la flexibilidad del sistema. En general, el texto subraya como una arquitectura bien estructurada favorece la creación de sistemas más sostenibles y fáciles de mantener.

# Lenguajes de patrones de diseño bajo una perspectiva cognoscitivista

## Resumen.

Este artículo nos ayuda a analizar la aplicación del concepto de "lenguaje de patrones" en el diseño de software, basándose en las ideas de Alexander (1979) y su relación con la estructura de sistemas de patrones. Aunque el término "lenguaje de patrones" en el diseño de software, basándose en las ideas de Alexander (1979) y su relación con la estructura de sistemas de patrones. Aunque el término "lenguaje de patrones" se ha utilizado ampliamente, aún persisten dudas teóricas y prácticas sobre su significado y aplicación.

Además se sugiere que el diseño de un lenguaje de patrones debe enfocarse en su organización y evolución para mejorar su eficacia en el campo del desarrollo de software.

## Reflexión.

La reflexión que nos deja este artículo resalta la complejidad y la importancia del lenguaje de patrones en el diseño de software, señalando que aunque el concepto ha sido ampliamente utilizado, su comprensión aún carece de una base consolidada. Esta falta de claridad teórica podría dificultar su implementación efectiva.

Además el enfoque en la evolución y organización del conocimiento sobre patrones arroja una mejora continua en su aplicabilidad, lo que puede ser clave para avanzar en el campo de desarrollo de software.

## Patrones de diseño para mejorar accesibilidad y uso de aplicaciones sociales para adultos mayores.

**Resumen.** El trabajo aborda el diseño de aplicaciones para adultos mayores.

El propósito de este artículo es proponer un conjunto incompleto de 36 modelos para el diseño de interacción en aplicaciones sociales dirigidas a personas mayores. La propuesta se basa en esfuerzos previos, a menudo expresados como normas y directrices de diseño, y busca identificar posibles problemas de usabilidad de dichas interfaces.

El trabajo aborda temas desde dos perspectivas: la del técnico y la del grupo Social de personas mayores. Los resultados muestran que el modelo propuesto facilita la creación de interfaces diseñadas, que brindan una mejor experiencia de usuario y tienen impacto positivo en la calidad de vida de los Personas mayores.

### P reflexión

la reflexión que nos resalta en este artículo de crear interfaces tecnológicas accesibles y usables para las personas mayores, un grupo que a menudo enfrentan barreras en el uso de aplicaciones sociales debido a limitaciones cognitivas y físicas.

Al integrar un conjunto de modelos de interacción y técnicas de evaluación heurística, se ofrece un enfoque centrado tanto en la perspectiva técnica como en las necesidades sociales de este colectivo, con el fin promover la adopción de la tecnología y mejorar calidad de vida de los usuarios mayores.

## Arquitectura de software de una aplicación móvil para desarrollar un sistema identificado por radiofrecuencia

### Resumen

Los sistemas de identificación por radiofrecuencia (RFID) se utilizan para identificar objetos automáticamente, y en su caso, se está desarrollando un sistema RFID para mejorar el control de inventarios en el sustituto tecnológico de Orizaba. Actualmente, el sistema de inventarios del instituto presenta problemas como falta de concordancia entre los activos registrados y realmente existentes, así como un tiempo de actualización lento. La actualización e implementación de este sistema incrementa la seguridad y el control de los bienes y permitir mantener el inventario actualizando en tiempo real, evitando errores humanos y robos.

### Reflexión

El uso de tecnologías RFID para el control de inventarios como en el Instituto Tecnológico de Orizaba representa un avance significativo en la eficiencia y precisión en la gestión de activos. Este sistema no solo mejora la actualización y consulta de los inventarios, sino que también facilita la localización de los bienes de manera más rápida y precisa, reduciendo el margen de error humano.

Este tipo de innovación es crucial para avanzar hacia un modelo de gestión más modernos y eficiente en el ámbito educativo y organizacional. Sin embargo también es importante considerar los costos iniciales e implementación y la capacitación necesaria para el personal, lo que podría representar un desafío en términos de inversión y adopción.

# Lenguaje de patrones de arquitectura de software: una aproximación al estado de artes.

## Resumen.

Examina el estado actual de los lenguajes de patrones en la arquitectura de Software. Se analizan, de origenes y avances y aplicaciones en la construcción de arquitecturas en diversos dominios, destacando su importancia para diseñadores y desarrolladores. Se revisa la evolución de la ingeniería de Software. El artículo menciona Christopher Alexander y Frank Buschmann, quienes contribuyeron al desarrollo de patrones en la arquitectura civil y de software.

## Reflexión

El artículo ofrece una visión clara sobre los lenguajes de patrones en la arquitectura de Software, destacando su evolución y relevancia en el desarrollo de sistemas mantenibles y alta calidad.

Conecta sus orígenes desde los autores Frank Buschmann en la arquitectura civil hasta su adaptación. Además, presenta ejemplos prácticos como su aplicación en e-business y gestión de entidades, lo que refleja la versatilidad de estos patrones.

# D M A

## Desarrollo de sistemas de software con patrones de diseño orientada a objetos

### Resumen.

Aborda la implementación de patrones de diseño orientados a objetos en la creación de un sistema de software denominado "Intranet Industrial" desarrollando en la Facultad de ingeniería Industrial de la Universidad Nacional Mayor de San Marcos. Se analiza el impacto económico de la adopción de estos patrones en comparación con un enfoque que no los considera.

La sección final ofrece una descripción del sistema, sus módulos y herramientas, así como una estimación de costos utilizando el patrón "Informador" y el método COCOMO. Se concluye que la implementación de patrones contribuyen a mejorar la eficiencia y disminuir los costos en el desarrollo de software.

### Reflexión:

El enfoque de utilizar patrones de diseño en el desarrollo de software es una estrategia altamente efectiva que no solo optimiza el proceso de creación, sino que también promueve la reutilización y la escalabilidad. Este es un trabajo excelente para el ejemplo de cómo integrar la teoría y práctica, mostrando que la implementación de patrones no solo es beneficiosa desde el punto de vista técnico, sino que también tiene repercusiones positivas en la gestión generadas por los costos.

Arquitectura de software para el desarrollo de herramientas tecnológicas de costos, presupuestos y programación de obra.

## Resumen.

El artículo se centra en el diseño del software académico para la gestión de costos y presupuestos en ingeniería civil. Se organiza la información relacionada con materiales, mano de obra, maquinaria y transporte en grupos de procesos, buscando aumentar la productividad y facilitación en la elaboración de productos y presupuestos.

La propuesta tiene como objetivo modernizar la enseñanza y mejorar la calidad educativa, ofreciendo una herramienta accesible y gratuita para los estudiantes.

## Reflexión

Me parece interesante porque propone un software práctico y accesible, pensado para facilitar la gestión de costos, presupuesto y programación. Me gusta por que utilizan metodologías como la EBT y el método de la ruta crítica ya que ayudan a conectar la teoría con la práctica. Además, que sea gratuito lo hace aún más valioso. Solo me pregunto como se asegura su actualización, ya ha sido probada por usuarios de resto el artículo ayuda a entender las arquitecturas.

# Una arquitectura basada en software libre para archivos web

## Resumen

La información en la web es extensa pero puede ser eliminada. Un documento web está compuesto por archivos HTML y otros elementos. Se seleccionaron los más adecuados para la arquitectura propuesta, priorizando el uso de software libre.

Se está desarrollando un prototipo para arrancar la integración y rendimiento de estos componentes. La perseveración de la información web es crucial para garantizar el acceso continuo a recursos digitales y proteger el patrimonio cultural.

## Reflexión

La información en la web es extensa pero puede ser eliminada. Un documento web está compuesto por archivos HTML y otros. La abundante información en Internet puede desaparecer sin previo aviso. Un documento web y sus otros elementos.

Resulta esencial garantizar la perseveración de datos en la red para garantizar el continuo acceso a la información digital y así poder proteger nuestra rica herencia cultural. El objetivo principal consiste crear un marco de referencia que facilite la ejecución de sistemas.

## Patrones de adaptación para arquitecturas de software basadas en tecnologías de acuerdo

### Resumen.

Los sistemas actuales son cada vez más complejos y requieren un enfoque adaptativo para gestionar su evolución. La auto-adaptación no solo abarca el comportamiento funcional, sino también las propiedades no funcionales, es esencial en los sistemas.

La propuesta presentada se basa en un ecosistema de servicios de tecnologías del acuerdo (AT) donde los agentes se organizan y coordinan dinámicamente, formando organizaciones adaptativas.

### Reflexión.

La capacidad de auto-adaptación es fundamental para los sistemas modernos, ya que permiten una mayor flexibilidad y eficiencia frente a cambios en su entorno o en los requerimientos.

El desarrollo de patrones de adaptación y de integración en plataformas como Thomas es un paso crucial en la creación de arquitecturas que pueden autoorganizarse y adaptarse en tiempo real, este enfoque no solo mejora la eficiencia, sino que también proporciona una base sólida.

## Arquitectura de software basada en microservicio para desarrollo.

Muchas empresas en el sector público y privado desarrollan software para automatizar sus procesos internos mediante arquitecturas tradicionales y monolíticas. Aunque estos sistemas han sido útiles, presentan limitaciones significativas: el mantenimiento se vuelve complicado, la escalabilidad es limitada y las actualizaciones nuevas funcionalidades son complejas y lentas. Por ello la investigación actual se enfoca en explorar nuevas arquitecturas de software que puedan superar estos obstáculos y adaptarse mejor a las tecnologías modernas. La migración hacia una arquitectura más moderna en una respuesta logrando ante los desafíos planteados por los sistemas monolíticos, lo que permiten a la empresa mejorar en agilidad, capacidad de respuesta y eficiencia en la gestión y desarrollo de sus soluciones tecnológicas.

## Bibliografía

López, D., y Mayo, E. Arquitectura de software basada en microservicios para desarrollo de aplicaciones.

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

El avance tecnológico ha impulsado nuevas metodologías y herramientas en las empresas, destacando la importancia de la arquitectura de software. El trabajo Analiza los frameworks más utilizados en el desarrollo de aplicaciones empresariales y su aplicabilidad según el proyecto. La arquitectura adecuada, los requerimientos específicos y la elección adecuada del framework son clave para el éxito de una aplicación empresarial. La correcta combinación de componentes, una buena gestión y comunicación constante son fundamentales para la implementación exitosa. La investigación proporciona una guía inicial sobre el uso de frameworks en proyectos empresariales.

Bibliografía: Sarasti Espino, H.F (2016 Abril 12)  
<http://sedici.unpl.edu.ar/handle/1005/52783>

# COMO EMPLEAR EL SOFTWARE LIBRE EN LA ARQUITECTURA Y EL DISEÑO?

## Resumen.

Este trabajo expone el uso del software libre en disciplinas fuera de la informática, específicamente en arquitectura y diseño. El enfoque destaca cómo estos herramientas pueden mejorar la calidad de los trabajos realizados con recursos limitados.

El autor muestra ejemplos prácticos que demuestran las bondades y ventajas de estas herramientas en áreas como el diseño arquitectónico e industrial. El Software libre fomenta el intercambio de ideas y la creación de modelos de trabajo que promueven la libertad y creatividad en el proceso de diseño.

## Reflexión

Este trabajo expone el uso del software libre en disciplinas lo cual reduce el ámbito tecnológico. Sino en campos como la arquitectura y el diseño donde la creatividad y la accesibilidad son cruciales. Además, el intercambio libre de ideas y la colaboración que posibilita el software libre son fundamentales para avanzar hacia un diseño más inclusivo y democrático.

Con estas herramientas, los profesionales pueden explorar nuevas formas de creación sin depender de costosos programas privativos, lo que abre el campo a una mayor diversidad de enfoques y soluciones creativas.

## Evaluación de una arquitectura de software.

### Resumen

El sistema de evaluación y calificación en los direcciones territoriales de salud, los ens y los IPS en Colombia está regulado por la resolución 4505 de 2012, que establece la estructura del reporte de actividades de salud pública. La Secretaría Municipal de Salud Villavicencio (SIRSU) recibe reportes periódicos de la IPS con información sobre la población no asegurada (lo que representan un alto volumen de datos), lo que generando errores en esos registros.

### Reflexión:

El desarrollo de un sistema automatizado para la validación de datos en salud es fundamental en el contexto en el manejo de grandes volúmenes de información es crucial para tomar decisiones acertadas. La transparencia en el proceso validación, como lo que ocurre en el SIRSU, no solo reflejiza el flujo de trabajo sino que también pone en la precisión de la información, afectando decisiones importantes sobre la salud pública.

Marco de referencia de arquitectura de software para aplicaciones web y móviles.

## Resumen.

La movilidad en el ámbito de las aplicaciones corporativas es una necesidad, pero también representa un reto debido a sus costos. Para abordar esta cuestión, se propone una plataforma que gestione el desarrollo de la interfaz y la integración con aplicaciones internas de manera segura y escalable. La optimización de recursos en red y bases de datos que mejoran los tiempos de respuesta, y el uso de controles de interfaz, como ExtJS y Sencha Touch ofrecen soluciones fáciles de personalizar y usar.

## P reflexión.

El desarrollo de aplicaciones (web y móviles), especialmente en entornos corporativos, esto se debe centrarse no solo en la funcionalidad y la integración sino también en la optimización de los recursos y escalabilidad.

La modularidad y portabilidad de las aplicaciones, junto con la mejora en los tiempos de respuesta, son aspectos clave en la creación de sistemas eficientes y de alto rendimiento.

# DIMIA

## Desarrollo de una herramienta para el aprendizaje de patrones de diseño de Software.

### Resumen.

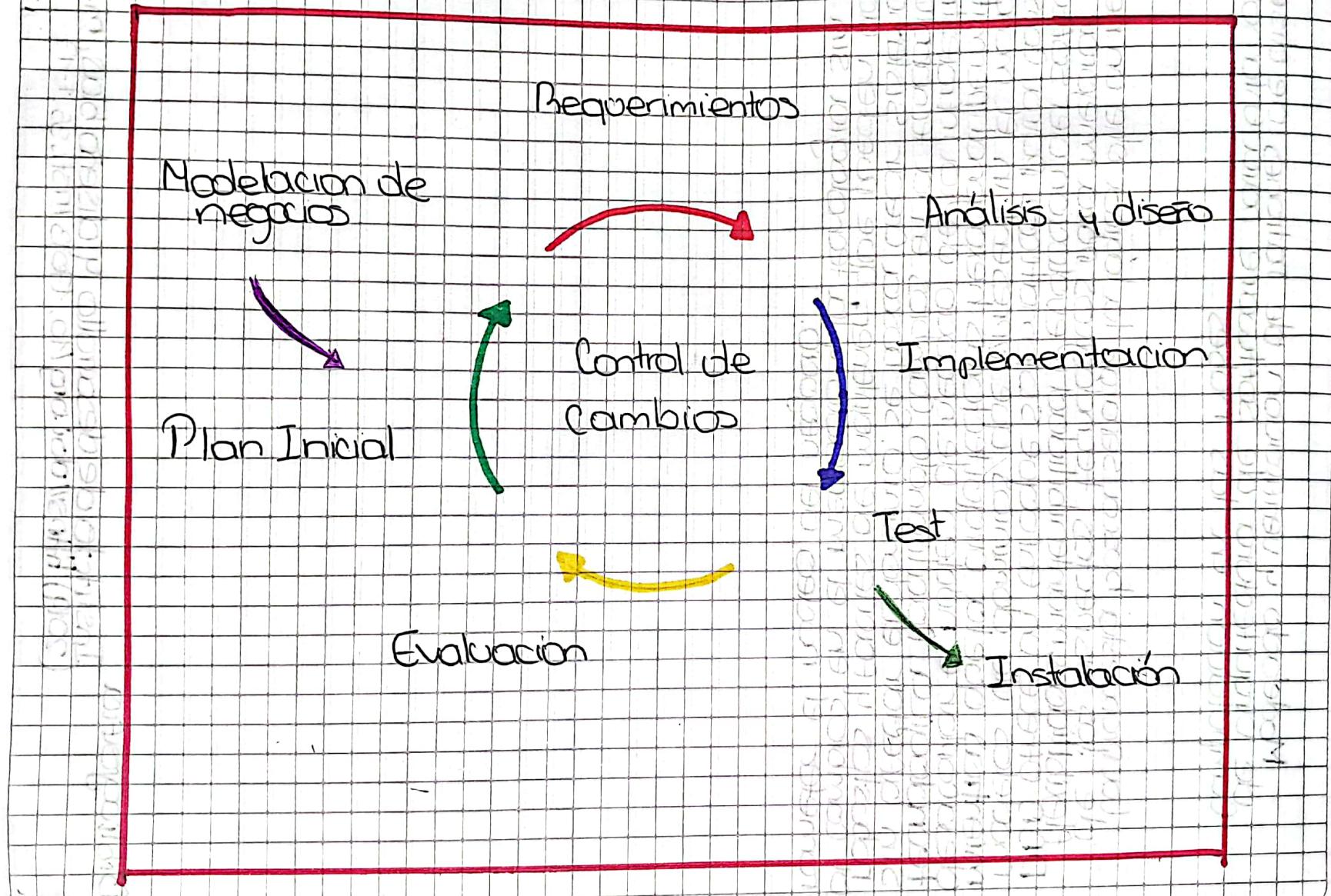
Los patrones de diseño ofrecen soluciones probadas y eficaces a problemas recurrentes en programación, facilitando la creación de código limpio y fácil de mantener. Estos patrones no sólo resuelven problemas técnicos, sino que también mejoran la comunicación entre los desarrolladores, ya que el conocimiento de un patrón permite comprender rápidamente su estructura y propósito.

El prototipo desarrollado puede ser la base para futuras mejoras y evoluciones por parte de otros estudiantes.

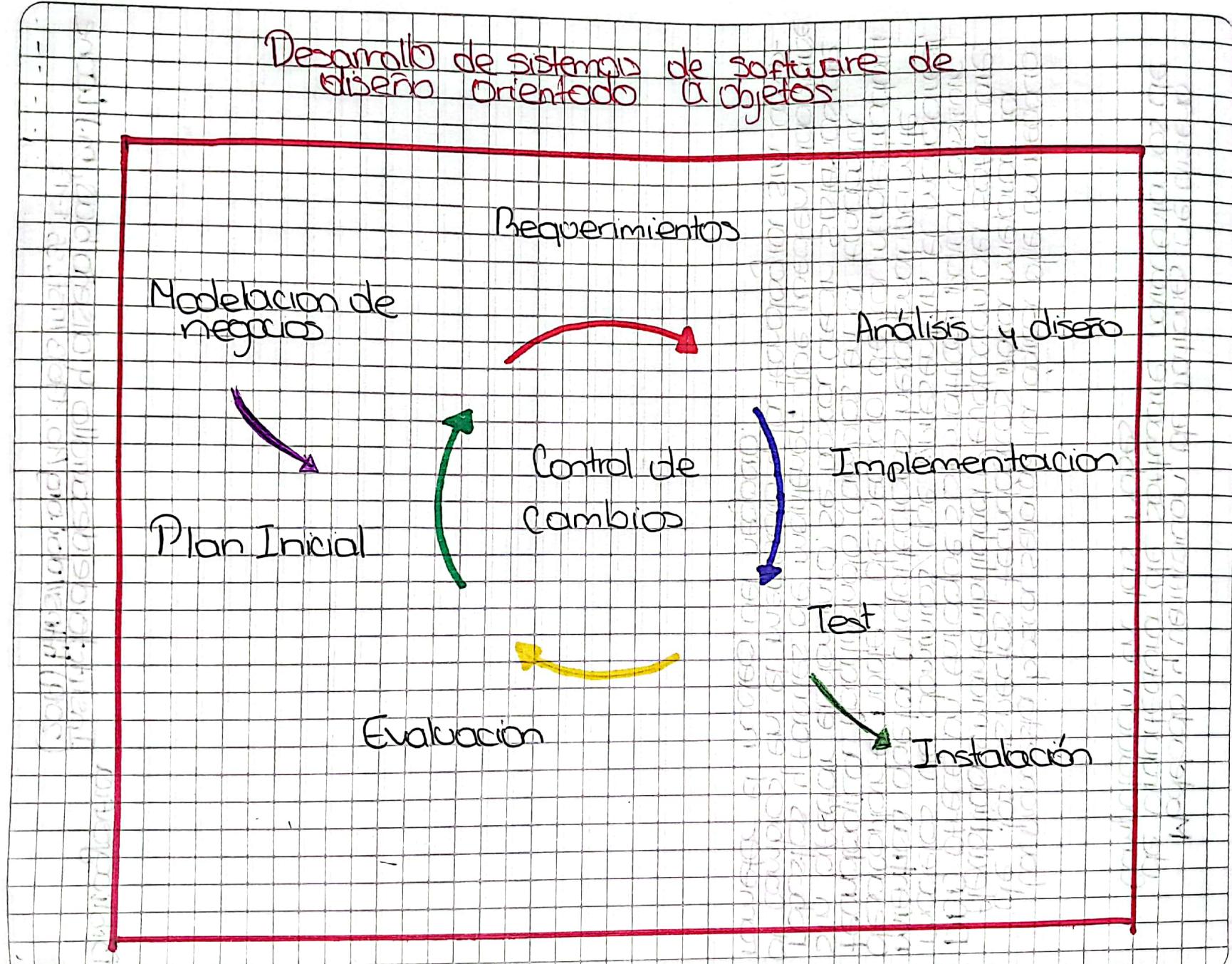
### Reflexión.

El desarrollo de una herramienta para aprender patrones de diseño es un paso significativo en la formación de futuros programadores, ya que permite que los principiantes comprendan la importancia de un diseño estructurado desde el inicio. El aprendizaje de tecnologías como Spring y el desarrollo front-end, inicialmente desconocidos para el autor, muestra cómo la resolución de problemas a través de la práctica y la investigación puede superar barreras y expandir habilidades. Este prototipo tiene un gran potencial para evolucionar y servir como base para futuras interacciones que mejoren su funcionalidad y la experiencia del usuario.

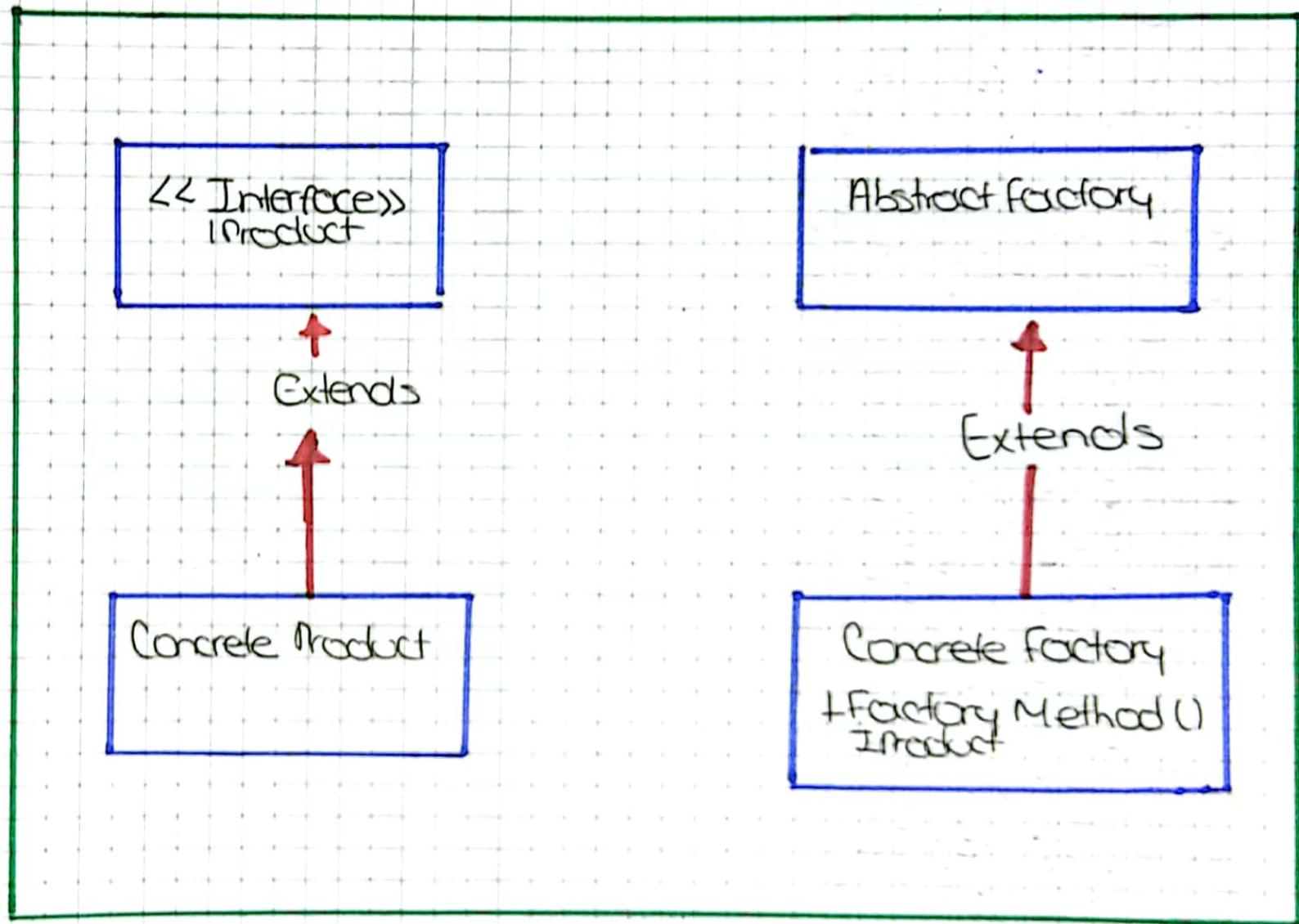
# Desarrollo de sistemas de software de diseño Orientado a Objetos



# Desarrollo de sistemas de software de diseño Orientado a Objetos

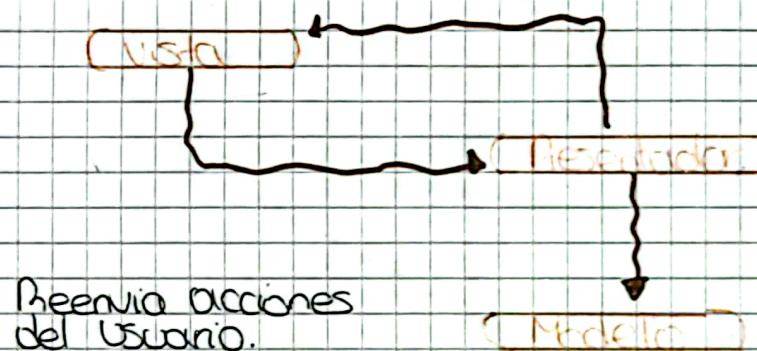


## Introducción de patrones de diseño



## Analisis Comparativo de Patrones de diseño de Software

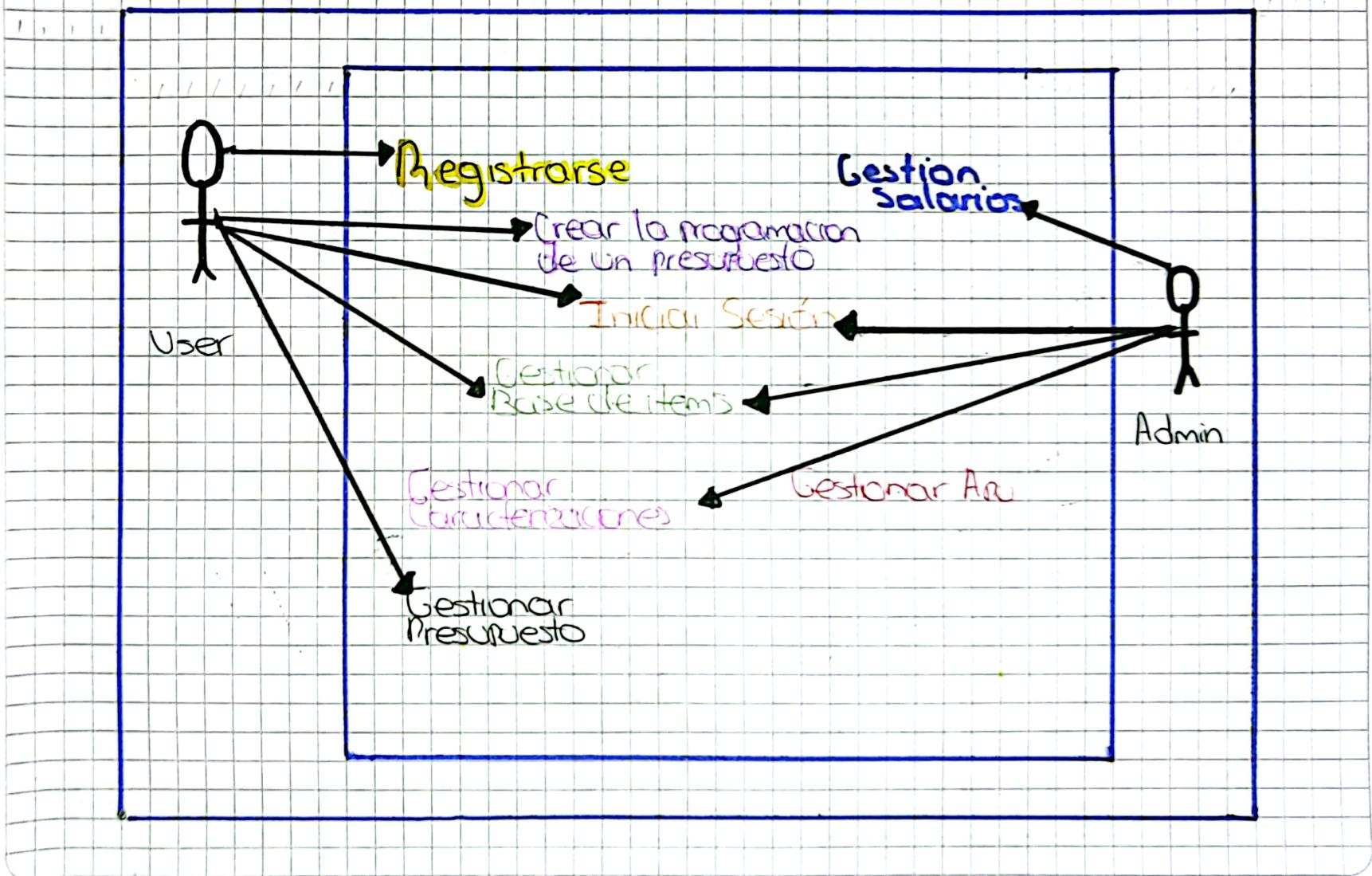
Utiliza el contrato para leer/escribir datos de vista



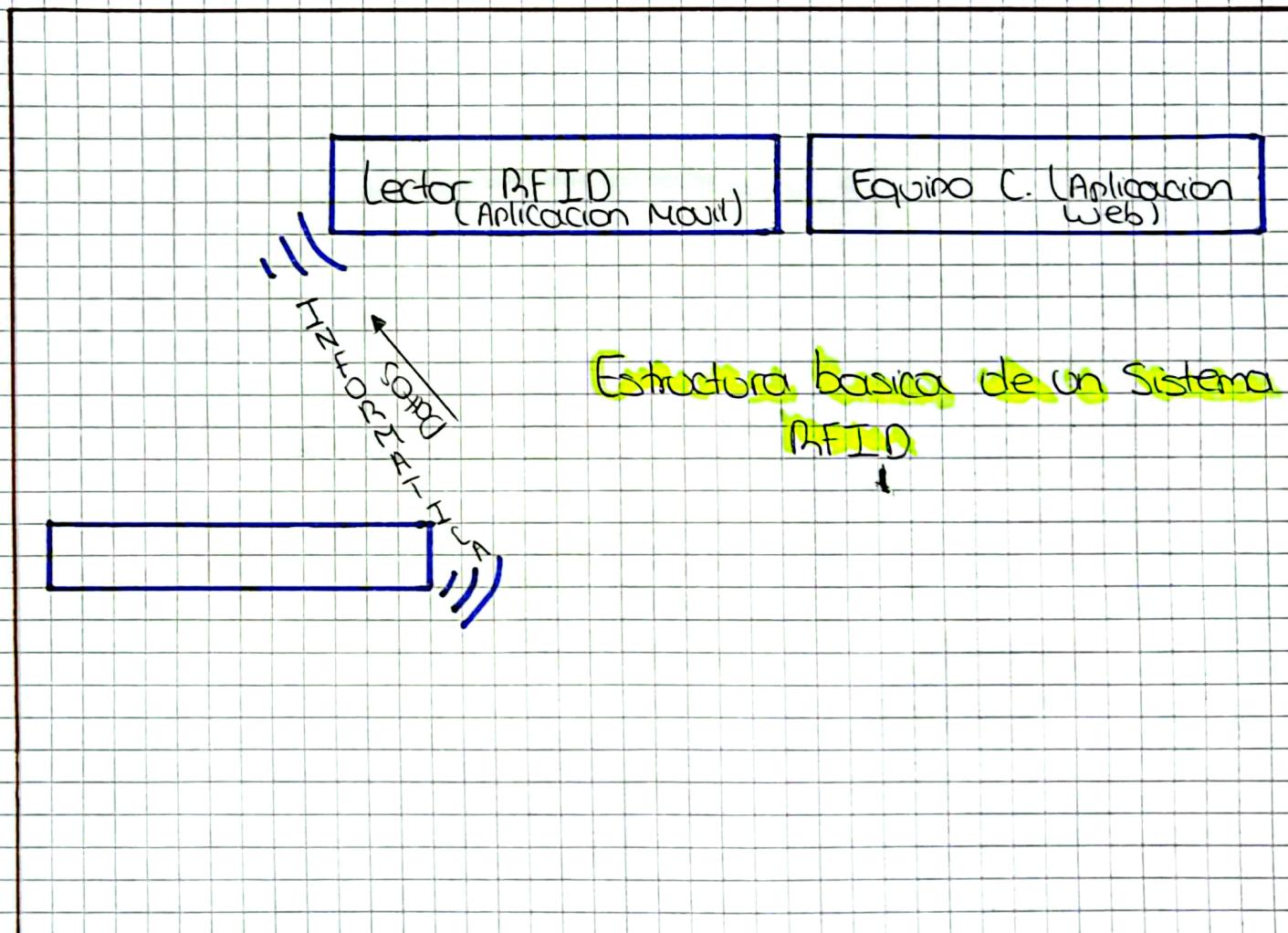
Invoca un método según la acción del usuario.

Reenvio acciones del usuario.

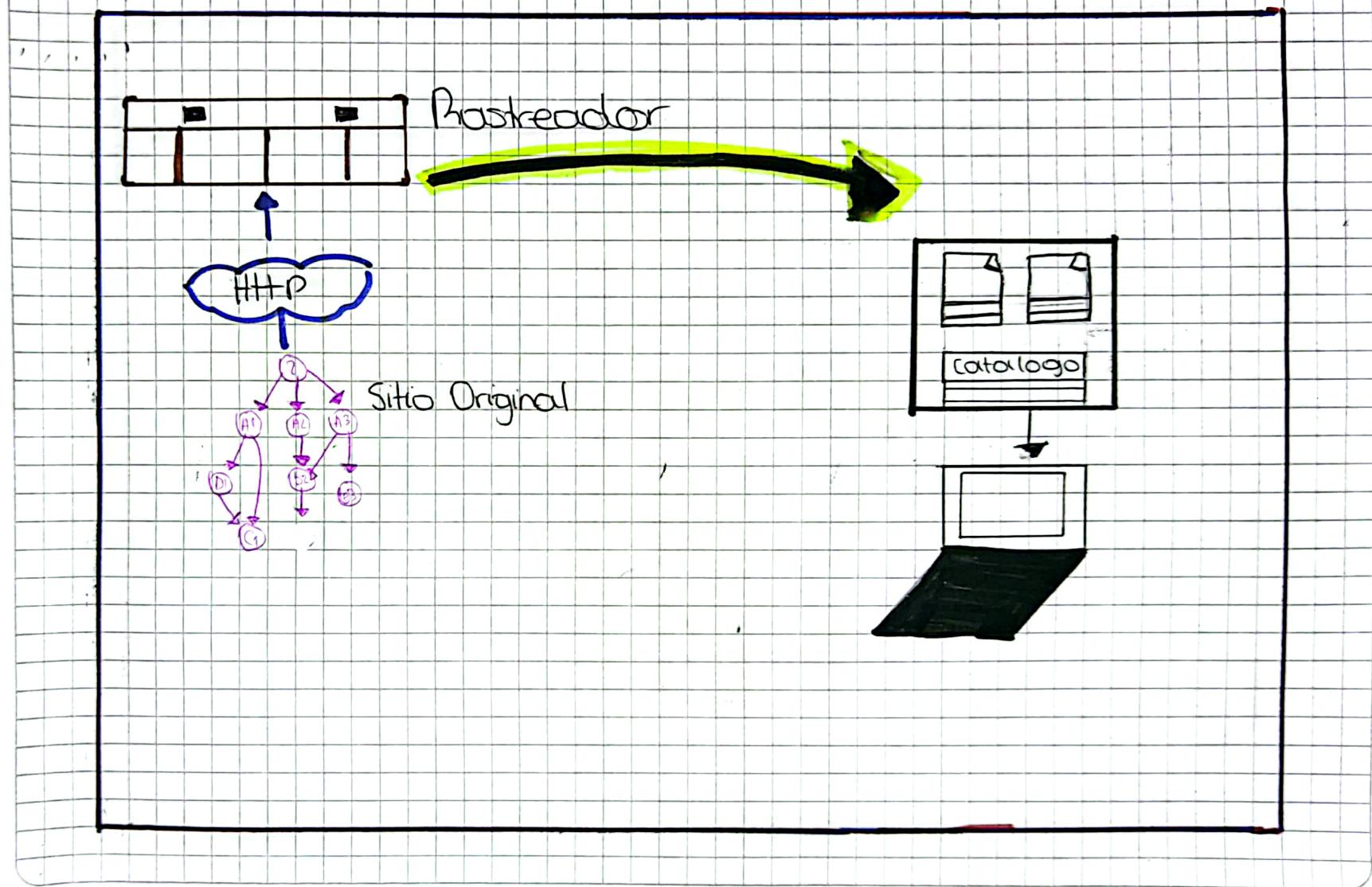
Arquitectura de software para el desarrollo de herramienta  
Tecnológica de Costos, Presupuestos y Programación  
de Obra.



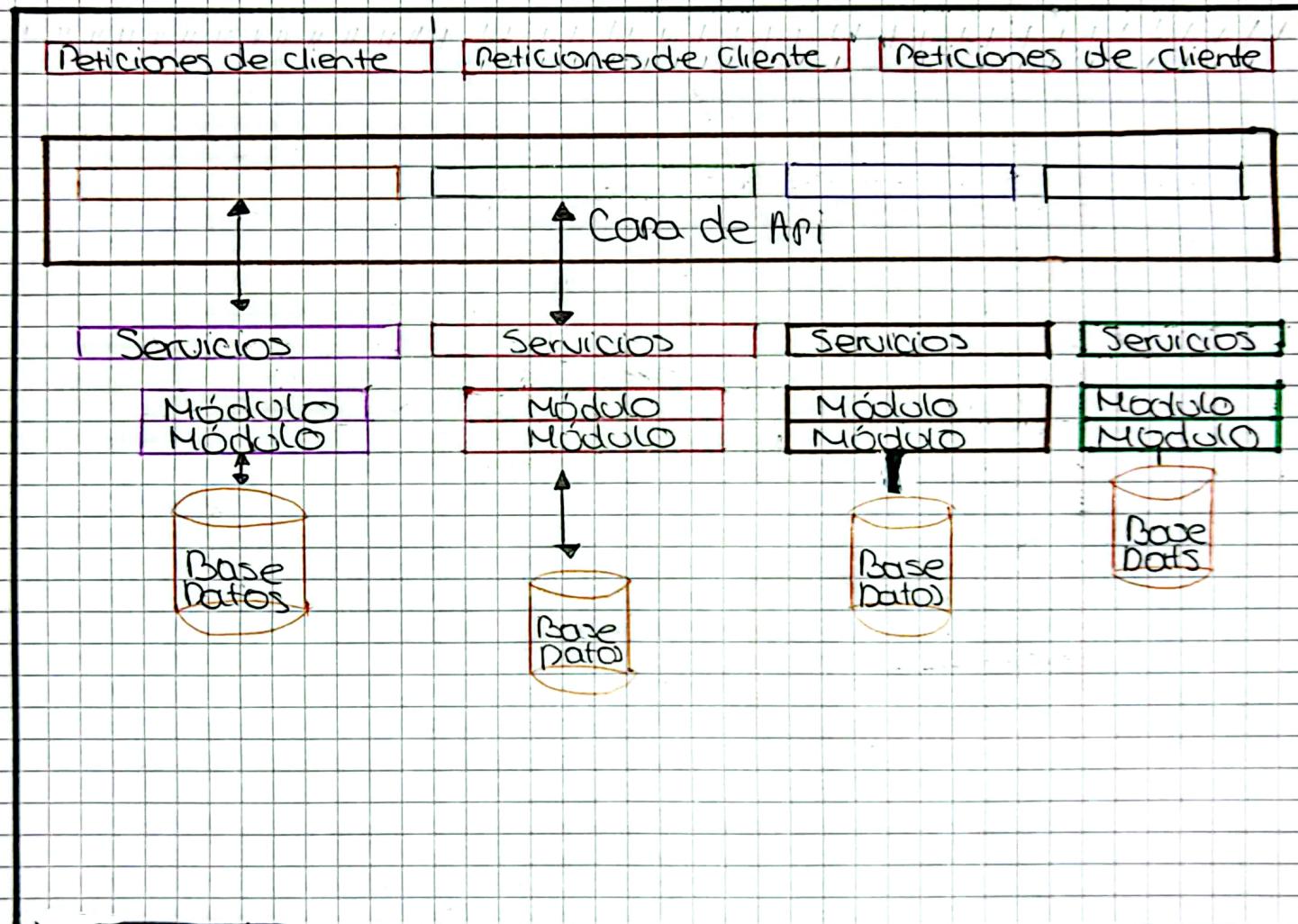
Arquitectura de software de una aplicación móvil para desarrollar un sistema de identificación por radiofrecuencia.



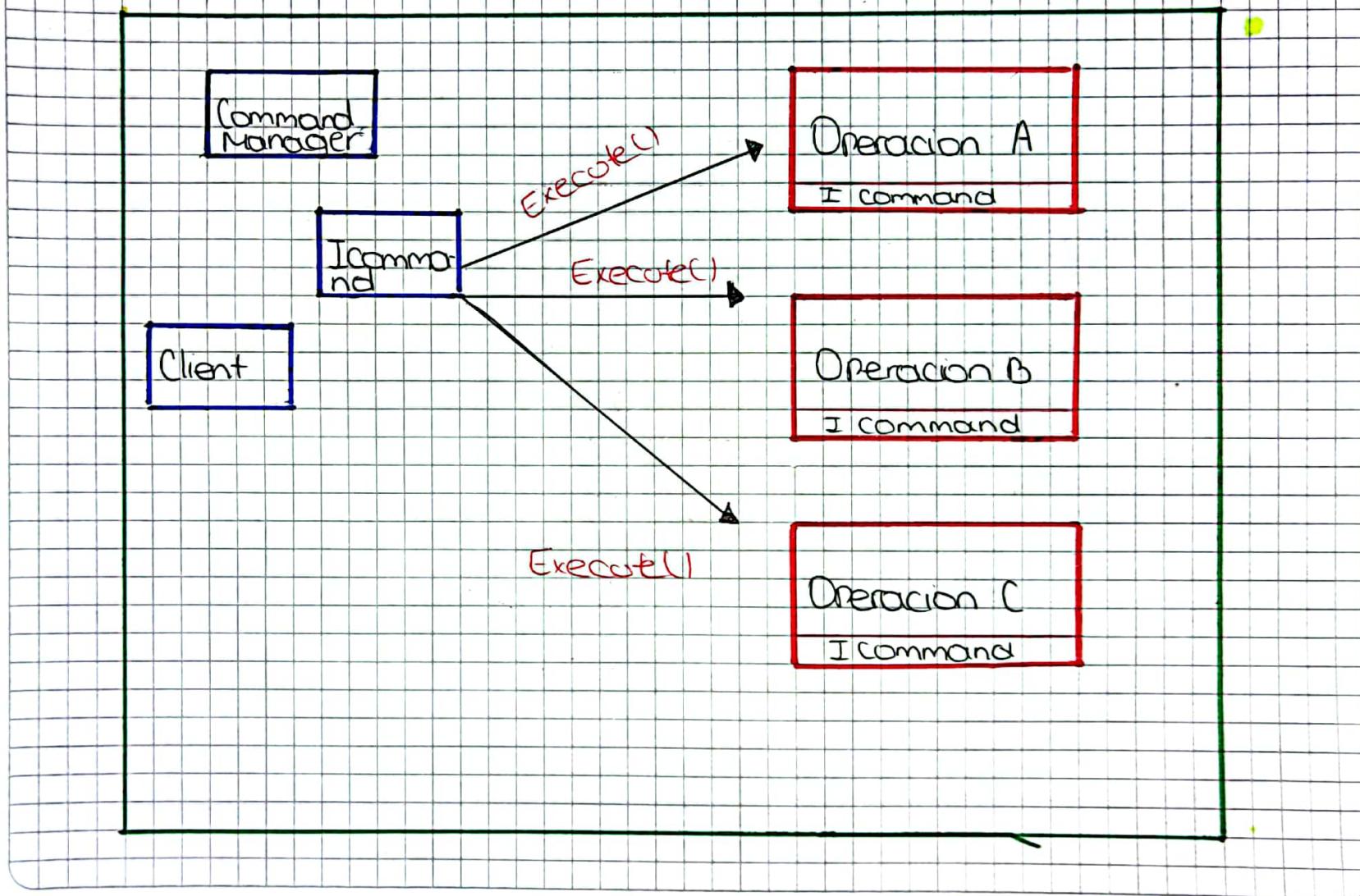
Una arquitectura basada en software libre para archivo web



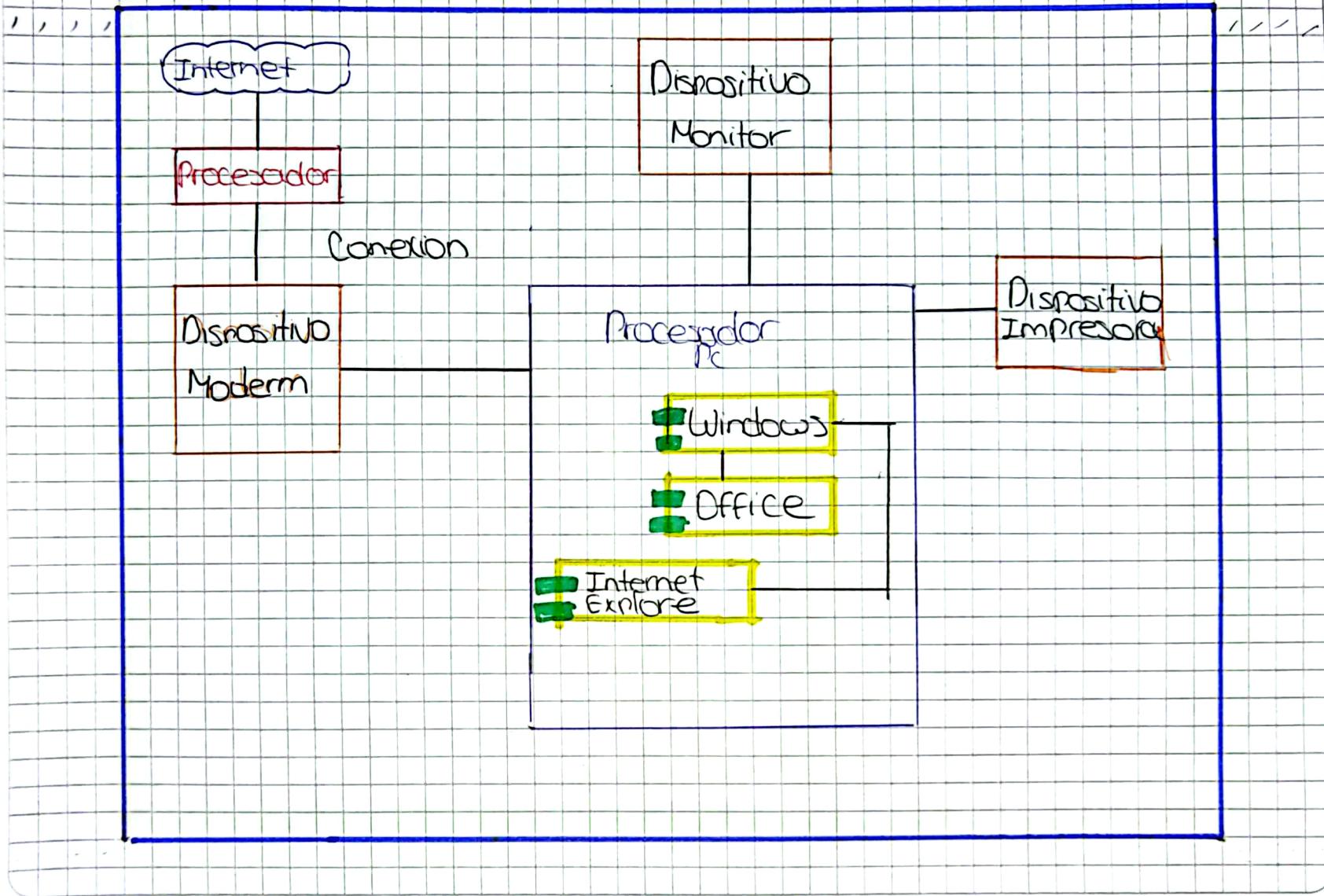
## Patrones de diseño para mejorar la accesibilidad y uso de aplicaciones sociales para adulto mayor



(lenguajes) de patrones de diseño de software bajo una perspectiva cognoscitivista.



# Una teoría para el diseño de Software



Herramienta para reuso de código JavaScript orientado a patrones de interacción.

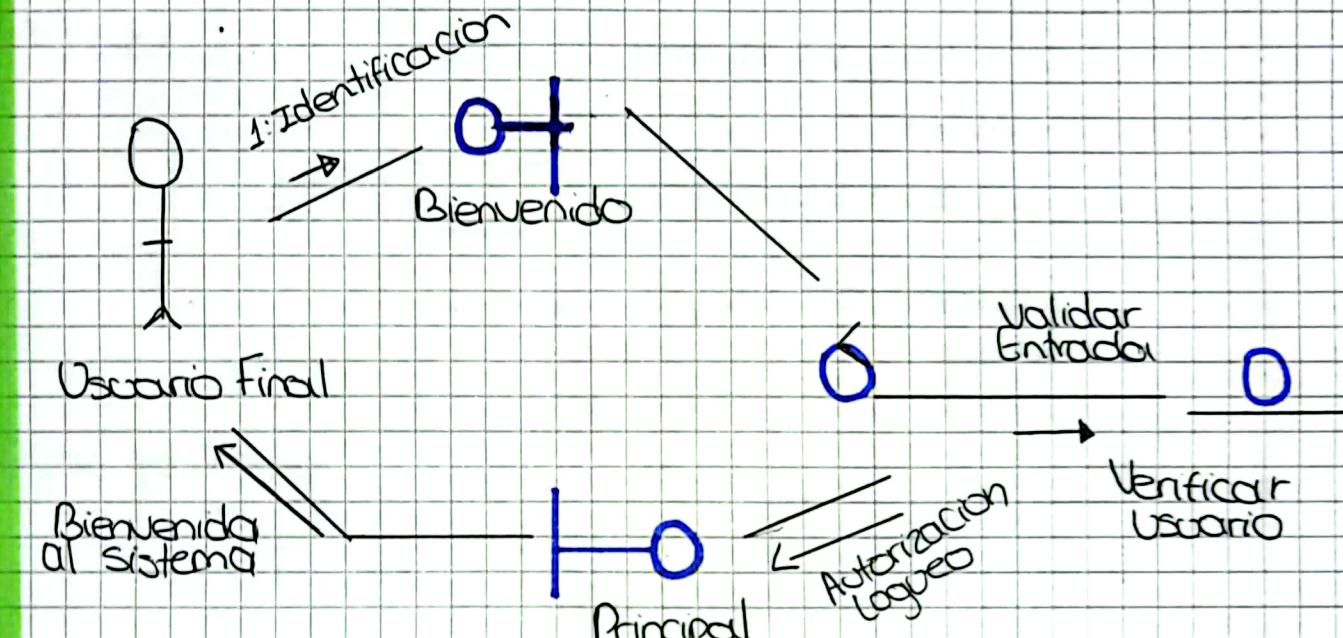
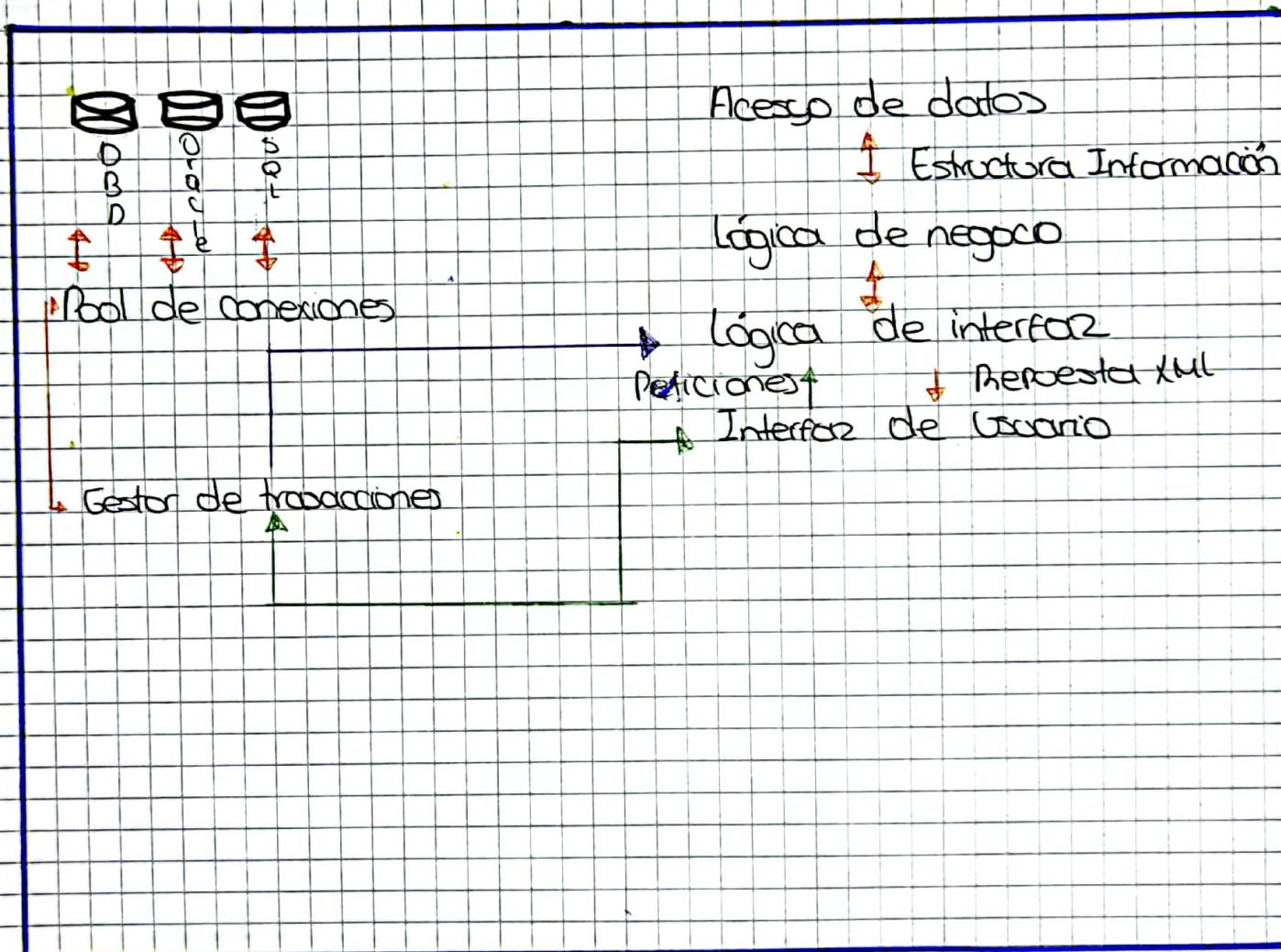
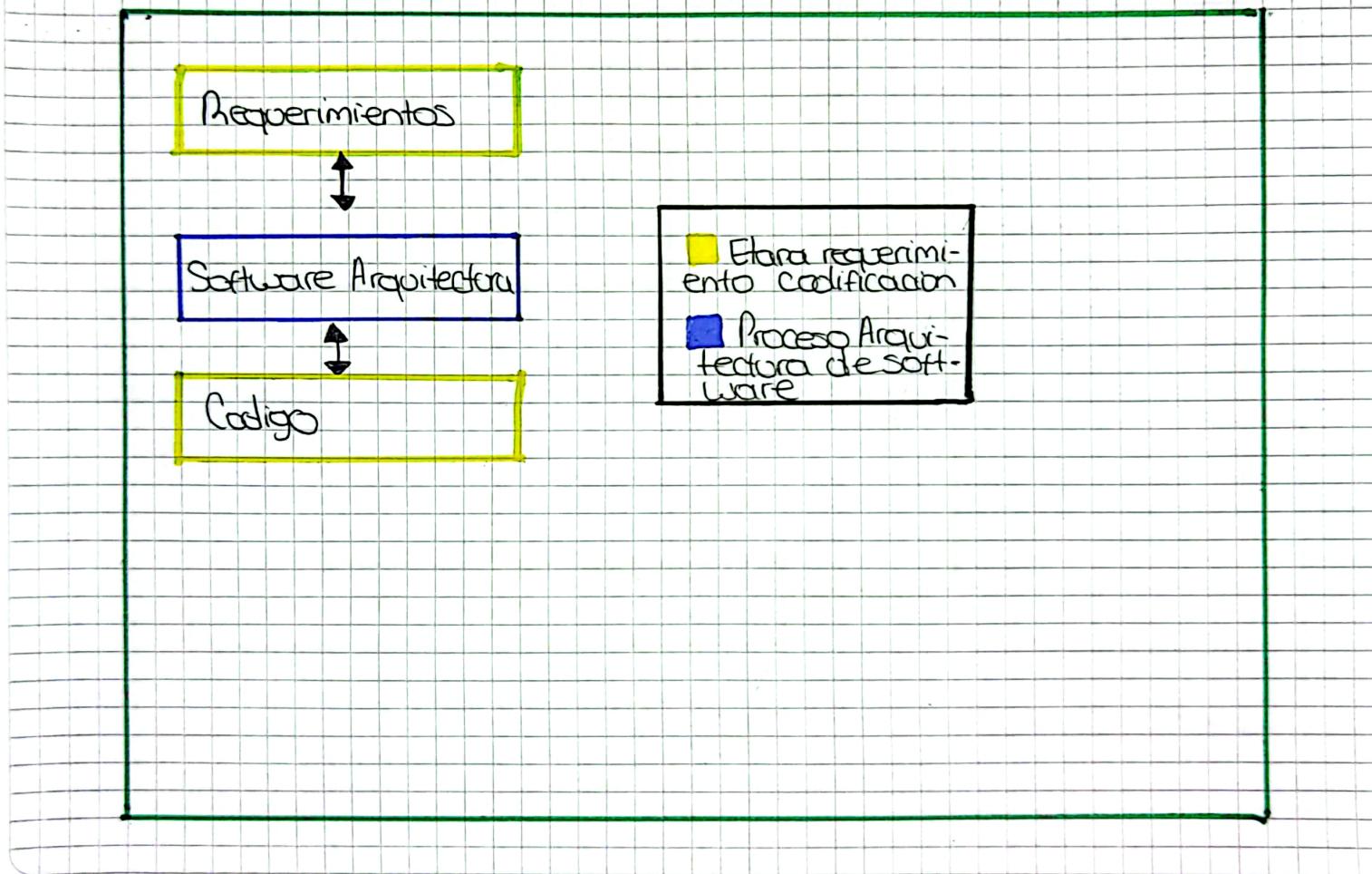


Diagrama de colaboración: logearse.

## Atributo de calidad y arquitectura Software



Revisión de elementos para la representación de las  
arquitecturas de referencia de software



Uso de notaciones de doblez: Un caso práctico.

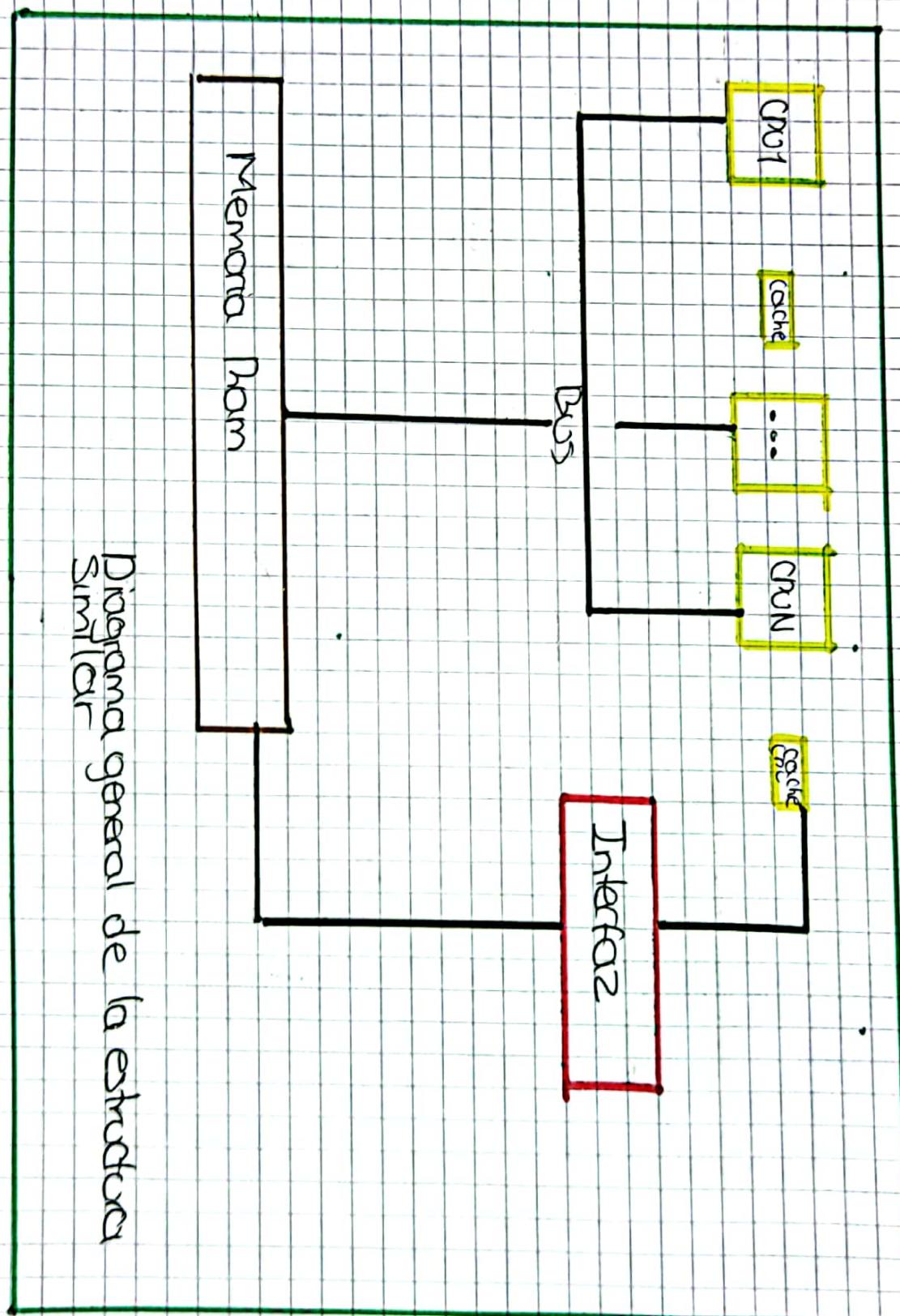
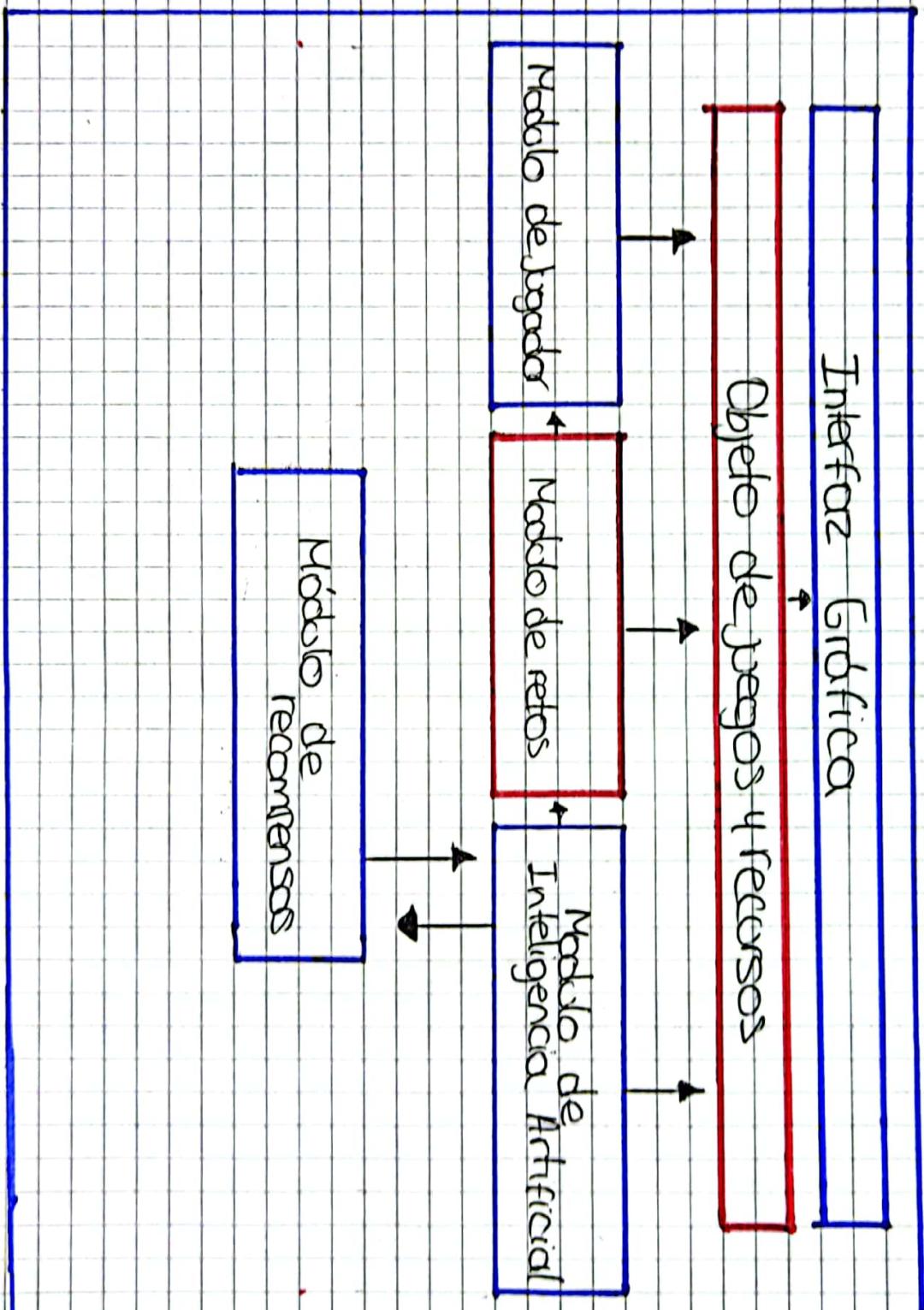


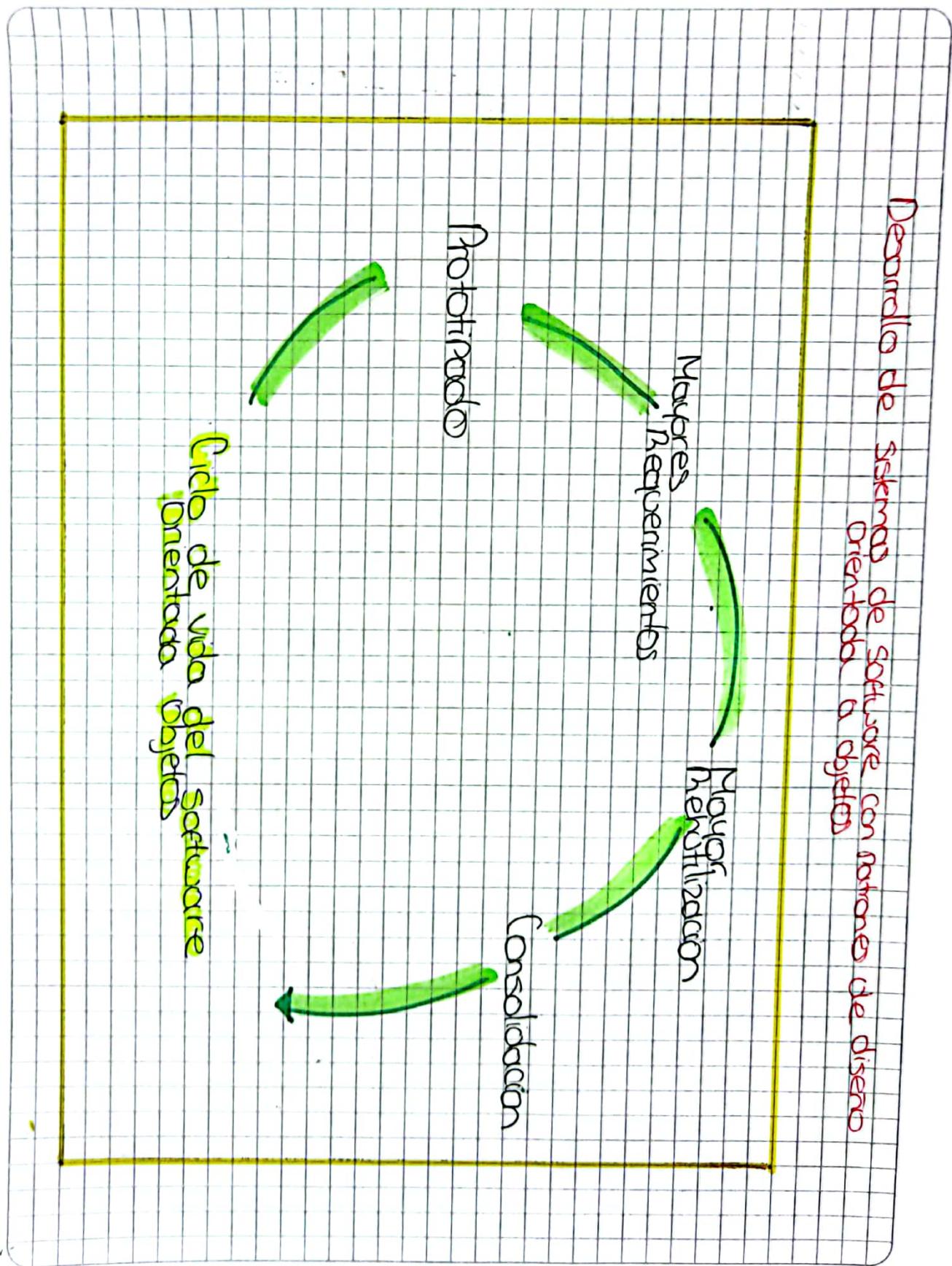
Diagrama general de la estructura  
Similar

Arquitectura de Software para el desarrollo de Videjuegos  
Sobre el motor Unity 3D.

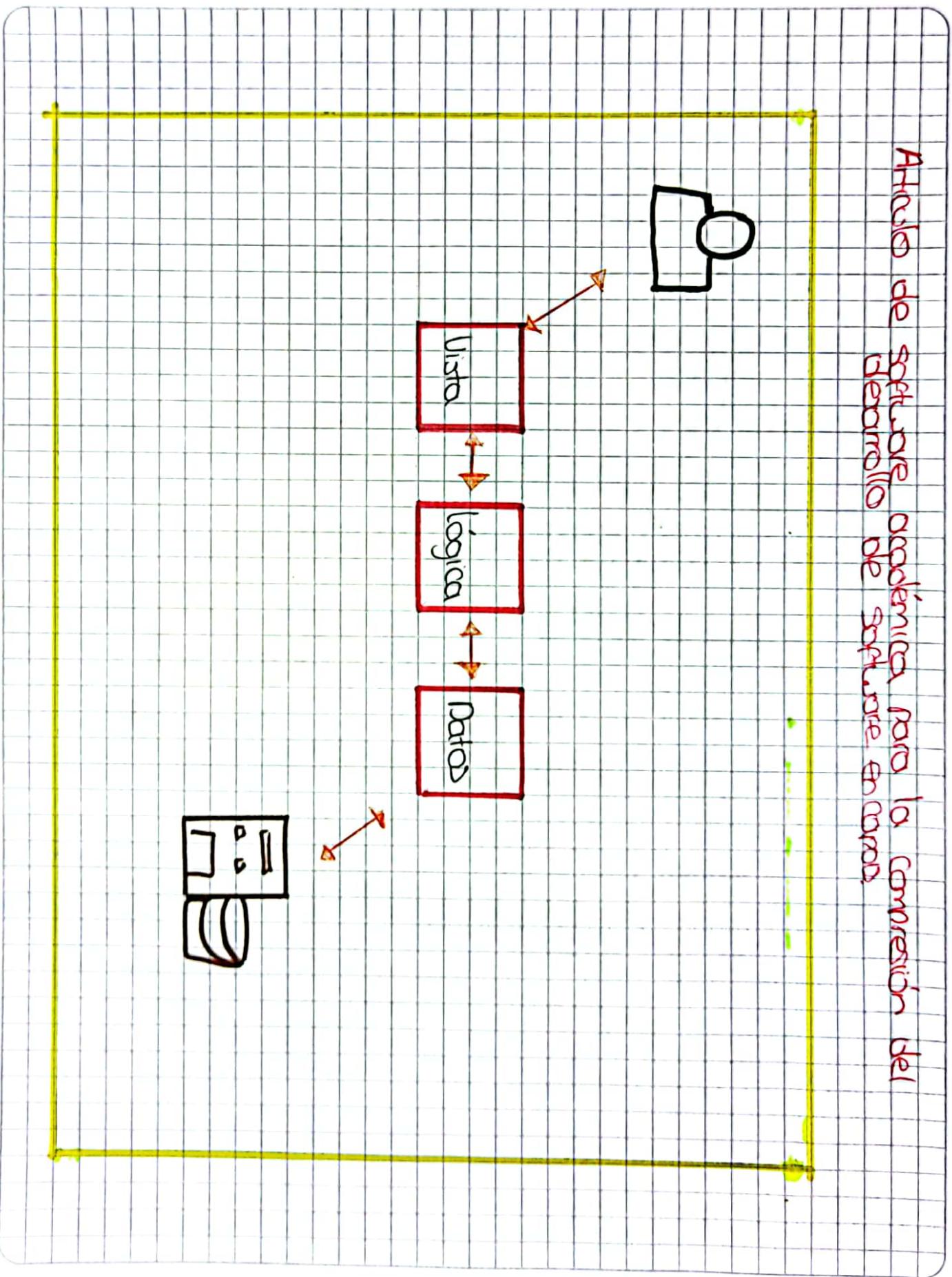


Desarrollo de sistemas de software con patrones de diseño  
orientados a objetos

Ciclo de vida del software  
orientado a objetos



Ahorro de software académico para la compresión del desarrollo de software en la clase.



Patrones de diseño (The gang of four) en el contexto de procesos de desarrollo de aplicaciones orientadas a la web

Patrón de diseño	Categoría
Builder	Creacionales
Factory Method	Creacionales
Singleton	Creacionales
Decorator	Estructurales
Facade	Estructurales
Iterator	Estructurales
Strategy	Comportamiento
Lazy Load Method	Comportamiento

Arquitectura de software basada en microservicios para desarrollo de aplicaciones web

Peticion de cliente

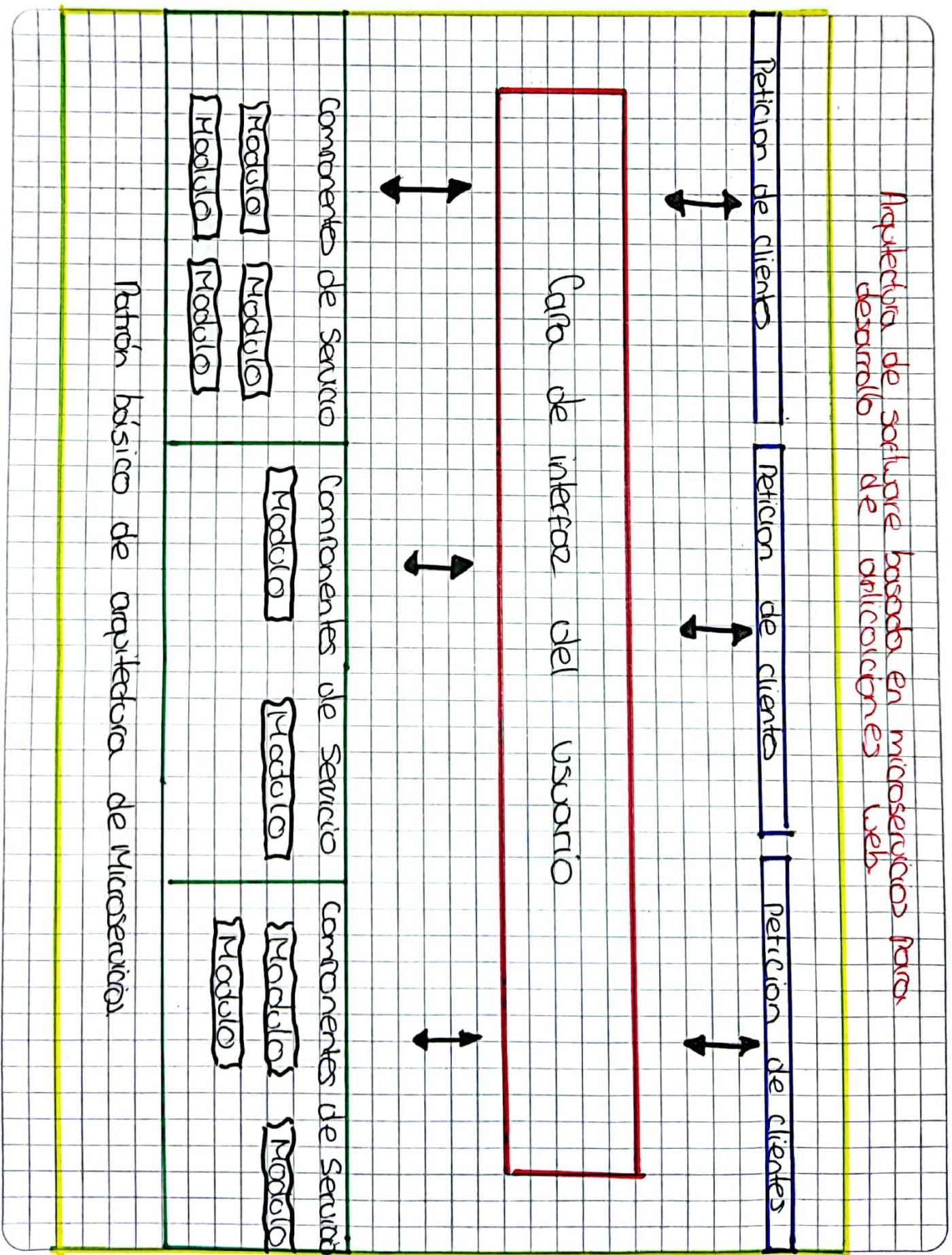
Peticion de cliente

Peticion de cliente

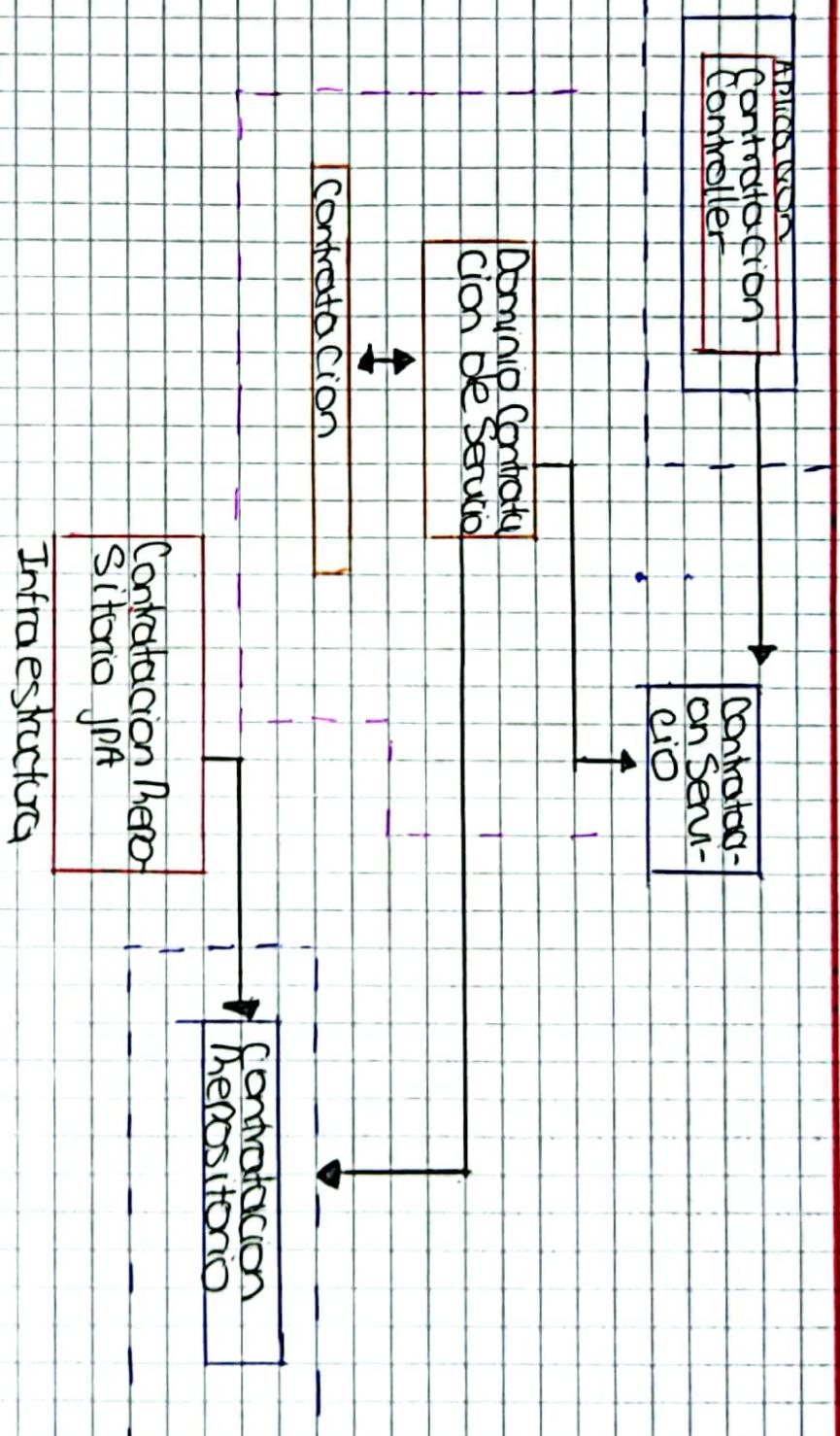
Capa de interfaz del usuario



Patrón básico de arquitectura de microservicios.

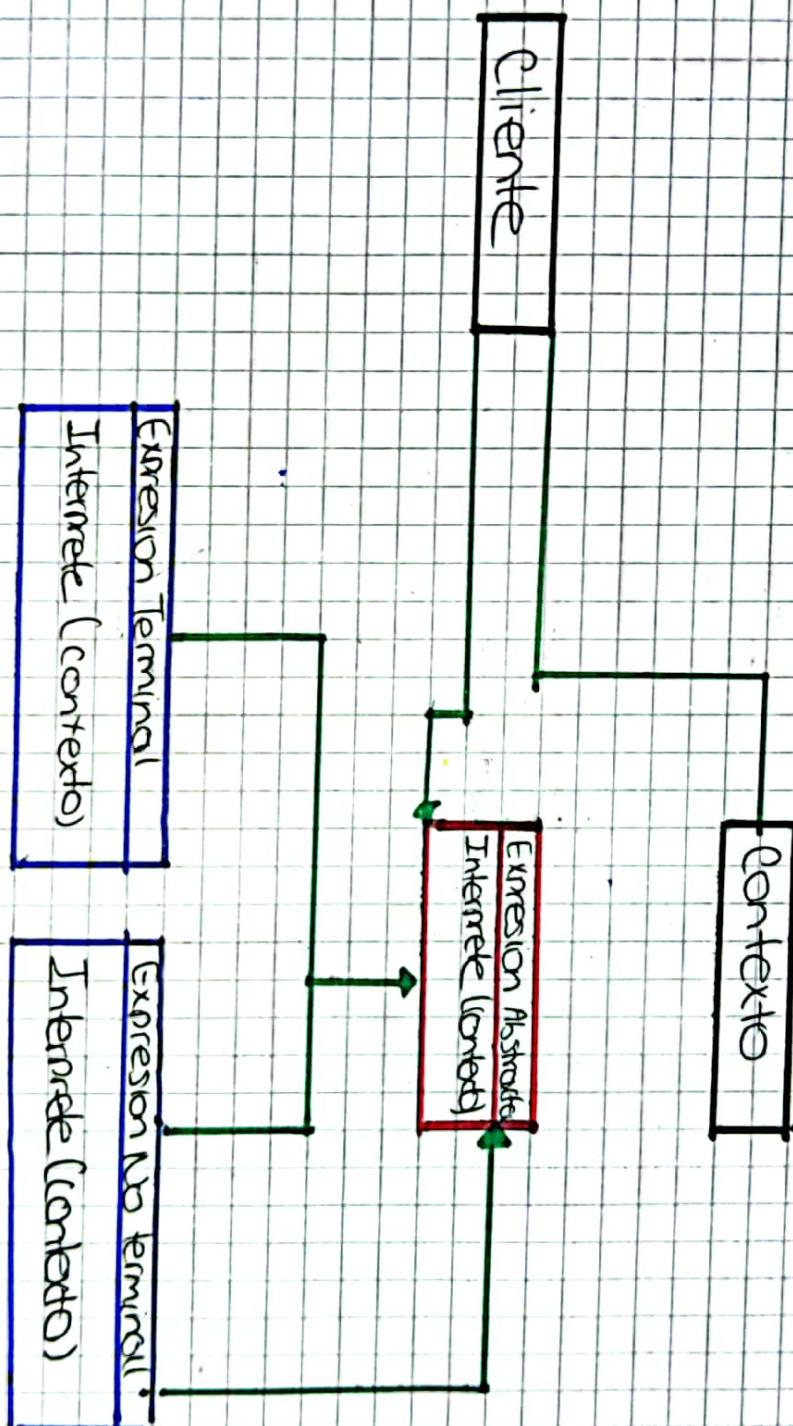


## Implementación de una arquitectura de software para el dominio.



Representación del caso de uso "Contratación de Servicio".

Lenguajes de patrones de proyección de software: uno con Kinson al estado de arte.



## Arquitectura de software esquemas y servicios

