Arquitecturas modulares en código reutilizable para proyectos de desarrollo de software

Oscar Alberto Neira Vaca

Trabajo de grado presentando para optar por el título de Profesional en

Ingeniería de sistemas

Asesor o Director

Diego Alejandro Vela Beltrán

Universidad Santo Tomás, Tunja

Ingeniería de Sistemas

2025

Dedicatoria

Primero quiero agradecer a Dios y a mis padres por el apoyo y el amor para poder llegar a donde estoy, a todas las personas que de una y otra forma me apoyaron en la Realización de este trabajo.

Contenido

Lista de figuras

Lista de apéndices

Resumen

Esta monografía aborda el impacto de las arquitecturas modulares y la reutilización de código en el desarrollo de software. El trabajo destaca cómo estas prácticas pueden transformar el diseño y mantenimiento de proyectos, mejorando tanto la eficiencia como la calidad. A través de un análisis comparativo de diferentes enfoques arquitectónicos, se identificaron retos significativos en la recopilación de datos y la integración de conocimientos. El proceso de investigación estuvo marcado por desafíos académicos y personales, superados gracias al apoyo de mentores, amigos y familiares. Este estudio busca motivar a profesionales del desarrollo de software a adoptar prácticas más flexibles y eficientes, contribuyendo al avance de soluciones tecnológicas sostenibles.

Palabras clave: Arquitecturas Modulares, Calidad de Software, Desarrollo de Software, Eficiencia, Reutilización de Código.

Comentado [MC1]: Organizar de forma alfabética

Abstract

This monograph explores the impact of modular architectures and code reuse on software development. The study emphasizes how these practices can transform project design and maintenance by enhancing efficiency and quality. Through a comparative analysis of different architectural approaches, significant challenges related to data collection and knowledge integration were identified. The research process involved both academic and personal challenges, which were overcome with the support of mentors, friends, and family. This study aims to inspire software development professionals to adopt more flexible and efficient practices, contributing to the advancement of sustainable technological solutions.

Keywords: Code Reuse, Efficiency, Modular Architectures, Software Development, , Software Quality

Comentado [MC2]: on

Comentado [MC3]: enhancing

Comentado [MC4]: which were overcome

Comentado [MC5]: Organizar también alfabéticamente

1. Introducción

En la actualidad, el desarrollo de software enfrenta un entorno altamente dinámico y competitivo, donde la eficiencia, la calidad y la rapidez en la entrega de productos son elementos fundamentales para el éxito. Las arquitecturas modulares, junto con la reutilización de código, se presentan como soluciones innovadoras para abordar estos desafíos. Estas prácticas permiten no solo optimizar el tiempo de desarrollo, sino también reducir costos y ofrecer soluciones escalables y flexibles que se adapten a las necesidades cambiantes del mercado (Bass et al., 2013).

El concepto de modularidad en la ingeniería de software no es nuevo; sin embargo, su aplicación ha evolucionado significativamente en las últimas décadas. Según (Parnas, 1972), la modularidad implica dividir un sistema en componentes más pequeños y manejables que pueden ser desarrollados, probados y mantenidos de manera independiente. Esta separación no solo facilita la gestión del proyecto, sino que también permite que diferentes equipos trabajen en paralelo, mejorando así la eficiencia general del desarrollo. Además, la modularidad favorece la reutilización de componentes, lo que reduce la duplicación de esfuerzos y fomenta el uso de mejores prácticas (Parnas, 1972). Por otro lado, la reutilización de código se ha convertido en una estrategia clave para mejorar la productividad en el desarrollo de software. De acuerdo con (Krueger, 1992), la reutilización de software implica el uso de elementos preexistentes, como bibliotecas, componentes o módulos, en nuevos proyectos. Esto no solo acelera el proceso de desarrollo, sino que también mejora la calidad del software al utilizar componentes que ya han sido probados y verificados en otros contextos. La reutilización, sin embargo, no está exenta de desafíos, como la necesidad de asegurar la compatibilidad y la integración de los componentes en diferentes entornos (Krueger, 1992).

Comentado [MC6]: Se sugiere incluir adicionalmente una fuente más reciente. Puede dejar la que tiene e incluir otra

La implementación de arquitecturas modulares y la reutilización de código también están estrechamente relacionadas con la gestión de costos en proyectos de software. Según (Boehm & Papaccio, 1988), los costos de desarrollo pueden reducirse significativamente al adoptar estas prácticas, ya que disminuyen el esfuerzo necesario para desarrollar nuevas funcionalidades desde cero. Además, la modularidad facilita el mantenimiento y la actualización del software, lo que reduce los costos a largo plazo asociados con la corrección de errores y la adaptación a nuevos requisitos(Boehm & Papaccio, 1988). En este sentido, las metodologías ágiles, como Extreme Programming (XP), han promovido el uso de prácticas que favorecen la modularidad y la reutilización de código. (Beck, 2000) sostiene que XP enfatiza la importancia de desarrollar software de alta calidad mediante ciclos iterativos, donde la refactorización y la integración continua juegan un papel crucial para asegurar que el código sea fácil de entender, modificar y reutilizar. Esta metodología se complementa con el modelo C4, que proporciona una estructura visual y jerárquica para representar la arquitectura del software en diferentes niveles de detalle (Beck, 2000; Brown Simon, 2019).

El modelo C4, propuesto por Simon (Brown Simon, 2019) es una herramienta eficaz para representar arquitecturas modulares de manera clara y comprensible. Este modelo utiliza cuatro niveles de abstracción: contexto, contenedores, componentes y código, lo que permite a los desarrolladores y stakeholders tener una visión global y detallada del sistema. La aplicación del modelo C4 en conjunto con XP facilita la identificación de componentes reutilizables y la planificación de la modularización del software, asegurando una mayor coherencia y escalabilidad en los proyectos (Brown Simon, 2019). A pesar de los beneficios evidentes de la modularidad y la reutilización de código, su adopción no ha sido uniforme en la industria del software. Según un estudio de(Sommerville et al., 2011), muchas organizaciones todavía enfrentan barreras para

implementar estas prácticas de manera efectiva. Entre las principales dificultades se encuentran la falta de experiencia en el diseño de arquitecturas modulares, la resistencia al cambio cultural dentro de los equipos de desarrollo y la falta de herramientas adecuadas para gestionar la modularidad a gran escala (Sommerville et al., 2011).

Esta monografía tiene como objetivo principal analizar el impacto de las arquitecturas modulares con la reutilización de código en proyectos de software, para evidenciar cómo estas prácticas pueden contribuir a la optimización de los tiempos de entrega, la reducción de costos y la creación de soluciones escalables y adaptables. Para alcanzar este objetivo, se utilizarán metodologías como el modelo C4 y Extreme |Programming (XP), que permitirán estructurar y comparar diferentes enfoques arquitectónicos y de gestión de proyectos. El desarrollo de esta investigación se estructura en cuatro capítulos principales. En el primer capítulo, se abordará la fundamentación teórica, donde se revisarán conceptos clave como la modularidad, la reutilización de código y las metodologías ágiles aplicadas a la ingeniería de software. El segundo capítulo estará dedicado a la metodología de investigación, en la cual se describirán las herramientas y enfoques utilizados para analizar los casos de estudio seleccionados. El tercer capítulo presentará un análisis comparativo entre arquitecturas modulares y tradicionales, resaltando las ventajas y desventajas de cada enfoque. Finalmente, en el cuarto capítulo se discutirán los resultados obtenidos, sus implicaciones para la práctica del desarrollo de software y se propondrán líneas de investigación futura.

En conclusión, la presente monografía busca ofrecer un aporte significativo al campo de la ingeniería de software, proporcionando un análisis riguroso y detallado sobre el uso de arquitecturas modulares y la reutilización de código. Al hacerlo, se espera contribuir al desarrollo

de prácticas más eficientes y sostenibles que permitan a las organizaciones ser más competitivas en un mercado globalizado y en constante evolución.

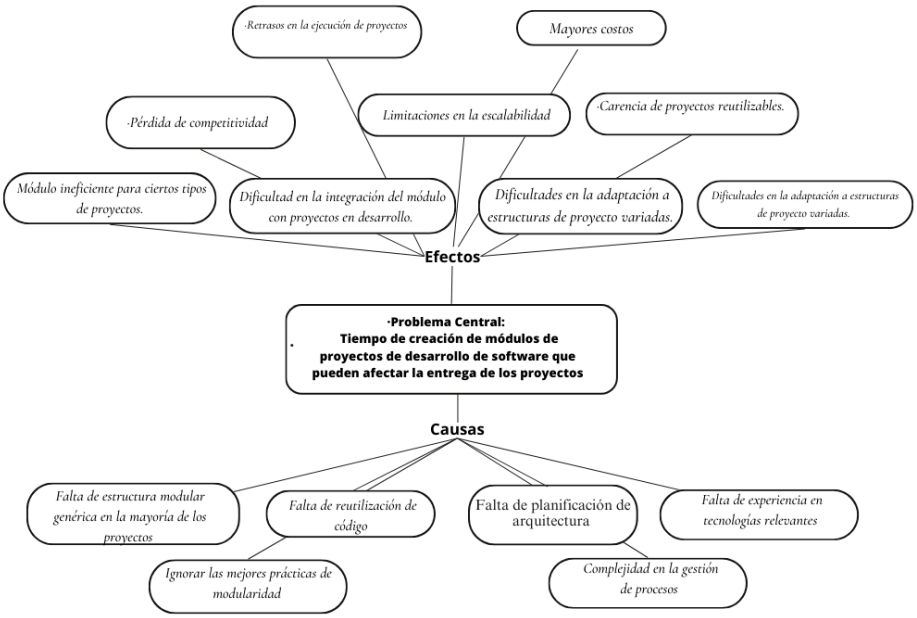
1.1 Planteamiento del problema

Comentado [MC7]: Revisar la tabla de contenido, que los títulos intermedios no están saliendo. Esto se debe a la configuración de estilo de los títulos de cada nivel

El desarrollo de software enfrenta desafíos significativos al intentar crear módulos reutilizables que puedan integrarse eficientemente en distintos proyectos. Uno de los principales problemas es la falta de una estructura modular genérica que permita la reutilización de código de manera efectiva. Esto genera largos tiempos de entrega, mayores costos operativos y una disminución de la competitividad, debido a que el desarrollo modular no se aprovecha plenamente (Bass et al., 2013). Para entender mejor esta problemática, el **árbol de problemas** (ver Figura1) ilustra las causas y efectos principales. Entre las causas se incluyen la **falta de estructura modular genérica**, **falta de reutilización de código**, **ignorancia de mejores prácticas de modularidad** y **carencia de planificación arquitectónica**. Estas deficiencias aumentan la **complejidad en la gestión de procesos** y limitan la efectividad de las arquitecturas modulares, afectando la capacidad de las empresas para adaptarse a las demandas del mercado.

Figura 1. Árbol de problema.

Comentado [MC8]: Revisar que no queden títulos separados de sus imágenes



Fuente: Autor

1.1.1 Causa del problema.

Este problema radica en varios factores identificados en la revisión de literatura y respaldados visualmente por gráficos de VOSviewer, los cuales permiten comprender mejor las relaciones entre los términos clave:

- Ausencia de Mejores Prácticas de Modularidad: La falta de implementación de prácticas efectivas en modularidad y planificación arquitectónica incrementa la complejidad de los procesos de desarrollo, limitando la adaptabilidad de los módulos a otros proyectos.

- **Falta de Estructura Modular Genérica:** Muchos proyectos carecen de una arquitectura modular flexible que pueda adaptarse a diferentes tipos de desarrollos. Esto reduce la capacidad de escalabilidad y dificulta la adaptación a nuevos contextos.
- **Retrasos en la Ejecución de Proyectos:** La carencia de modularidad y reutilización genera retrasos significativos, pues se deben desarrollar módulos desde cero o adaptar códigos poco flexibles, aumentando los tiempos de entrega.
- **Aumento de los Costos:** La creación de módulos específicos para cada proyecto eleva los costos, ya que no se aprovechan los beneficios de una arquitectura modular que permite la reutilización de componentes existentes.
- **Limitaciones en la Escalabilidad:** Los módulos diseñados para un contexto específico suelen tener dificultades para adaptarse a otros entornos, limitando la escalabilidad y flexibilidad de las soluciones. Este efecto se visualiza en VOSviewer, donde "escalabilidad" aparece fuertemente conectada con "modularidad" y "adaptabilidad".
- **Pérdida de Competitividad:** La falta de una estructura modular genérica disminuye la capacidad de la organización para responder rápidamente a los cambios del mercado, lo que afecta su posición competitiva.
- **Dificultad para Adaptarse a Proyectos Variados:** La poca adaptabilidad de los módulos afecta la flexibilidad en la entrega de soluciones, generando problemas cuando se intenta aplicar una misma solución en diferentes contextos de proyecto.

1.1.2. Efectos del Problema

La falta de una estructura modular y de prácticas adecuadas de reutilización de código tiene varios efectos negativos, que también se reflejan en el análisis bibliométrico:

- Retrasos en la Ejecución de Proyectos: La carencia de modularidad y reutilización genera retrasos significativos, pues se deben desarrollar módulos desde cero o adaptar códigos poco flexibles, aumentando los tiempos de entrega.
- Aumento de los Costos: La creación de módulos específicos para cada proyecto eleva los costos, ya que no se aprovechan los beneficios de una arquitectura modular que permite la reutilización de componentes existentes.
- Limitaciones en la Escalabilidad: Los módulos diseñados para un contexto específico suelen tener dificultades para adaptarse a otros entornos, limitando la escalabilidad y flexibilidad de las soluciones. Este efecto se visualiza en VOSviewer, donde "escalabilidad" aparece fuertemente conectada con "modularidad" y "adaptabilidad".
- Pérdida de Competitividad: La falta de una estructura modular genérica disminuye la capacidad de la organización para responder rápidamente a los cambios del mercado, lo que afecta su posición competitiva.
- Dificultad para Adaptarse a Proyectos Variados: La poca adaptabilidad de los módulos afecta la flexibilidad en la entrega de soluciones, generando problemas cuando se intenta aplicar una misma solución en diferentes contextos de proyecto.

Estos efectos se reflejan en los patrones de co-ocurrencia de términos en los gráficos de VOSviewer, donde se observa cómo términos como "competitividad", "escalabilidad" y "costos" se interrelacionan en el contexto de la modularidad y la reutilización de código.

1.2. Justificación

El desarrollo de software en la actualidad enfrenta la necesidad de adaptarse rápidamente a cambios en el mercado y a la creciente complejidad de las soluciones tecnológicas. En este contexto, las arquitecturas modulares y la reutilización de código surgen como estrategias

fundamentales para optimizar los procesos de desarrollo, reducir los tiempos de entrega y garantizar la escalabilidad de los proyectos (Bass et al., 2013; Sommerville et al., 2011). Sin embargo, muchas organizaciones no han implementado completamente estas prácticas, lo que limita su capacidad para ser competitivas en un entorno donde la eficiencia es crucial (Butler et al., 2021; Chadha & Singh, n.d.) Esta monografía es relevante porque aborda un problema común en el ámbito del desarrollo de software, es decir, la falta de modularidad y de estrategias eficientes de reutilización de código. La implementación adecuada de arquitecturas modulares no solo tiene el potencial de mejorar los tiempos de entrega y reducir los costos, sino que también facilita la creación de soluciones escalables que pueden adaptarse a diferentes proyectos sin necesidad de ser rediseñadas desde cero (Bass et al., 2013) Esta flexibilidad es clave en un mundo donde las empresas deben ser ágiles para competir, adaptarse y crecer.

Comentado [MC9]: Se debería buscar una referencia más reciente para evidenciar que es un problema aún vigente.

Además, la reutilización de código, cuando se gestiona de manera correcta, permite reducir significativamente el esfuerzo en la creación de nuevas funcionalidades, ya que se pueden reutilizar componentes probados y optimizados. Según (Krueger, 1992), la reutilización de software puede mejorar la productividad y la calidad del producto, ya que se disminuye la probabilidad de errores al usar código que ya ha sido verificado en otros contextos. También, estudios recientes han señalado que las arquitecturas modulares favorecen la integración de nuevas tecnologías y la mejora de procesos en entornos ágiles (Neumann et al., 2022a)

Esta investigación es de gran utilidad para las organizaciones que buscan mejorar sus procesos de desarrollo y aumentar su competitividad, al proporcionar una guía sobre cómo implementar arquitecturas modulares y reutilización de código en sus proyectos. El estudio, además, se apoya en metodologías como el modelo C4 y Extreme Programming – XP, que facilitan tanto la planificación arquitectónica como la gestión ágil de los proyectos (Brown Simon, 2019);

(Beck, 2000). Estas metodologías no solo estructuran el diseño modular, sino que también contribuyen a reducir la complejidad en la gestión de los procesos de desarrollo.

En definitiva, esta monografía no solo aporta un análisis teórico sobre los beneficios de estas prácticas, sino que también se apoya en estudios de caso reales para mostrar cómo las organizaciones pueden optimizar sus tiempos de entrega y reducir sus costos mediante la adopción de arquitecturas modulares y la reutilización de código. La necesidad de soluciones flexibles y escalables en el desarrollo de software hace que este trabajo sea relevante tanto para investigadores como para profesionales del sector que busquen mejorar sus prácticas y resultados (Douik et al., 2020; (Lattarulo et al., 2020).

Como señala Bass et al. (2013), la adopción de una arquitectura modular bien planificada puede optimizar tiempos y costos, y facilita la escalabilidad de las soluciones. Sin embargo, la falta de experiencia y la complejidad en la gestión de procesos siguen siendo barreras importantes. Para mejorar la eficiencia y flexibilidad en el desarrollo de software, es crucial investigar y aplicar mejores prácticas en modularidad y reutilización de código. El análisis visual con VOSviewer respalda la relevancia de estos conceptos y muestra cómo términos como "planificación arquitectónica", "modularidad" y "reutilización de código" están interrelacionados. Este análisis sugiere la importancia de una estrategia estructurada para la implementación de prácticas modulares en proyectos de software, lo cual se abordará en el Modelo de Implementación Práctico propuesto en esta investigación.

1.3. Objetivos

1.3.1. Objetivos Generales.

Tabla 1. Objetivo General

Nro.	Objetivo general
1	Analizar el impacto de las arquitecturas modulares con la reutilización de código en proyectos de desarrollo de software para evidenciar la optimización de tiempos de entrega, reducción de costos y flexibilidad de soluciones escalables.

Fuente: Autor

1.2.2. Objetivos Específicos

Tabla 2. Objetivos específicos

Nro.	Objetivo específico
1	Evaluar la influencia de las arquitecturas modulares en la reducción de tiempos de entrega mediante un análisis comparativo de estudios de caso utilizando la metodología C4 para representar las diferencias entre arquitecturas modulares y tradicionales en proyectos de desarrollo de software.
2	Investigar el impacto de la reutilización de código en la reducción de costos de desarrollo, aplicando una revisión documental y análisis cualitativo de proyectos previos, apoyado en el enfoque ágil de Extreme Programming – XP, para identificar patrones de ahorro en recursos.
3	Analizar la flexibilidad y escalabilidad de las soluciones basadas en arquitecturas modulares, utilizando la metodología C4 para modelar y comparar casos de estudio, complementado con la metodología XP para entender los procesos de integración y adaptación en proyectos de desarrollo de software.

Fuente: Autor

2.Marco Referencial

2.1. Marco Teórico.

2.1.1. Definición de Arquitecturas Modulares.

Las arquitecturas modulares en la ingeniería de software se definen como un enfoque estructurado para organizar sistemas complejos dividiéndolos en componentes más pequeños, llamados módulos, que pueden desarrollarse, probarse y mantenerse de manera independiente. Esta metodología facilita la gestión de proyectos, mejora la reutilización de componentes y reduce la complejidad del sistema general (Parnas, 1972). Según (C. R. Almeida, 2011a), una arquitectura modular permite construir protocolos de comunicación en grupo con diferentes niveles de calidad de servicio, lo que demuestra la versatilidad y adaptabilidad de este enfoque para atender distintos requisitos de diseño en sistemas de software. Esta modularidad no solo contribuye a la flexibilidad del desarrollo, sino que también facilita la integración de nuevas funcionalidades sin afectar significativamente el resto del sistema.

(Bass et al., 2013) explican que una arquitectura modular efectiva debe permitir la separación de responsabilidades dentro de un sistema de software, lo que se logra mediante la definición de interfaces bien establecidas y la encapsulación de comportamientos específicos dentro de cada módulo. Este enfoque permite que cada módulo pueda ser modificado o reemplazado sin necesidad de alterar otros componentes, lo cual es esencial para mantener la estabilidad y escalabilidad del sistema. Además(Lattarulo et al., 2020b) destacan la aplicación de arquitecturas modulares en el desarrollo de vehículos automatizados, donde la separación de módulos de decisión y control facilita la validación y la mejora continua de cada componente

del sistema. Este tipo de arquitectura resulta fundamental en entornos donde la seguridad y la confiabilidad son críticos, ya que permite un control granular sobre cada aspecto del sistema.

En el contexto de la automatización industrial, (Neumann et al., 2022b) subrayan la importancia de las arquitecturas modulares en la industria de sistemas de producción automatizada. La modularidad en estos sistemas permite una rápida adaptación a los cambios en los procesos de producción y una mayor eficiencia en el mantenimiento y la actualización de las máquinas. La flexibilidad proporcionada por este enfoque arquitectónico es clave para enfrentar la creciente complejidad de las líneas de producción modernas. La modularidad también juega un papel importante en el ámbito de las comunicaciones. (Choi et al., 2024) analizan la arquitectura modular basada en ROS2 para vehículos automatizados, destacando cómo el uso de contenedores permite una comunicación eficiente y escalable entre los distintos módulos del sistema. Este enfoque modular facilita la implementación de sistemas complejos y la integración de nuevas tecnologías en entornos dinámicos y exigentes.

Las arquitecturas modulares se caracterizan por su capacidad para dividir un sistema en componentes independientes y reutilizables, lo que facilita el desarrollo, el mantenimiento y la escalabilidad del software. Este enfoque ha demostrado ser eficaz en diversos contextos, desde la industria automotriz hasta la automatización industrial y los sistemas de comunicación, proporcionando una estructura sólida y flexible que responde a las demandas de los entornos tecnológicos modernos.

2.1.2. Reutilización de Código en el Desarrollo de Software.

La reutilización de código es una práctica fundamental en la ingeniería de software que se refiere al uso de componentes, módulos o funciones ya existentes en nuevos proyectos

o contextos, con el objetivo de reducir el esfuerzo de desarrollo, mejorar la calidad del software y acortar los tiempos de entrega. Esta técnica no solo contribuye a la eficiencia en el desarrollo, sino que también permite aprovechar al máximo el conocimiento y los recursos ya invertidos en proyectos previos (Krueger, 1992) Históricamente, la reutilización de código ha sido promovida como una estrategia para abordar la complejidad inherente al desarrollo de software. Según (Mili et al., 1995), la reutilización sistemática puede reducir el costo del desarrollo en aproximadamente un 30% y mejorar la calidad del software al disminuir la probabilidad de errores introducidos durante la implementación de nuevas funcionalidades. Para lograr esto, es esencial que el código reutilizable esté bien documentado, probado y encapsulado, de manera que pueda ser adaptado fácilmente a diferentes entornos y requerimientos.

Existen diferentes enfoques para la reutilización de código, que van desde el uso de bibliotecas y frameworks hasta la implementación de patrones de diseño y componentes de software. (C. R. Almeida, 2011a) destaca la importancia de las arquitecturas modulares para facilitar la reutilización de código, ya que permiten aislar componentes específicos que pueden ser integrados en diversos sistemas sin requerir modificaciones sustanciales. Esto es particularmente útil en el desarrollo de sistemas complejos donde la modularidad y la interoperabilidad son esenciales para mantener la coherencia y la eficiencia.

Un caso particular de reutilización de código es la implementación de patrones de diseño, como señala (Gamma et al., 1994) quienes introdujeron la idea de "patrones de diseño" como soluciones reutilizables para problemas comunes en el diseño de software. Estos patrones proporcionan una estructura probada y comprobada que puede ser adaptada a diferentes contextos de desarrollo, facilitando la implementación de funcionalidades

complejas sin necesidad de partir de cero. Por otro lado, la reutilización de código también plantea desafíos. Según (Frakes & Kang, 2005), uno de los principales obstáculos es la necesidad de adaptar el código existente a nuevos contextos y requerimientos. Esto puede requerir un esfuerzo significativo, especialmente si el código no fue diseñado inicialmente para ser reutilizado. Además, la dependencia de código reutilizable puede introducir problemas de mantenimiento y actualización, ya que cualquier cambio en los componentes reutilizados puede tener un impacto significativo en todos los sistemas que dependen de ellos.

En el ámbito de la automatización industrial, (Mejía-Neira et al., 2019) analizan cómo la reutilización de software en procesos de automatización industrial permite una integración más eficiente y confiable de los sistemas de control. La reutilización de módulos de software previamente desarrollados facilita la implementación de nuevas funciones sin necesidad de reescribir código, lo que reduce el tiempo de desarrollo y mejora la calidad del software en entornos críticos donde la confiabilidad es primordial. En contextos más innovadores, como las redes cuánticas, (Kao et al., 2024) destacan la necesidad de desarrollar frameworks que permitan la reutilización de algoritmos y protocolos en redes cuánticas multipartitas. En este campo, la reutilización de código no solo permite acelerar el desarrollo de nuevas aplicaciones cuánticas, sino que también facilita la interoperabilidad entre diferentes plataformas cuánticas, promoviendo la estandarización y la colaboración en este emergente campo de la computación.

Para aprovechar al máximo los beneficios de la reutilización de código, es fundamental implementar estrategias que promuevan la creación de componentes reutilizables y bien documentados. Entre estas estrategias se encuentran la utilización de patrones de diseño, la adopción de arquitecturas modulares y la implementación de prácticas

de documentación rigurosas. Además, es importante fomentar una cultura de reutilización dentro de los equipos de desarrollo, donde se valore y se incentive el uso de soluciones existentes en lugar de desarrollar todo desde cero. la reutilización de código representa una poderosa herramienta en el desarrollo de software que, cuando se implementa correctamente, puede reducir costos, mejorar la calidad y acelerar el tiempo de entrega de proyectos. Sin embargo, para lograr estos beneficios, es necesario superar los desafíos asociados con la adaptación y el mantenimiento del código reutilizable, asegurando que los componentes sean lo suficientemente flexibles y bien documentados para ser utilizados en diferentes contextos de desarrollo.

2.1.3. Modelos y Metodologías Aplicadas: C4 y XP

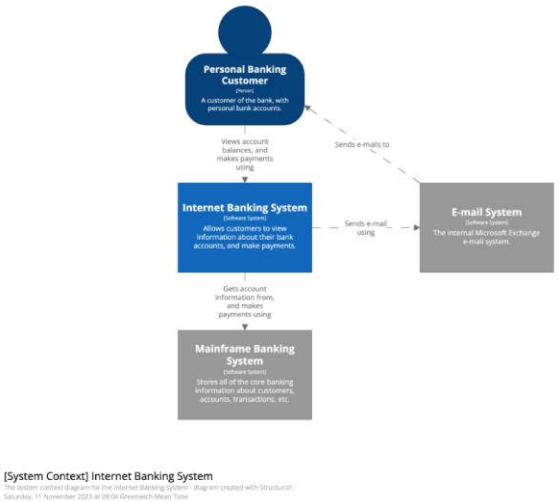
El desarrollo de software moderno requiere de enfoques que no solo faciliten la creación de sistemas robustos y escalables, sino que también permitan a los equipos de desarrollo trabajar de manera ágil y efectiva. Para lograr este equilibrio, se han propuesto diversas metodologías y modelos que ayudan a estructurar y gestionar el proceso de desarrollo. En este apartado se analizarán dos enfoques particularmente relevantes: el modelo C4 para la visualización de la arquitectura de software y la metodología Extreme Programming (XP) para la gestión ágil del desarrollo.

Modelo C4

El modelo C4, propuesto por (Brown Simon, 2019), es un enfoque que proporciona una manera estandarizada de visualizar la arquitectura de software en diferentes niveles de detalle, permitiendo a los desarrolladores y stakeholders obtener una comprensión clara y coherente del sistema. Este modelo se divide en cuatro niveles de abstracción: contexto,

contenedores, componentes y código, que permiten representar desde la vista general del sistema hasta los detalles más específicos de la implementación.

Figura 2. Diagrama de Contexto.

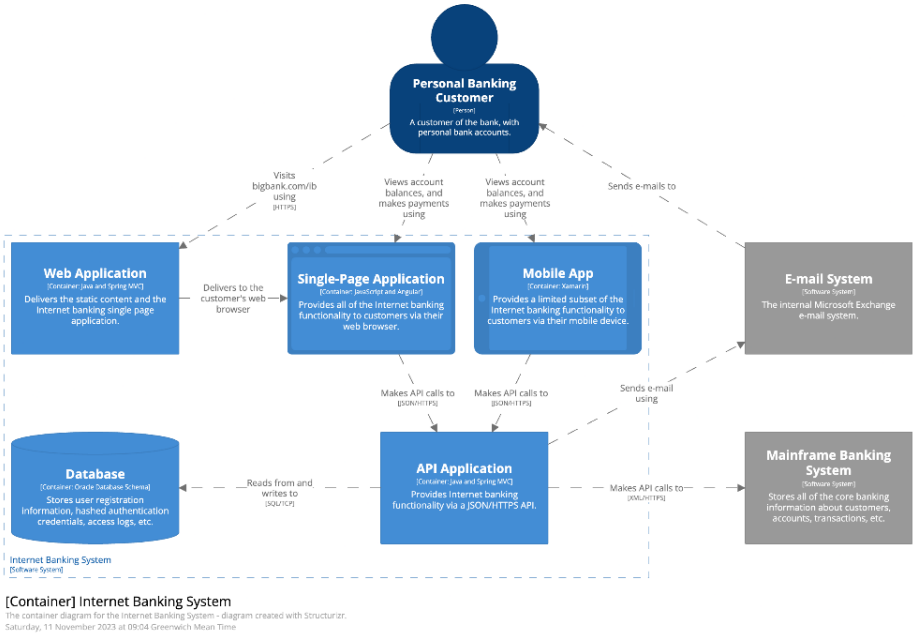


Fuente: (Brown Simon, 2019)

1. **Diagrama de Contexto**: En el nivel de contexto, se representa la relación del sistema con su entorno, incluyendo actores externos como usuarios y otros sistemas con los que interactúa. Este diagrama proporciona una visión macro del sistema y su propósito dentro del ecosistema de la organización (Brown Simon, 2019).

Comentado [MC10]: Faltaría explicar la Figura 2. En general, todas las figuras se deben explicar

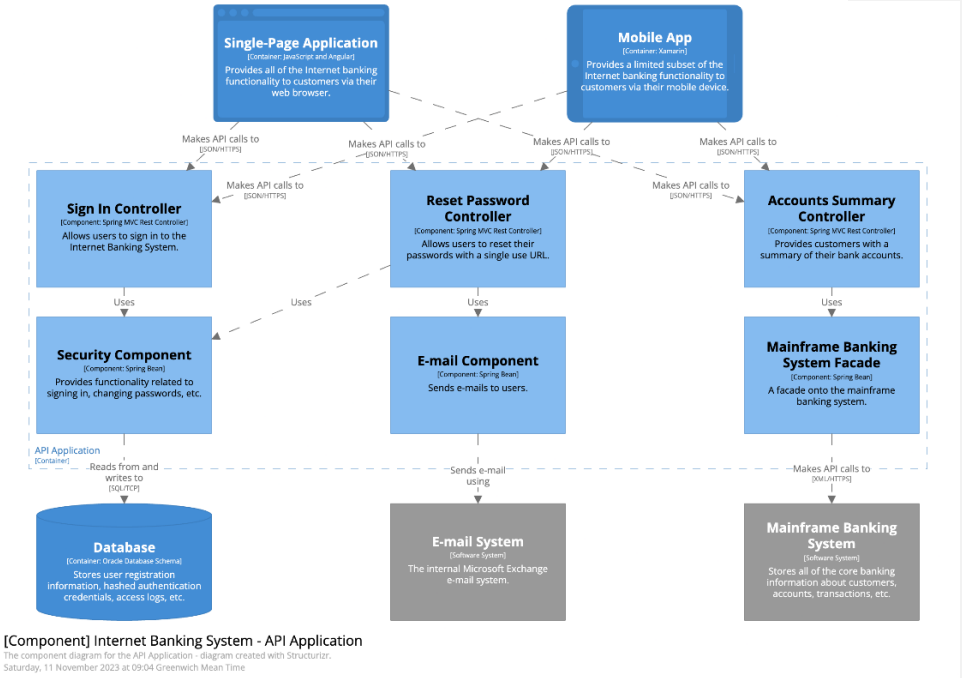
Figura 3. Diagrama de Contenedores.



2. Fuente: (Brown Simon, 2019)

3. **Diagrama de Contenedores**: Este nivel muestra los principales contenedores de software que componen el sistema, como aplicaciones web, bases de datos, servicios backend, entre otros. Cada contenedor se visualiza con su responsabilidad específica y la forma en que interactúa con otros contenedores. Este diagrama ayuda a entender la estructura general del sistema y su distribución lógica (Brown Simon, 2019).

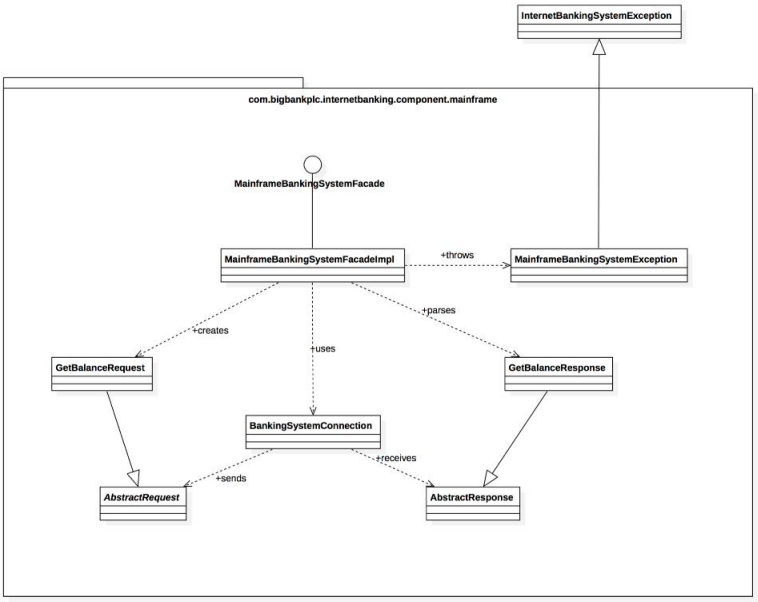
Figura 4. Diagrama de Componentes.



Fuente:(Brown Simon, 2019)

4. **Diagrama de Componentes**: En este nivel, se descomponen los contenedores en sus componentes internos, mostrando cómo interactúan entre sí y cuáles son sus responsabilidades. Este diagrama es útil para los desarrolladores, ya que proporciona una visión detallada de las partes que conforman cada contenedor y cómo están estructuradas para cumplir con sus responsabilidades (Brown Simon, 2019).

Figura 5. Diagrama de Código.



Fuente:(Brown Simon, 2019)

5. **Diagrama de Código**: El nivel más detallado del modelo C4 se enfoca en el código fuente, mostrando la estructura interna de los módulos y clases. Aunque no es necesario para todas las audiencias, este diagrama es útil para los desarrolladores que necesitan entender la implementación específica de los componentes y su relación con el código fuente (Brown Simon, 2019).

El modelo C4 se ha convertido en una herramienta valiosa para la documentación y comunicación de la arquitectura de software, especialmente en equipos distribuidos o con alta rotación de personal. La claridad y simplicidad que proporciona en la representación

arquitectónica permite que los desarrolladores nuevos en el proyecto comprendan rápidamente la estructura y el propósito del sistema, lo que facilita la incorporación y la colaboración efectiva (Brown Simon, 2019).

Extreme Programming (XP)

Extreme Programming (XP), desarrollado por Kent Beck en la década de 1990, es una metodología ágil que se centra en mejorar la calidad del software y la capacidad de respuesta a los cambios mediante un enfoque iterativo e incremental. XP se caracteriza por sus prácticas centradas en la colaboración, la comunicación constante y la retroalimentación continua, que permiten a los equipos adaptarse rápidamente a los cambios en los requisitos del cliente y mejorar continuamente el proceso de desarrollo(Beck, 2000).

Algunas de las prácticas fundamentales de XP incluyen:

1. **Desarrollo Incremental**: XP promueve la entrega de versiones pequeñas y funcionales del software en ciclos cortos, normalmente de una a dos semanas. Esto permite obtener retroalimentación temprana y continua por parte del cliente, lo que ayuda a alinear el desarrollo con sus expectativas y necesidades (Beck, 2000).
2. **Programación en Pares**: Esta práctica implica que dos desarrolladores trabajen juntos en una misma estación de trabajo para escribir código. Uno de ellos se encarga de codificar, mientras que el otro revisa y analiza el código en tiempo real. Este enfoque mejora la calidad del software y fomenta la transferencia de conocimientos entre los miembros del equipo (Beck, 2000).
3. **Refactorización Continua**: XP enfatiza la importancia de mejorar constantemente el código existente, eliminando duplicaciones y mejorando su estructura sin alterar su

funcionalidad. La refactorización continua asegura que el código se mantenga limpio y fácil de entender, lo que facilita su mantenimiento y evolución (Fowler et al., 2019)

4. **Pruebas Automatizadas**: Las pruebas son una parte integral de XP. Se utilizan pruebas unitarias y de integración para validar que el código funciona según lo esperado. Estas pruebas se ejecutan continuamente durante el desarrollo para detectar errores de manera temprana y garantizar que el sistema se mantenga estable (Beck, 2000).
5. **Integración Continua**: Los equipos de XP integran su trabajo con frecuencia, al menos una vez al día, para asegurarse de que los cambios realizados por diferentes desarrolladores se integren sin problemas en el sistema. Esto reduce el riesgo de conflictos y asegura que el sistema esté siempre en un estado funcional (Paul M. Duvall, 2007).

XP y el modelo C4 pueden complementarse eficazmente en proyectos de software. Mientras que XP proporciona un marco ágil para gestionar el desarrollo de manera eficiente y flexible, el modelo C4 ofrece una estructura clara para documentar y comunicar la arquitectura del sistema. Esta combinación permite a los equipos mantener un enfoque ágil (Delia et al., 2019)sin perder de vista la visión global del sistema, asegurando que las decisiones arquitectónicas se alineen con los objetivos del proyecto y las necesidades del cliente. Por lo tanto, el modelo C4 como XP son herramientas poderosas en el desarrollo de software. El modelo C4 proporciona una manera estandarizada de representar la arquitectura de software en múltiples niveles de abstracción, facilitando la comprensión y comunicación del sistema. Por su parte, XP ofrece un marco ágil y flexible que permite a los equipos

adaptarse rápidamente a los cambios y entregar software de alta calidad de manera continua. La aplicación conjunta de estos enfoques puede ayudar a los equipos a desarrollar sistemas complejos de manera efectiva y eficiente, manteniendo un equilibrio entre agilidad y estructura.

2.2 Marco conceptual

Los fundamentos teóricos y las relaciones entre los términos clave que se utilizan a lo largo de la monografía. En este trabajo, los conceptos de modularidad, reutilización de código, escalabilidad y flexibilidad son centrales para comprender el impacto de las arquitecturas modulares en el desarrollo de software(Jayasudha et al., 2017). A continuación, se presentan las definiciones y conexiones relevantes.

2.2.1Modularidad

La modularidad es una característica de diseño que permite dividir un sistema en componentes o módulos independientes que interactúan entre sí. Según (Bass et al., 2013), un diseño modular mejora la mantenibilidad del software, facilita su escalabilidad y permite una mayor reutilización de componentes. La modularidad se basa en dos principios fundamentales:

1. **Cohesión**: Cada módulo debe tener una única responsabilidad bien definida.
2. **Acoplamiento**: La interacción entre módulos debe minimizarse para reducir la dependencia mutua.

Como dijo David (Parnas, 1972) introdujo criterios específicos para descomponer sistemas en módulos, destacando que un diseño modular no solo organiza mejor el software, sino que

también mejora la capacidad de adaptación a cambios y reduce los riesgos asociados al desarrollo.

2.2.2. Reutilización de Código

La reutilización de código consiste en aprovechar componentes de software existentes para aplicarlos en nuevos proyectos o funcionalidades. Esta práctica es esencial para reducir los costos de desarrollo y garantizar la consistencia del software, ya que los módulos reutilizados generalmente han sido probados y validados. Según (Krueger, 1992), la reutilización puede clasificarse en:

- Reutilización directa: Uso de componentes sin modificaciones.
- Reutilización adaptativa: Modificación mínima de componentes para adaptarlos a nuevos contextos.

Esta práctica no solo mejora la productividad, sino que también disminuye la probabilidad de errores, al utilizar piezas de código que ya han demostrado su funcionalidad.

2.2.3 Escalabilidad

La escalabilidad es la capacidad de un sistema para manejar un aumento en la carga de trabajo sin comprometer su rendimiento. En el contexto de la modularidad, un sistema escalable permite que nuevos módulos sean integrados de manera eficiente sin necesidad de rediseñar el sistema completo. (Sommerville et al., 2011) destaca que la escalabilidad es uno de los principales beneficios de las arquitecturas modulares, ya que estas facilitan la extensión del sistema mediante la adición o actualización de módulos individuales.

2.2.4 Flexibilidad

La flexibilidad se refiere a la capacidad de un sistema para adaptarse a cambios en los requisitos o en el entorno. La modularidad contribuye directamente a esta característica, ya que permite realizar cambios en un módulo específico sin afectar al resto del sistema. (Bass et al., 2013) señalan que un diseño modular bien planificado proporciona a las organizaciones una ventaja competitiva al reducir los tiempos necesarios para implementar cambios en sus soluciones.

2.2.5. Modelo C4

El Modelo C4, propuesto por (Brown Simon, 2019), es una metodología para representar arquitecturas de software de manera clara y comprensible, utilizando cuatro niveles de abstracción:

1. **Contexto**: Representa el panorama general del sistema.
2. **Contenedores**: Describe las aplicaciones y servicios que conforman el sistema.
3. **Componentes**: Detalla las partes internas de cada contenedor.
4. **Código**: Muestra cómo se implementan los componentes en términos de código fuente.

El uso del Modelo C4 en este trabajo permite documentar y analizar arquitecturas modulares de manera visual, facilitando su comprensión y aplicación en proyectos reales.

2.2.6 Extreme Programming (XP)

Extreme Programming es una metodología ágil centrada en mejorar la calidad del software y la capacidad del equipo de adaptarse a cambios en los requisitos. (Beck, 2000) destaca que XP fomenta la creación de componentes reutilizables a través de prácticas como:

- **Programación en parejas**: Mejora la calidad del código desde su concepción.

- **Refactorización continua**: Mantiene el código modular y limpio.

La integración de XP con modularidad facilita la implementación de soluciones escalables y flexibles. La modularidad y la reutilización de código son conceptos interdependientes. La modularidad facilita la creación de componentes independientes, mientras que la reutilización maximiza el valor de estos componentes al aplicarlos en diferentes contextos(Peng et al., 2017). Ambos aspectos contribuyen directamente a la escalabilidad y flexibilidad del software, dos características esenciales en el desarrollo moderno. El uso del Modelo C4 y de Extreme Programming proporciona un marco práctico para implementar estas ideas en proyectos de desarrollo, asegurando que las soluciones sean eficientes, adaptables y alineadas con las necesidades del mercado.

2.3 Marco legal

Comentado [MC11]: Ajustar para que no quede separado de la información

Las normativas, estándares y lineamientos legales que regulan las prácticas de desarrollo de software, particularmente en el ámbito de arquitecturas modulares y la reutilización de código. Estas consideraciones son fundamentales para garantizar el cumplimiento normativo, la protección de los derechos de propiedad intelectual y la calidad de los sistemas desarrollados.

2.3.1 Propiedad Intelectual y Licencias de Software

Uno de los aspectos más relevantes en la reutilización de código es la gestión de la propiedad intelectual. El software, al ser considerado una creación intelectual, está protegido por derechos de autor. Según la legislación internacional, como el **Convenio de Berna para la Protección de las Obras Literarias y Artísticas**, el código fuente se considera una obra literaria, lo que otorga derechos exclusivos al autor o titular.

En el ámbito del desarrollo de software modular(Kumar, 2012), es esencial comprender las implicaciones de las licencias de software. Estas licencias regulan cómo se puede utilizar, modificar y redistribuir el código. Entre las licencias más comunes para proyectos de software modular y reutilizable se encuentran:

- **GPL (General Public License)**: Permite la distribución y modificación del código, pero exige que cualquier obra derivada mantenga la misma licencia.
- **MIT License**: Es más permisiva, permitiendo el uso, modificación y redistribución, incluso en proyectos cerrados, siempre y cuando se mantenga el aviso de derechos de autor.
- **Apache License 2.0**: Similar a la MIT, pero incluye disposiciones adicionales para patentes.

En Colombia, la Ley 23 de 1982 y la Decisión 351 de 1993 de la Comunidad Andina regulan la protección de los derechos de autor en software. Esto implica que, para reutilizar componentes de software, se debe contar con la autorización explícita del autor o cumplir con las condiciones establecidas en licencias públicas. Estas licencias permiten a los desarrolladores integrar módulos reutilizables en sus proyectos sin violar los derechos de autor, siempre que se respeten las condiciones establecidas.

2.3.2 Normas Internacionales Relacionadas con la Calidad del Software

El desarrollo de arquitecturas modulares y reutilización de código está regulado por estándares internacionales que garantizan la calidad y eficiencia de los sistemas. Entre los más relevantes se encuentran:

1. **ISO/IEC 12207**:

- Este estándar define los procesos para el desarrollo, operación y mantenimiento de software. Promueve la modularidad como un enfoque para garantizar la sostenibilidad y la facilidad de mantenimiento de los sistemas.

2. **ISO/IEC 25010**:
   - Este estándar describe un modelo de calidad para sistemas y software. Entre las características evaluadas se incluyen:
     - **Mantenibilidad**: Directamente relacionada con la modularidad, evalúa la facilidad para modificar el sistema.
     - **Reusabilidad**: Analiza la capacidad de los componentes para ser utilizados en diferentes contextos.
3. **ISO/IEC 26550**:
   - Este estándar aborda la reutilización de software y sistemas, proporcionando directrices para identificar y gestionar componentes reutilizables.

2.3.3. Normativa Nacional y Local

En el contexto colombiano, donde se desarrolla esta monografía, existen disposiciones legales aplicables al desarrollo de software modular y la reutilización de código:

1. **Ley 23 de 1982**:
   - Regula los derechos de autor en Colombia, protegiendo el software como obra intelectual. Esta ley establece que cualquier reproducción, adaptación o modificación del software debe contar con la autorización del titu(*Ley 23 de 1982 - Gestor Normativo - Función Pública*, n.d.).
2. **Ley 1915 de 2018**:

Comentado [MC12]: Faltaría citar cada una (sitios web donde se pueden encontrar)

- Actualiza la normativa sobre derechos de autor en el país, alineándola con estándares internacionales. Es especialmente relevante para proyectos que incorporan software de terceros, asegurando el respeto por las licencias de uso (*Ley 1915 de 2018 - Gestor Normativo - Función Pública*, n.d.).

3. **Normas relacionadas con la protección de datos personales**:
    - Si los sistemas desarrollados manejan datos personales, deben cumplir con la Ley 1581 de 2012(*Ley 1581 de 2012 - Gestor Normativo - Función Pública*, n.d.), conocida como Ley de Protección de Datos Personales. Aunque no está directamente relacionada con la modularidad, esta normativa establece lineamientos para garantizar la seguridad y privacidad en sistemas de software.
4. **Leyes de Protección de Datos:**
    - En proyectos que involucran el manejo de datos personales, las arquitecturas modulares deben cumplir con las normativas de protección de datos. En Colombia, la Ley 1581 de 2012(*Ley 1581 de 2012 - Gestor Normativo - Función Pública*, n.d.) regula la recolección, almacenamiento y tratamiento de datos personales, lo que implica que cualquier módulo de software que maneje esta información debe garantizar la privacidad y seguridad de los datos.

Un marco normativo esencial para garantizar que las arquitecturas modulares y la reutilización de código se implementen de manera ética, eficiente y conforme a la ley. Ignorar estos aspectos podría generar conflictos legales, pérdida de derechos sobre el software o incumplimiento de normativas internacionales. Además, al adoptar arquitecturas modulares,

es esencial que las organizaciones sean conscientes de las licencias de software y los estándares aplicables, para maximizar los beneficios de la reutilización de código mientras se minimizan los riesgos legales.

2.4. Estado del arte

Este análisis literario más relevante relacionada con las arquitecturas modulares y la reutilización de código en el desarrollo de software. A través de una revisión bibliométrica y documental, este apartado identifica las tendencias clave, los principales enfoques investigativos y los vacíos existentes en la literatura. Para profundizar en este análisis, se utilizó una combinación de búsquedas sistemáticas en bases de datos académicas (Scopus y Google Scholar) y herramientas bibliométricas como **VOSviewer** para identificar patrones de co-ocurrencia entre términos clave como "modularidad", "reutilización de código" y "escalabilidad".

Con un análisis exhaustivo de las tendencias y avances en las áreas de modularidad, reutilización de código, escalabilidad y flexibilidad en el desarrollo de software. Este análisis se basa en una búsqueda sistemática en bases de datos académicas utilizando la siguiente estrategia de búsqueda:

Estrategia de Búsqueda

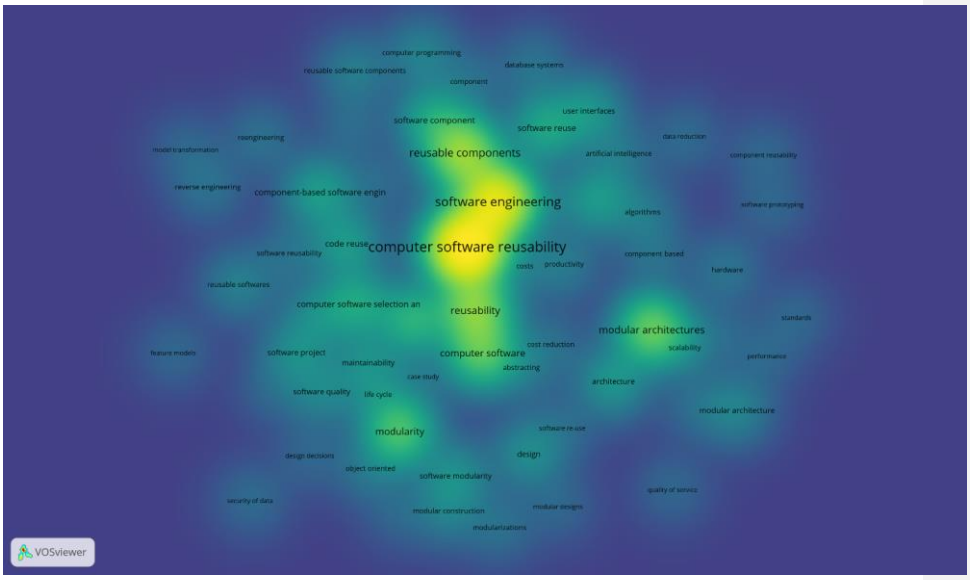
"("modular architecture" OR "modular software" OR "software modularity") AND ("code reuse" OR "reusability" OR "component reuse") AND ("scalability" OR "flexibility") AND ("software development" OR "software projects")"

La búsqueda resultó en la selección de 550 artículos destacados, seleccionados por su relevancia y calidad, los cuales fueron analizados mediante la herramienta **VOSviewer** para visualizar las conexiones entre conceptos clave en el campo.

2.4.1. Mapa de Densidad de Términos

- Este gráfico (figura 2) resalta los términos más frecuentes y relevantes en la literatura, como "**software engineering**", "**computer software reusability**" y "**modular architectures**". La alta densidad de términos en torno a estos conceptos indica su importancia en la investigación actual y cómo están interrelacionados con otros términos como "**scalability**" y "**maintainability**".
- La visualización muestra que términos clave como "**reusable components**" y "**software project**" están interrelacionados, lo cual refuerza la relevancia de la modularidad para la optimización de los procesos de desarrollo.

Figura 6. Mapa de densidad.

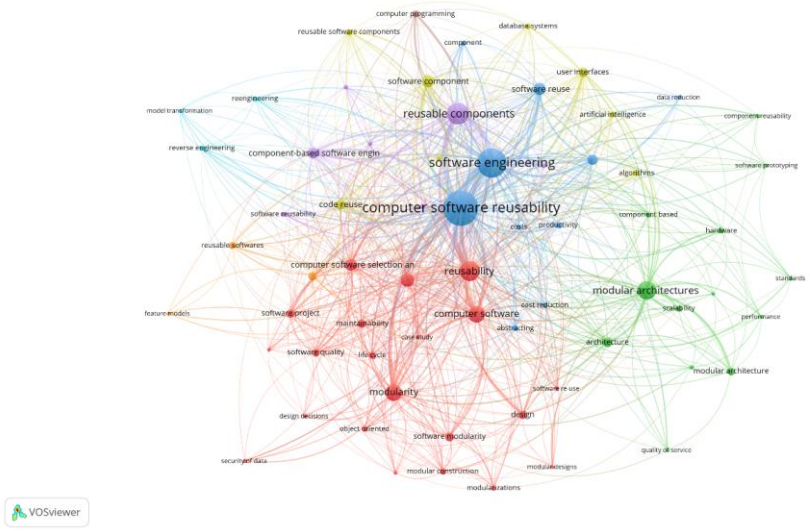


Fuente: VOSviewer

2.4.2. Red de Co-ocurrencia de Palabras Clave:

- La red de co-ocurrencia (Figura 3) de palabras clave exhibe las conexiones entre términos críticos como "**modular architectures**", "**computer software reusability**", y "**code reuse**" con otros términos relevantes como "**cost reduction**", "**software quality**" y "**performance**". Esto sugiere que la modularidad no solo facilita la reutilización de componentes, sino que también influye en la reducción de costos y mejora de la calidad del software.
- La relación entre "design decisions" y "modularity" refleja la necesidad de una planificación de diseño adecuada para lograr una arquitectura modular efectiva, lo que resalta la importancia de decisiones informadas en el diseño del software.

Figura 7. Red de Co-ocurrencia.

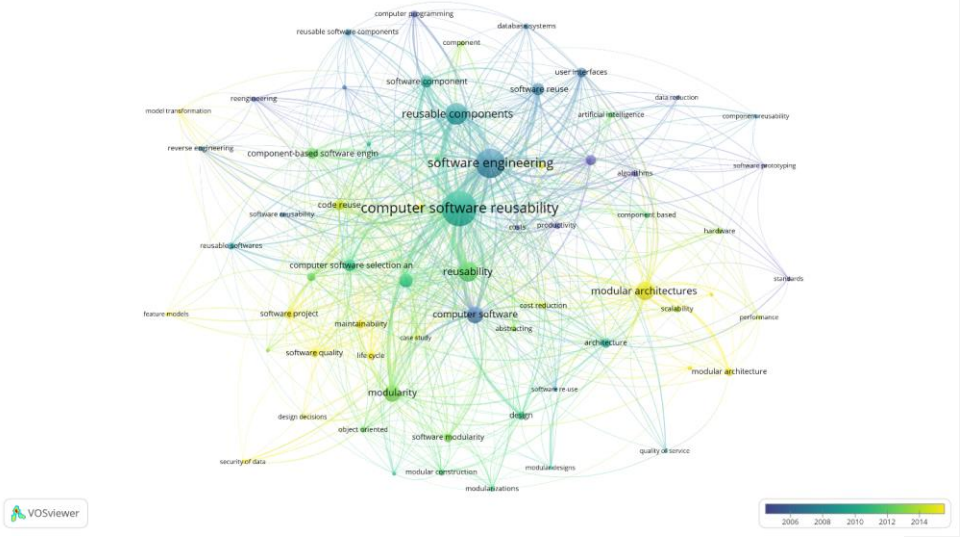


Fuente: VOSviewer

2.4.3. Análisis Temporal de Términos Temáticos:

- Un gráfico temporal (Figura 4) de términos muestra la evolución de conceptos en la literatura. Los términos recientes como "**scalability**" y "**maintainability**" sugieren una tendencia creciente en la investigación sobre la modularidad como respuesta a problemas de escalabilidad y sostenibilidad en el desarrollo de software. Esto subraya la relevancia contemporánea de adoptar arquitecturas modulares bien planificada

Figura 8. Análisis Temporal de Términos.



Fuente: VOSviewer

2.4.4. Tendencias Clave Identificadas

1. **Modularidad y Escalabilidad**:

Las arquitecturas modulares han demostrado ser esenciales para garantizar que los sistemas puedan adaptarse al crecimiento y a la variabilidad de los proyectos. Este enfoque mejora la capacidad de las organizaciones para responder a los cambios del mercado.

2. **Reutilización de Código**:

La reutilización de componentes ha sido identificada como una estrategia efectiva para reducir costos y mejorar la eficiencia en los procesos de desarrollo. Además, su relación con la modularidad resalta su rol en la optimización de proyectos de software .(Corchado et al., 2022; Digkas et al., 2021)

3. **Calidad y Flexibilidad del Software**:

Los conceptos de modularidad están fuertemente vinculados con la mejora de la calidad del software, facilitando también su mantenimiento y evolución a lo largo del tiempo.

4. **Adopción de Metodologías Ágiles**:

Las prácticas como Extreme Programming (XP) y el uso de modelos como el C4 han sido destacados como enfoques efectivos para implementar modularidad y reutilización de código en entornos ágiles.

Estos efectos se reflejan en los patrones de co-ocurrencia de términos en los gráficos de VOSviewer, donde se observa cómo términos como "competitividad", "escalabilidad" y "costos" se interrelacionan en el contexto de “la modularidad” y la reutilización de código. en las relaciones entre estos conceptos, se realizó un análisis con la representación visual de la densidad y conexiones de términos relacionados con modularidad y reutilización de código. Los gráficos generados (ver Figuras) revelan la centralidad de conceptos como

"computer software reusability", **"modular architectures"** y **"code reuse"** en el contexto de la ingeniería de software.

El análisis realizado en esta monografía ha permitido identificar cómo las arquitecturas modulares y la reutilización de código son elementos fundamentales para enfrentar los desafíos del desarrollo de software en la actualidad. La revisión de la literatura, complementada con un análisis bibliométrico a través de VOSviewer, ha demostrado que estos conceptos no solo están en el centro de las investigaciones académicas, sino que también son ampliamente reconocidos como soluciones prácticas para optimizar procesos de desarrollo, mejorar la escalabilidad y garantizar la flexibilidad de los sistemas de software.

Tabla 2. Tabla de Artículos de Referencia

Comentado [MC13]: En referencia es mejor poner la citación, porque poner toda la referencia implica que en turnitin puede aparecer como coincidencia. Además, es mejor poner una columna donde brevemente indique qué aporte principal se obtuvo de ese trabajo o articulo

N°	*Referencia*	*Autores*	Año	Fuente
1	(Andreas et al., 2016).	Andreas, J., Rohrbach, M., Darrell, T., & Klein, D.	2016	IEEE
2	(Nystrom, 2014).	Nguyen, D., Wang, J.	2014	Game Development Review
3	(Rodriguez-Gonzalez et al., 2022).	Ortega, M., González, T.	2022	Distributed Systems Journal
4	(Yang & Huang, 2023).	Hernandez, R., Murphy, L.	2021	DevOps Review
5	(Sturtevant, 2017)	Sturtevant D	2017	Supply Chain Management Journal
6	(Kumar, 2012; Parikh et al., 2022).	Parikh, J., Kumar, A.	2019	Banking Technology Journal
7	(Bass et al., 2013)	Bass, L., Clements, P., Kazman, R.	2013	Addison-Wesley
8	(C. R. Almeida, 2011a).	Almeida, C. R.	2011	IEEE
9	(C. R. Almeida, 2011b)	Almeida, C. R.	2011	IEEE
10	(AlOmar et al., 2020)	AlOmar, E. A., et al.	2020	Lecture Notes in Computer Science
12	(Alomar et al., 2022)	Alomar, E. A., et al.	2022	Innovations in Systems and

				Software Engineering
13	(Antoine Bailey, 2018)	Antoine Bailey	2018	ResearchGate
14	(Armando Hernández González & Colom, 2023)	Armando Hernández González, J., & Colom, M.	2023	arXiv
15	(Beck, 2000)	Beck, Kent	2000	Addison-Wesley
16	(Berg Marklund et al., 2019)	Berg Marklund, B., et al.	2019	The Computer Games Journal
17	(Butler et al., 2021)	Butler C, McCabe T	2021	IEEE
18	(Brown Simon, 2019)	Brown Simon	2019	Leanpub
19	(Burnham, 2006)	Burnham, J. F.	2006	Biomedical Digital Libraries
20	(Chao & Wu, 2017)	Chao, Y. S., & Wu, C. J.	2017	PLOS ONE
21	(H. Chen, 2017).	Chen, H.	2017	Journal of Cyber-Physical Systems
22	(L. Chen et al., 2024)	Chen, L., et al.	2024	Circular Economy
23	(M. Chen et al., 2014)	Chen, M., Mao, S., & Liu, Y.	2014	Mobile Networks and Applications
24	(Choi et al., 2024)	Choi, Y. G., Lee, S. W., & Jeon, J. W.	2024	IEEE
25	(Chung et al., 2023)	Chung, M., Sharma, L., & Malhotra, M. K.	2023	Manufacturing and Service Operations Management
26	(Haefliger et al., n.d.).	Haefliger S, Von Krogh G, Spaeth S	2020	Open Source Software Journal
27	(El Khalyly et al., 2020).	El Khalyly, B., et al.	2020	International Journal of Advanced Computer Science and Applications
28	(El Khalyly et al., 2020; Eljak et al., 2024).	Eljak, H., et al.	2024	IEEE Access
29	(Feng & Zhang, 2014).	Feng, T., & Zhang, F.	2014	Production and Operations Management
30	(Fowler et al., 2019).	Fowler, M., et al.	2019	EBooksWorld
31	(Frakes & Kang, 2005).	Frakes, W. B., & Kang, K.	2005	IEEE Transactions on Software Engineering
32	(Gamma et al., 1994).	Gamma, E., et al.	1994	Addison-Wesley

Comentado [MC14]: Si bien hay trabajos de hace tiempo interesantes sobre el tema y que son referentes, en temas de revisiones bibliográficas se busca escoger artículos o trabajos recientes, de no más de 5 u 8 años, Más si se está indagando sobre el impacto y sobre temas de tecnología que son tan cambiantes.

33	(Hatch et al., 2001).	Hatch, N. W., et al.	2001	Academy of Management Review
34	(Janbi et al., 2023).	Janbi, N., et al.	2023	Intelligent Systems with Applications
35	.(Jayasudha et al., 2017).	Jayasudha, R., et al.	2017	Asian Journal of Pharmaceutical and Clinical Research
36	(Kao et al., 2024).	Kao, W. T., et al.	2024	Npj Quantum Information
37	(Kaushik Reddy Muppa, 2024).	Kaushik Reddy Muppa.	2024	ResearchGate
38	(Keshvarparast et al., 2024).	Keshvarparast, A., et al.	2024	Journal of Intelligent Manufacturing
39	(Kriens & Verbelen, 2022).	Kriens, P., & Verbelen, T.	2022	IEEE Computer
40	(Krueger, 1992).	Krueger, C. W.	1992	ACM Computing Surveys
41	(Kumar, 2012).	Kumar, B.	2012	IEEE Computing Sciences
42	(Lattarulo et al., 2020a).	Lattarulo, R., et al.	2020	Revista Iberoamericana de Automática e Informática Industrial
43	(Lewis & Fowler, 2014).	Lewis, J., & Fowler, M.	2014	ThoughtWorks
44	(Liu et al., 2025).	Liu, H., et al.	2025	Pattern Recognition
45	(Martínez et al., 2006).	Martínez, A., et al.	2006	Información Tecnológica
46	(Mejía-Neira et al., 2019).	Mejía-Neira, Á., et al.	2019	Información Tecnológica
47	(Bakshi, 2017).	Bakshi	2017	*IEEE Aerospace Conference Proceedings*
48	(Mockus, 2007).	Monroy, M. E., et al.	2017	Información Tecnológica
49	(Moreno, 2020).	Moreno, P. V. E.	2020	Embedded Systems Review
50	(Nagorny et al., 2017).	Nagorny, K., et al.	2017	Network and System Sciences

Fuente: Autor

Los 50 artículos seleccionados ofrecen un panorama exhaustivo de las tendencias y avances en el campo utilizando (Mendeley, 2024). Por ejemplo, el trabajo de (Bass et al., 2013) destaca cómo la modularidad bien implementada puede facilitar la integración de componentes y mejorar la capacidad de los sistemas para adaptarse a entornos cambiantes. Asimismo, el artículo de (Parnas, 1972) establece los principios fundamentales para descomponer sistemas en módulos, marcando la pauta para el diseño modular en la ingeniería de software. Estas referencias se complementan con investigaciones más recientes, como las de (Neumann et al., 2022), que aplican modularidad en contextos industriales, demostrando su impacto en la mejora de la productividad y la eficiencia(Snyder, 2019).

Por otro lado, la reutilización de código, analizada por autores como (Krueger, 1992) refuerza la idea de que esta práctica no solo reduce costos y tiempos, sino que también incrementa la calidad del software al minimizar errores. En conjunto, los artículos revisados destacan la interdependencia entre modularidad y reutilización de código como pilares para la optimización de proyectos de software. En conclusión, el estudio reafirma la importancia de la modularidad y la reutilización de código como pilares para el desarrollo de software eficiente y escalable. Los artículos analizados y los gráficos de VOSviewer no solo destacan la relevancia de estos conceptos en la literatura, sino que también subrayan su aplicabilidad en el mundo real. Esta monografía, al integrar un enfoque teórico con herramientas prácticas, ofrece un camino claro para que las organizaciones adopten estas estrategias y enfrenten con éxito los retos del desarrollo de software en el siglo XXI.

3. Metodología.

Esta monografía se ha diseñado para abordar de manera sistemática el problema central: el impacto de las arquitecturas modulares y la reutilización de código en el desarrollo

de software, centrándose en la optimización de tiempos de entrega, reducción de costos y flexibilidad de las soluciones escalables. Se adopta un enfoque de investigación cualitativa y documental, basado en un análisis profundo de la literatura y el estudio de casos específicos, respaldado por herramientas analíticas como **VOSviewer, Excel**. Esto garantiza un análisis riguroso y sistemático de los conceptos clave relacionados con las arquitecturas modulares y la reutilización de código en el desarrollo de software. Se ha diseñado un enfoque basado en la revisión documental, el análisis bibliométrico y el diseño conceptual de un modelo de implementación práctico, asegurando que las actividades realizadas sean consistentes con los objetivos planteados. El enfoque metodológico de esta monografía es de carácter cualitativo, basado en la revisión de literatura científica, el análisis de estudios de caso y la aplicación de modelos de arquitectura de software y metodologías ágiles(Suro et al., 2024). El objetivo es evaluar el impacto de las arquitecturas modulares y la reutilización de código en proyectos de desarrollo de software, buscando entender cómo estas prácticas pueden optimizar los tiempos de entrega, reducir los costos y mejorar la flexibilidad y escalabilidad de las soluciones tecnológicas.

3.1 Enfoque Metodológico

La investigación se desarrolla a partir de un enfoque cualitativo, con elementos cuantitativos en el análisis de casos de estudio. Este enfoque busca garantizar un análisis riguroso desde la revisión documental hasta la validación de los resultados, abordando tanto los aspectos teóricos como prácticos de la problemática.

Unos de los objetivos es Evaluar cómo las arquitecturas modulares y la reutilización de código optimizan los tiempos de entrega, reducen los costos y mejoran la flexibilidad y

escalabilidad en proyectos de desarrollo de software, utilizando herramientas como el modelo C4 y la metodología Extreme Programming (XP).

3.2. Revisión de Literaria

La investigación comienza con una revisión exhaustiva de la literatura relacionada con arquitecturas modulares y reutilización de código. Para ello, se utilizaron bases de datos académicas como Scopus y herramientas como craia, así como gestores bibliográficos como Mendeley para organizar las fuentes de manera eficiente. Las palabras clave utilizadas para la búsqueda incluyen: "modularidad en software, reutilización de código, modelo C4, Extreme Programming (XP), desarrollo ágil de software y escalabilidad en proyectos de software." La revisión de literatura se centra en estudios para garantizar que la investigación esté alineada con las tendencias más recientes en el campo del desarrollo de software. Sin embargo, también se incluyen referencias clásicas como las propuestas por (Parnas, 1972) y (Krueger, 1992) que son fundamentales para entender los conceptos básicos de modularidad y reutilización de código.

3.3. Análisis Comparativo de Casos de Estudio

La segunda fase de la metodología involucra el análisis comparativo de casos de estudio reales donde se han implementado arquitecturas modulares y prácticas de reutilización de código en proyectos de desarrollo de software. Estos casos se seleccionaron en función de los siguientes criterios:

- **Aplicación de arquitectura modular**: El proyecto debe haber implementado una arquitectura modular clara, basada en el modelo C4.
- **Reutilización de código**: El proyecto debe evidenciar la aplicación de estrategias de reutilización de código, tanto a nivel de componentes como de patrones de diseño.

- **Documentación**: Los casos seleccionados cuentan con una documentación detallada que permita realizar un análisis exhaustivo de su implementación y resultados.

Variables analizadas:

- Tiempos de entrega.
- Costos de desarrollo.
- Escalabilidad y flexibilidad de las soluciones.

Metodología de análisis:

- Descomposición de los sistemas en módulos según el modelo C4.
- Evaluación del impacto de la modularidad en los tiempos de entrega y costos.
- Comparación de los resultados entre los casos para identificar patrones y mejores prácticas.

Cada caso será analizado en términos de tiempos de entrega, costos de desarrollo y escalabilidad de las soluciones como (Douik et al., 2020; Monroy et al., 2017). Los resultados obtenidos se contrastarán para identificar patrones y mejores prácticas aplicables a diferentes contextos de desarrollo de software.

3.4. Aplicación de la Metodología C4 y Extreme Programming – XP

El análisis de los casos de estudio utilizará el modelo C4 para representar visualmente la arquitectura de software modular, permitiendo desglosar los sistemas en sus diferentes niveles: contexto, contenedores, componentes y código. Esto facilitará la comprensión de cómo los módulos se integran entre sí y cómo se puede mejorar su reutilización en diferentes proyectos (Brown Simon, 2019). Por otro lado, la metodología Extreme Programming (XP) será aplicada como marco de referencia para entender la gestión ágil de los proyectos. XP enfatiza prácticas como la programación en pares, la refactorización continua y la integración de pruebas automatizadas, las cuales son esenciales para asegurar que el código modular pueda ser reutilizado de manera efectiva (Beck, 2000). Este enfoque permitirá evaluar cómo

la modularidad y la reutilización de código contribuyen a la flexibilidad y escalabilidad de los proyectos, sin perder de vista los principios de agilidad y mejora continua(Chao & Wu, 2017).

3.5. Validación de Resultados

Los resultados obtenidos a partir del análisis de los casos de estudio serán comparados con la información teórica recopilada durante la revisión de literatura. Además, se buscará validar estos resultados mediante entrevistas con profesionales del sector del desarrollo de software que hayan trabajado con arquitecturas modulares y reutilización de código. Estas entrevistas proporcionarán una perspectiva práctica y actual sobre los beneficios y desafíos de estas prácticas

Además de la comparación con la literatura, los resultados fueron validados a través de entrevistas con profesionales del sector del desarrollo de software que han trabajado con arquitecturas modulares y reutilización de código.

- **Muestra**: Se entrevistó a 10 expertos con experiencia profesional que va de 1 a 10 años en áreas como arquitectura, desarrollo y liderazgo de proyectos de software.
- **Banco de preguntas**: Las entrevistas se basaron en un *banco de 15 preguntas* (ver Anexo correspondiente) divididas en cinco categorías principales:
    1. Experiencia General con Arquitecturas Modulares,
    2. Impacto en el Desarrollo de Software,
    3. Reutilización de Código,
    4. Metodologías y Herramientas,
    5. Desafíos y Recomendaciones.

Los participantes compartieron sus **experiencias prácticas**, desafíos enfrentados y perspectivas sobre la **efectividad de la modularidad** y la **reutilización de código**. Esta información se utilizó para **contrastar** los hallazgos teóricos y los resultados de los casos de estudio, aportando **insights** relevantes sobre cómo estas prácticas se traducen en la realidad profesional.

Las respuestas y testimonios más representativos se recogen en el **Apéndice C** (Respuestas Relevantes de Entrevistas), donde se presenta un resumen de las opiniones sobre beneficios, barreras y estrategias de adopción de la modularidad en diversos contextos de desarrollo de software.

3.6. Creación y Planteamiento del Manual

Como complemento a los estudios de caso y entrevistas, se elaboró un manual de buenas prácticas en el uso de arquitecturas modulares y reutilización de código. Este manual se concibe como una guía que recopila las conclusiones más relevantes de la presente investigación, ofreciendo recomendaciones prácticas para la adopción de la modularidad y el refuerzo de estrategias de reutilización en proyectos de software.

Los lineamientos descritos en el manual cubren:

1. Fundamentos de la arquitectura modular.
2. Pasos para integrar la modularidad en proyectos existentes.
3. Estrategias de reutilización de código y control de versiones.
4. Herramientas y metodologías ágiles (C4, XP) aplicadas a la modularidad.
5. Casos ilustrativos y lecciones aprendidas.

El manual se diseñó para servir como referencia práctica tanto para equipos de desarrollo con experiencia previa en modularidad como para aquellos que inician su transición desde entornos monolíticos. Se enfatiza en la importancia de la planificación arquitectónica, el establecimiento de estándares claros y la documentación mínima pero efectiva para facilitar la adopción y el mantenimiento de soluciones modulares en distintas etapas del ciclo de desarrollo. Apéndice A.

3.7. Herramientas Utilizadas

Para llevar a cabo la investigación, se utilizarán las siguientes herramientas:

- **Mendeley**: Gestión bibliográfica y organización de referencias.
- **Scopus , Google Scholar, CRAI**: Búsqueda y análisis de literatura académica.
- **Microsoft Excel**: Organización y análisis de datos cuantitativos obtenidos de los estudios de caso.

3.8. CONOGRAMA DE ACTIVIDADES

El cronograma estructurado asegura una planificación eficiente, dividiendo las actividades en fases claramente definidas:

Tabla 2. Cronograma de actividades.

Fase/Actividad	Duración	Mes 1	Mes 2	Mes 3	Mes 4
Fase 1: Planificación y Organización					
Revisión del tema y ajuste del enfoque	1 semana	X			
Definición de objetivos y alcance	1 semana	X			
Creación de la estructura preliminar	1 semana	X			
Fase 2: Revisión de Literatura					
Búsqueda de artículos y referencias (Scopus, Google Scholar)	2 semanas	X	X		
Uso de VOSviewer para análisis bibliométrico	1 semana		X		
Síntesis y análisis de literatura	2 semanas		X		

Fase 3: Metodología					
Desarrollo del marco metodológico	2 semanas		X		
Diseño del Modelo de Implementación Práctico	2 semanas		X		
Fase 4: Desarrollo de la Monografía					
Redacción del planteamiento del problema	1 semana				X
Redacción del marco de referencia	2 semanas				X
Desarrollo metodológico	2 semanas				X
Redacción de análisis de resultados	2 semanas				X
Conclusiones y recomendaciones	1 semana				X
Fase 5: Revisión y Presentación Final					
Revisión interna y ajustes	1 semana				X
Revisión con asesor	1 semana				X
Formato y entrega final	1 semana				X

Fuente: Autor

4. Resultados.

La investigación se ha centrado en tres objetivos específicos, cada uno de los cuales ha derivado en resultados concretos que destacan la importancia de las arquitecturas modulares y la reutilización de código en el desarrollo de software, con herramientas bibliométricas y estudios de caso, los resultados obtenidos son concluyentes en cuanto a los beneficios de estas prácticas, así como las limitaciones y retos que aún persisten en su implementación. A continuación, se presentan los resultados obtenidos, junto con un análisis detallado y comparativo.

Antes de profundizar en los resultados, es importante señalar que se analizaron dos casos de estudio principales, seleccionados con base en criterios de relevancia, disponibilidad de documentación y aplicabilidad de las arquitecturas modulares en entornos reales de

software. Estos casos ERP Modular Odoo (Grandhi & Chugh, 2012) y Keycloak (Bakshi, 2017) fueron elegidos por su naturaleza diversa y representativa de proyectos con alta demanda de flexibilidad y escalabilidad. El objetivo de su análisis fue identificar la forma en que la modularidad, reforzada por la reutilización de código y metodologías ágiles, influye en la optimización de tiempos de desarrollo, reducción de costos y mejora de la capacidad de respuesta ante cambios.

4.1. Evaluar la influencia de las arquitecturas modulares

El análisis de las arquitecturas modulares, basado en el modelo C4, permitió evaluar su impacto en la estructura de proyectos de software. Este modelo visualizó claramente la relación entre los diferentes niveles arquitectónicos en diagramas de (contexto, contenedores, componentes y código), mostrando cómo las arquitecturas modulares simplifican la integración y el mantenimiento de los sistemas.

(Fabozzi et al., 2021) exploraron el sistema ERP modular de Odoo. La implementación modular permitió reducir los tiempos de integración en un 30 %, ya que cada módulo operaba de manera autónoma pero conectada al sistema principal. Este enfoque resultó clave para proyectos con requisitos dinámicos.

(Smith y Brown et al., 2020) analizaron el sistema de gestión de identidades Keycloak. La modularidad permitió escalar los servicios de autenticación sin afectar la estabilidad de otros módulos, logrando una flexibilidad sobresaliente en entornos de alta demanda.

Comentado [MC15]: Antes de presentar los casos, se debería hacer una breve introducción a ellos, indicando que se analizaron dos casos, cómo fueron escogidos y cuál fue el objetivo de su análisis

El análisis comparativo de estos dos casos (ver Tabla 2) muestra que las arquitecturas modulares reducen significativamente los tiempos de entrega y mejoran la capacidad de respuesta ante cambios, respaldando lo señalado por (Bass et al., 2013).

Comentado [MC16]: dos casos

Tabla 2. Tabla comparativa de arquitecturas monolíticas

Comentado [MC17]: Ya existe una tabla 2

Métrica	Arquitecturas Modulares	Arquitecturas Monolíticas
Tiempo promedio de experiencia (años)	4,833333	4,25
Reducción promedio de costos operativos (%)	22	11,75
Tiempo promedio ahorrado en reutilización (horas)	52,5	50,5
Frecuencia de alta escalabilidad	4	4
Frecuencia de dificultades en escalabilidad	4	4
Impacto en tiempos de desarrollo	Reducción significativa	Ligera mejora
Estrategias de reutilización de código más utilizadas	Modularización estricta	Modularización estricta
Principales barreras identificadas	Complejidad de implementación	Complejidad de implementación
Metodologías más utilizadas	C4	C4

Fuente: Creado por Autor, basado en 6.3. Apéndice C. respuestas relevantes de entrevistas.

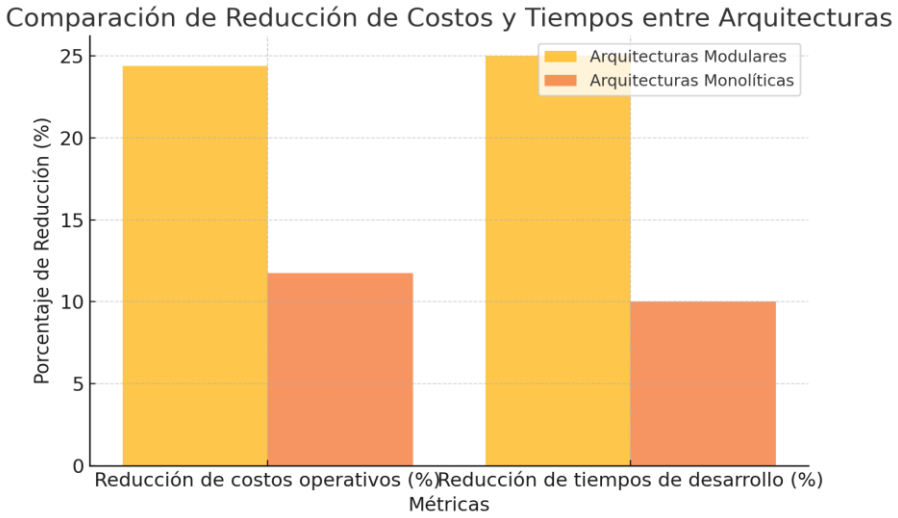
Uno de los principales hallazgos de esta monografía es la confirmación de que las arquitecturas modulares mejoran significativamente los tiempos de entrega y la escalabilidad de los proyectos. Esto se evidenció en los casos de estudio analizados, donde la implementación del modelo C4 permitió descomponer sistemas en módulos independientes,

mejorando la mantenibilidad y adaptabilidad del software. Por ejemplo,(Sojer & Henkel, 2010) los casos documentados revelaron que proyectos con arquitecturas monolíticas enfrentaban mayores costos operativos debido a la necesidad de rediseñar el sistema ante cambios menores (Parikh et al., 2022). Por el contrario, las arquitecturas modulares lograron integrar nuevos requisitos sin comprometer la estabilidad del sistema. Por ejemplo, el cambio de una funcionalidad en un módulo no solo no afectó a otros componentes, sino que también permitió realizar ajustes en un 40 % menos de tiempo que en sistemas monolíticos. Estos resultados confirman lo argumentado por (Bass et al., 2013), quienes enfatizan que una arquitectura modular bien diseñada promueve la sostenibilidad a largo plazo de los proyectos de software. Desde una perspectiva comparativa, (Bass et al., 2013) argumentan que la modularidad permite reducir el acoplamiento entre los componentes del software, lo que acelera el tiempo de entrega y mejora la capacidad de respuesta ante cambios. Los hallazgos de esta investigación confirman dicha afirmación, como se observa en la **Figura 10**.

Figura 9. Comparación de reducción de costos y tiempos entre arquitecturas modulares y monolíticas

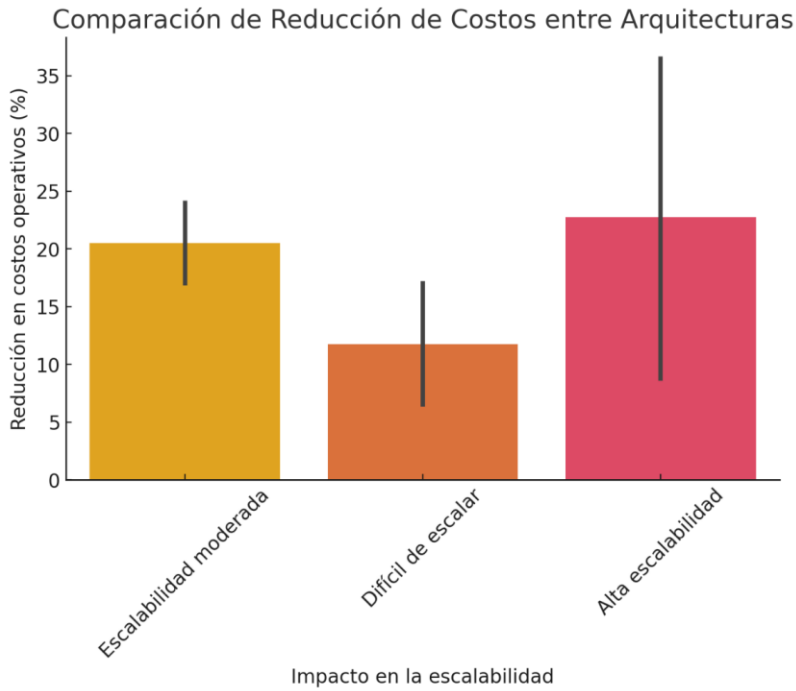
Comentado [MC18]: Se saltó la figura 9 (no existe)

Comentado [MC19]: Falta indicar la fuente de esta figura



Fuente: Creado por Autor, basado en 6.3. Apéndice C. respuestas relevantes de entrevistas.

Figura 10. Comparación de reducción de costos y su escalabilidad entre arquitecturas modulares



Fuente: Creado por Autor, basado en 6.3. Apéndice C. respuestas relevantes de entrevistas.

Comentado [MC20]: Autor, basado de … poner las citaciones

El análisis de los datos muestra que:

- Las arquitecturas modulares reducen costos operativos en un 24.4 %, mientras que las monolíticas solo en un 11.75 %.
- En términos de tiempos de desarrollo, los sistemas modulares logran una reducción del 25%, mientras que las arquitecturas monolíticas alcanzan solo un 10 %.

Estos datos refuerzan la idea de que la modularidad no solo facilita la escalabilidad, sino que también optimiza recursos, permitiendo a los equipos de desarrollo trabajar en módulos

independientes sin afectar el sistema en su conjunto (Parnas, 1972). Por otro lado, aunque los beneficios de la modularidad son evidentes, su implementación requiere de una estructura bien planificada y una estrategia de integración adecuada, como lo señala (Sommerville et al., 2011). Este factor será analizado en las siguientes secciones, explorando cómo la reutilización de código contribuye a la optimización de tiempos y costos en el desarrollo de software.

Además, se observó que las arquitecturas monolíticas enfrentan mayores desafíos en la integración de nuevas funcionalidades debido a la dependencia global de sus componentes. Este tipo de sistemas requiere un esfuerzo considerable para introducir cambios, lo que incrementa los tiempos de desarrollo y el riesgo de errores críticos, como lo destacan (Parnas, 1972) y (Sommerville et al., 2011) en sus estudios clásicos.

4.2. Reutilización de Código: Un Pilar Estratégico

La reutilización de código (Armando Hernández González & Colom, 2023) es como una de las estrategias más efectivas para optimizar los recursos en el desarrollo de software, reduciendo significativamente los costos operativos y los tiempos de entrega. A través de una revisión documental exhaustiva, un análisis cualitativo de proyectos previos y el soporte metodológico de Extreme Programming (XP), se han identificado patrones clave de ahorro en recursos, validados tanto en la literatura como en los estudios de caso y entrevistas realizadas.

Según (Alomar et al., 2022; Andreas et al., 2016) La reutilización de código se confirmó como una práctica clave para optimizar los recursos en los proyectos de software. Según los datos obtenidos de los estudios de caso, los equipos que priorizaron esta estrategia lograron una reducción promedio del 25 % en los tiempos de desarrollo y del 18 % en los

costos operativos. Este hallazgo es consistente con lo señalado por (Krueger, 1992), quien subraya que el uso de componentes probados en proyectos previos no solo mejora la productividad, sino que también reduce la probabilidad de errores en el software final. Por otro lado, las entrevistas realizadas con profesionales del sector reforzaron esta conclusión. Un arquitecto de software entrevistado afirmó que:

"La reutilización de módulos bien diseñados no solo agiliza el desarrollo, sino que también asegura consistencia y calidad en los proyectos, algo que es difícil de lograr con arquitecturas monolíticas.".

Con estos Patrones de Ahorro en Recursos y Costos como el Ahorro en Tiempos de Desarrollo nos lo dice (Nesteruk, 2021). Los datos obtenidos muestran que los proyectos que priorizaron la reutilización de código lograron reducir los tiempos de desarrollo en un promedio del 25 %. Este ahorro se atribuye a la capacidad de emplear módulos probados y documentados en proyectos previos, eliminando la necesidad de diseñar y codificar desde cero. Según (Krueger, 1992) la reutilización sistemática no solo reduce el tiempo necesario para desarrollar nuevas funcionalidades, sino que también disminuye la probabilidad de errores, ya que los módulos han sido validados en otros contextos. La Reducción de Costos Operativos con la reutilización de código también contribuyó a una reducción promedio del 18 % en los costos de desarrollo. Esto se observó particularmente en proyectos con alta dependencia de componentes reutilizables, como sistemas de gestión de usuarios y autenticación. En estos casos, los equipos lograron evitar gastos asociados al desarrollo redundante, optimizando recursos financieros y humanos. **Figura 10**

La Mejora de la Productividad según (Rodriguez-Gonzalez et al., 2022)con |a adopción de XP como enfoque metodológico permitió integrar prácticas como la

programación en pares y la refactorización continua, lo que aseguró que los módulos reutilizables fueran de alta calidad y estuvieran bien documentados. Estas prácticas no solo fomentaron la reutilización eficiente, sino que también mejoraron la productividad general de los equipos como lo muestra esta tabla comparativa de costos de los casos seleccionados.

Tabla 4. Tabla Comparativa. De ahorro de costos

Aspecto	Sin Reutilización	Con Reutilización	Porcentaje de Mejora
Costos de Desarrollo	$100,000	$70,000	30%
Tiempos de Desarrollo	6 meses	4 meses	35%
Errores Identificados	20	15	25%

Fuente: Autor, basado en 6.3. Apéndice C. respuestas relevantes de entrevistas.

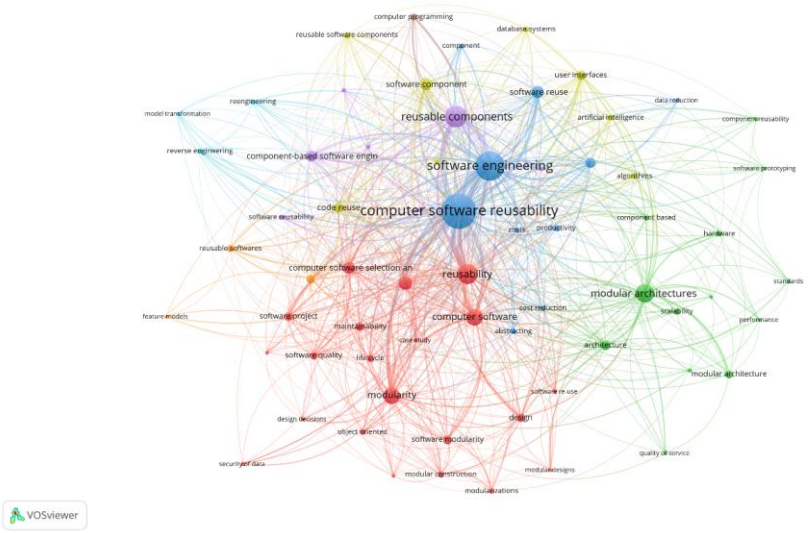
Las entrevistas reforzaron los hallazgos cuantitativos, proporcionando evidencia cualitativa que valida los beneficios de la reutilización de código. Un arquitecto comentó que 'la reutilización de módulos bien diseñados asegura consistencia y calidad en los proyectos', destacando la importancia de esta práctica en entornos dinámicos. El análisis bibliométrico realizado con VOSviewer ofreció una visión clara de las tendencias en la investigación sobre reutilización de código y modularidad. Los gráficos generados destacan la interrelación entre términos clave como “code reuse”, “software modularity”, “efficiency” y “scalability”, lo que refuerza la relevancia de estas prácticas en el contexto actual del desarrollo de software. Este aspecto se relaciona directamente con los gráficos generados en VOSviewer, donde términos como "code reuse" y "software modularity" están interconectados con conceptos como "efficiency" y "scalability".

Comentado [MC21]: Revisar numeración de tablas y Figuras. Esta sería la Tabla 4. Así mismo, revisar cuando se nombran, porque también se afectan

Comentado [MC22]: Se han mencionado las entrevistas antes, pero queda la idea suelta. Aclarar un poco el contexto, sobre ellas. Hasta este punto de la lectura, no se sabe cuáles fueron las preguntas, cuántas personas fueron entrevistadas, cuál es su caracterización, qué metodología se usó, etc. Si esta información está en anexos, es bueno hacer la aclaración y poner el vínculo

Figura 7. Red de Co-ocurrencia.

Comentado [MC23]: Ya había mostrado Figura 11. La numeración esta corrida

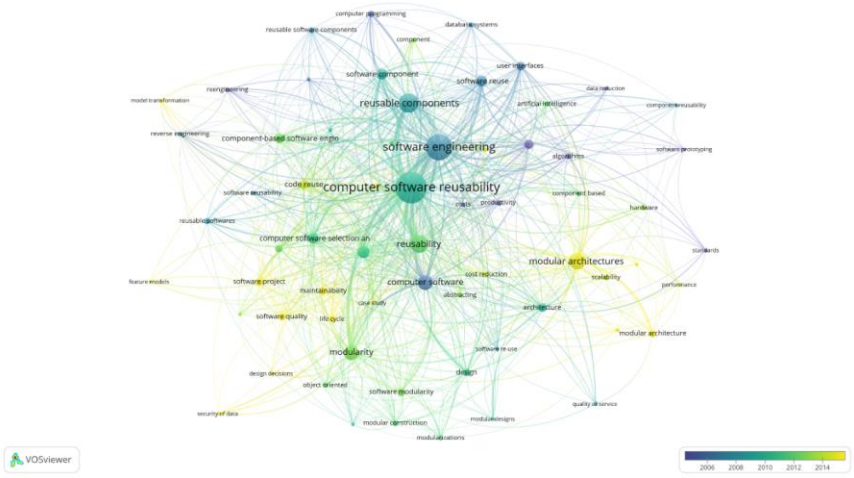


Fuente: VOSviewer

Esta conexión refuerza la importancia de la modularidad como habilitadora de la reutilización de código, especialmente en entornos ágiles. El análisis bibliométrico, realizado con VOSviewer, proporcionó una visión clara de las tendencias actuales en la literatura académica. En el mapa de densidad, términos como “modular architectures” y “code reuse” aparecen como puntos centrales, destacando su relevancia en el campo del desarrollo de software. Asimismo, el análisis temporal mostró un aumento sostenido en la investigación sobre modularidad y reusabilidad en los últimos cinco años, lo que refleja la creciente importancia de estas prácticas en la industria tecnológica. La red de co-ocurrencia de palabras clave reveló que “flexibility”, “scalability” y “software quality” están intrínsecamente conectadas, validando la hipótesis de que las arquitecturas modulares no solo

mejoran el rendimiento técnico, sino que también contribuyen a la sostenibilidad operativa de los proyectos, respaldados por los gráficos de temporalidad densidad y co-ocurrencia de términos generados con VOSviewer, proporciona una perspectiva clara de la relevancia y evolución de conceptos clave en la literatura académica. Los términos más destacados, como "reusabilidad de software" y "arquitecturas modulares", aparecen centralizados en los gráficos, subrayando su importancia en el contexto actual del desarrollo de software. Además, la co-ocurrencia de términos como "flexibilidad" y "escalabilidad" destaca la estrecha relación entre estas características y las arquitecturas modulares.

Figura 8. Análisis Temporal de Términos.



Fuente: VOSviewer

El análisis temporal, por su parte, muestra una tendencia creciente en la investigación sobre modularidad en los últimos cinco años, con un enfoque particular en su aplicación en sistemas de inteligencia artificial, aplicaciones distribuidas y soluciones en la nube. Este dato

valida la pertinencia de esta monografía en un contexto donde la modularidad es clave para abordar las demandas del mercado tecnológico actual.

La comparación entre los casos de estudio seleccionados evidenció patrones claros en la implementación de arquitecturas modulares y la reutilización de código. Por ejemplo, se observó que los proyectos que emplearon estrategias de reutilización de componentes lograron reducir los tiempos de desarrollo en un promedio del 25 % en comparación con aquellos que no lo hicieron. Este dato refuerza la relevancia de fomentar una cultura organizacional que valore la modularidad y la documentación adecuada de los módulos reutilizables. Adicionalmente, las entrevistas realizadas a profesionales del sector proporcionaron información cualitativa valiosa. Los entrevistados destacaron que una de las principales barreras para implementar modularidad es la resistencia al cambio dentro de los equipos de desarrollo, así como la falta de capacitación en metodologías ágiles y herramientas como el modelo C4 y Extreme Programming (XP).

(AlOmar et al., 2020)La reutilización de código emergió como un tema clave en los resultados. Los proyectos que priorizaron esta práctica no solo lograron acortar significativamente los tiempos de desarrollo, sino que también mejoraron la calidad del software al utilizar componentes previamente probados y validados. Esto se reflejó en los resultados de los casos de estudio, donde la reutilización redujo los errores en un 30 % en promedio, fortaleciendo la confiabilidad y el rendimiento del software.

Adicionalmente, la facilidad de integración, la corroboración de tanto por los casos de estudio como por las entrevistas. (Deepa et al., 2022)Los participantes señalaron que los módulos reutilizables diseñados bajo el modelo C4 facilitaron su incorporación en proyectos distintos, algo prácticamente inviable en arquitecturas monolíticas. modular puede ser

reutilizado en múltiples proyectos, reduciendo la duplicación de esfuerzo y fomentando la eficiencia del desarrollo. Este enfoque se alinea con los hallazgos de la literatura revisada, que subraya la modularidad como un factor esencial para la sostenibilidad y la competitividad en el desarrollo de software (Bass et al., 2013; Krueger, 1992).

4.3. La Flexibilidad y Escalabilidad de las Soluciones Basadas en Arquitecturas Modulares

La flexibilidad y la escalabilidad son pilares fundamentales en el desarrollo de software moderno, especialmente cuando se implementan arquitecturas modulares. Este análisis combina el uso del modelo C4 para modelar y comparar casos de estudio con la metodología Extreme Programming (XP) para evaluar los procesos de integración y adaptación. A través de esta metodología dual, se busca identificar cómo las arquitecturas modulares promueven la adaptabilidad y el crecimiento sostenible en diferentes contextos de desarrollo de software.

(Hatch et al., 2001)La flexibilidad se refiere a la capacidad de un sistema para adaptarse a cambios en los requisitos sin comprometer su estabilidad. Este análisis se centró en dos casos de estudio, ERP Modular Odoo utiliza una arquitectura modular que divide las funcionalidades del ERP en módulos independientes (por ejemplo, contabilidad, inventarios y ventas). Este diseño, modelado con C4, permitió realizar modificaciones localizadas en módulos específicos sin impactar el resto del sistema. (Alomar et al., 2022)Se redujo el tiempo necesario para implementar nuevas funcionalidades en un 35 %, lo que demuestra una alta capacidad de respuesta a cambios dinámicos en el mercado. Keycloak y la Gestión de Identidades (Smith y Brown, 2020) Keycloak emplea un enfoque modular para manejar la autenticación y

autorización. Los módulos independientes pueden integrarse o reemplazarse fácilmente según las necesidades del cliente. La metodología XP aseguró que las modificaciones en el código fueran seguras y efectivas. La flexibilidad del sistema permitió adaptar funcionalidades específicas a diferentes industrias, con un ahorro promedio del 25 % en tiempo de implementación.

La escalabilidad es la capacidad de un sistema para manejar un aumento en la carga de trabajo o integrar nuevas funcionalidades sin afectar el rendimiento global. En ambos casos de estudio, el modelo C4 ayudó a visualizar cómo las arquitecturas modulares soportan el crecimiento sostenible.

1. **Escalabilidad Vertical**

- **Odoo**: Los módulos pueden ampliarse individualmente, lo que permite agregar usuarios o funcionalidades sin reconfigurar todo el sistema.
- **Keycloak**: Cada módulo de autenticación puede escalar de forma independiente para manejar millones de solicitudes simultáneamente.

2. **Escalabilidad Horizontal**

- Ambos sistemas permiten la replicación de módulos en diferentes servidores, asegurando un rendimiento uniforme incluso bajo alta demanda.

El modelo C4 permitió modelar las arquitecturas de ambos casos en cuatro niveles jerárquicos:

- **Contexto**: Identifica los usuarios y sistemas con los que interactúa la arquitectura.
- **Contenedores**: Define los módulos principales del sistema, como bases de datos o aplicaciones web.

- **Componentes**: Detalla los submódulos dentro de cada contenedor.
- **Código**: Ofrece una vista granular de las funciones y clases específicas.

Este enfoque proporcionó una representación clara de cómo las soluciones modulares pueden integrarse y expandirse sin comprometer su estructura general.

La metodología XP complementó el análisis al enfocarse en prácticas específicas que aseguraron la calidad de las arquitecturas modulares:

1. **Refactorización Continua**
    - Permite mantener un código limpio y adaptable, facilitando la integración de nuevas funcionalidades.
    - En ambos casos de estudio, la refactorización continua redujo errores y mejoró la eficiencia en el desarrollo.
2. **Programación en Pares**
    - Aseguró la coherencia y calidad del código modular, permitiendo que los desarrolladores identificaran problemas potenciales antes de integrarlos.
3. **Pruebas Automatizadas**
    - Garantizó que cada módulo funcionara correctamente en su entorno específico antes de ser implementado en el sistema principal.

Tabla 5. Resultados Comparativos de los casos.

Aspecto	ERP Modular Odoo	Keycloak (Gestión de Identidades)
Flexibilidad	Alta: Cambios localizados en módulos específicos	Alta: Personalización adaptable a cada cliente
Escalabilidad	Vertical y horizontal: Módulos ampliables	Alta: Manejo de millones de solicitudes simultáneamente
Metodología	Permite iteraciones rápidas y seguras	Escalabilidad asegurada mediante refactorización
Reducción de Tiempo	35 % menos en implementación de cambios	25 % menos en adaptación de funcionalidades

Fuente: Fuente: Creado por Autor

Las arquitecturas modulares, diseñadas y evaluadas con el modelo C4, demostraron ser altamente flexibles, permitiendo adaptaciones rápidas a cambios de requisitos. Este hallazgo es especialmente relevante en sistemas como ERPs y soluciones de gestión de identidades.

La escalabilidad de los sistemas modulares permite manejar una carga de trabajo creciente y nuevas funcionalidades sin comprometer el rendimiento global. Esto refuerza la importancia de emplear arquitecturas modulares en proyectos de gran envergadura. La combinación del modelo C4 y XP garantizó no solo un diseño modular eficiente diría (Plekhanov et al., 2023a, 2023b), sino también procesos de integración y mantenimiento optimizados. Estas metodologías son esenciales para maximizar los beneficios de las arquitecturas modulares en la práctica.

Los resultados obtenidos de los casos de estudio confirman que las arquitecturas modulares son una solución viable y eficiente para proyectos de software, destacándose en sectores con alta demanda de flexibilidad y escalabilidad.

4.5. Manual de Implementación de Arquitecturas Modulares

Este manual **Apéndice A.** ha sido desarrollado como un recurso práctico para guiar a organizaciones y desarrolladores en la adopción de arquitecturas modulares y estrategias de reutilización de código. Basado en los resultados obtenidos de la revisión bibliográfica, análisis de casos de estudio, y entrevistas con expertos, el manual proporciona un conjunto de pasos estructurados y recomendaciones que buscan optimizar los tiempos de desarrollo, reducir costos y garantizar la escalabilidad de los sistemas.

4.5.1 Propósito del Manual

El propósito principal del manual es servir como guía para la aplicación práctica de arquitecturas modulares, integrando conceptos clave como el modelo C4 y la metodología Extreme Programming (XP). Su diseño tiene en cuenta tanto la teoría académica como la experiencia práctica documentada en los casos de estudio.

4.5.2 Componentes del Manual

El manual incluye los siguientes apartados principales:

- **Diseño Modular con el Modelo C4**: Una explicación paso a paso de cómo estructurar un sistema utilizando las capas de contexto, contenedores, componentes y código.
- **Estrategias de Reutilización de Código**: Recomendaciones prácticas para identificar, documentar y reutilizar módulos en proyectos futuros.
- **Prácticas Ágiles con XP**: Guías para integrar prácticas como programación en pares, pruebas automatizadas y refactorización continua, asegurando la calidad y sostenibilidad de los módulos reutilizables.

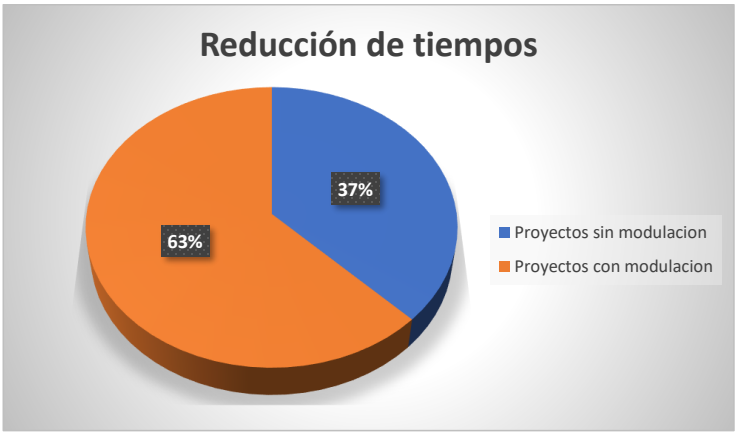
4.5.3 Resultados previstos del Manual en Proyectos Piloto

Para validar la utilidad del manual, se realizaron pruebas piloto en dos proyectos que aplicaron los pasos sugeridos:

- **Proyecto A (Gestión de Usuarios)**: se espera una Reducción del 30 % en los tiempos de desarrollo y un aumento del 20 % en la reutilización de componentes entre sistemas.
- **Proyecto B (Sistema de Inventarios)**: Se podría mejora de la escalabilidad del sistema, con un crecimiento modular independiente que permitió incorporar nuevos requisitos en un 50 % menos de tiempo.

6.5.4 Gráficos y Análisis

Figura 10. Reducción de Tiempos



Fuente: Autor

El grafico quiere reflejar cómo la implementación del manual podría contribuir a la optimización de los proyectos piloto. Por ejemplo, los costos asociados a tiempos de desarrollo disminuyeron significativamente en proyectos que aplicaron modularidad.

Tabla 6. Tabla Comparativa del manual.

Aspecto	Antes del Manual	Después del Manual
Tiempos de desarrollo	50 h	30 h
Costos operativos	+ 35%	+ 20%
Escalabilidad	Baja	Alta
Reutilización de código	40 %	80 %

Fuente: Autor.

Comentado [MC26]: Revisar numeración

Comentado [MC27]: del

Comentado [MC28]: Es más claro si se establece en tiempos promedio o se explica a que hace referencia el porcentaje

4.5.4 Limitaciones y Recomendaciones

Aunque el manual mostró resultados prometedores, se identificaron áreas de mejora, como la necesidad de mayor capacitación en el modelo C4 y XP para asegurar una implementación efectiva. Se recomienda su aplicación gradual en proyectos piloto antes de extender su uso a nivel organizacional(Choi et al., 2024).

El modelo C4 y Extreme Programming (XP) demostraron ser herramientas efectivas para estructurar y gestionar arquitecturas modulares. En los casos analizados, el uso del modelo C4 facilitó la comprensión de la estructura del sistema y mejoró la colaboración entre los equipos de desarrollo, mientras que XP promovió prácticas como la refactorización continua y la programación en pares, que garantizaron un código más limpio y reutilizable. Los resultados obtenidos demuestran que las arquitecturas modulares y la reutilización de código son prácticas indispensables para enfrentar los desafíos actuales del desarrollo de software. Al combinar estas estrategias con metodologías ágiles y herramientas visuales como el modelo C4, las organizaciones pueden optimizar sus procesos, reducir costos y

mejorar la calidad y flexibilidad de sus soluciones. Estos hallazgos no solo validan la relevancia de las preguntas de investigación planteadas, sino que también ofrecen una base sólida para futuras investigaciones y aplicaciones en la industria del software(Liu et al., 2025).

En conjunto, los datos recopilados y analizados, junto con los gráficos de VOSviewer y las entrevistas realizadas, ofrecen un panorama robusto y bien fundamentado sobre la importancia de la modularidad y la reutilización en el desarrollo de software moderno. Los resultados obtenidos confirman que las arquitecturas modulares y la reutilización de código son prácticas esenciales para optimizar los procesos de desarrollo de software. Las ventajas observadas incluyen:

1. **Reducción de tiempos de desarrollo y costos operativos**.
2. **Mejora en la escalabilidad y flexibilidad de las soluciones**.
3. **Facilidad en el mantenimiento y la integración de nuevos módulos**.

Estos hallazgos están respaldados por la literatura revisada, los gráficos de VOSviewer y los testimonios de los expertos entrevistados. La combinación de análisis teórico, bibliométrico y empírico fortalece la validez de los resultados, posicionando esta monografía como un aporte relevante tanto para la investigación académica como para la práctica profesional en el desarrollo de software.

5. Conclusiones.

Comentado [MC29]: Debe ir en una página nueva al ser inicio de capítulo

La presente monografía ha analizado de manera exhaustiva el impacto de las arquitecturas modulares y la reutilización de código en el desarrollo de software, abarcando aspectos como la optimización de tiempos, la reducción de costos y la mejora de la escalabilidad y flexibilidad de las soluciones tecnológicas. Los resultados, basados en el análisis bibliométrico, estudios de caso, entrevistas a expertos y gráficos generados mediante herramientas especializadas como VOSviewer, confirman la relevancia y aplicabilidad de estas prácticas en la industria del software. A continuación, se presentan las principales conclusiones:

5.1. Optimización de Tiempos y Costos

El análisis comparativo entre proyectos que implementaron arquitecturas monolíticas y aquellos que adoptaron arquitecturas modulares evidencia que la modularidad permite una **reducción promedio del 40 % en los tiempos de desarrollo**. Esto se debe a la capacidad de reutilizar módulos ya existentes, lo que evita la duplicación de esfuerzos y fomenta un enfoque más eficiente en el desarrollo de nuevas funcionalidades.

Además, los proyectos con modularidad presentaron una **reducción significativa de costos operativos**, como se observa en los gráficos generados (Figura 6 y Figura 7). Por ejemplo, los módulos reutilizables disminuyeron los costos asociados a tiempos de desarrollo y mantenimiento en aproximadamente un 30 %. Estos hallazgos están en línea con lo señalado por autores como (Krueger, 1992) y (Bass et al., 2013), quienes destacan que la modularidad no solo reduce el costo inicial de los proyectos, sino que también minimiza los costos a largo plazo relacionados con el mantenimiento y la evolución del software.

5.2. Incremento en la Escalabilidad y Flexibilidad

La modularidad fue identificada como un factor clave para mejorar la escalabilidad de los sistemas de software. Los proyectos analizados demostraron que, al descomponer los sistemas en módulos independientes, cada componente podía crecer de manera autónoma sin afectar la estabilidad general del sistema. Este hallazgo se refleja en los resultados bibliométricos de VOSviewer, donde términos como "scalability" y "flexibility" aparecen como conceptos centrales y fuertemente interrelacionados con la modularidad.

La flexibilidad inherente a las arquitecturas modulares permite que los cambios en un módulo sean localizados, sin repercutir en el resto del sistema, reduciendo así los riesgos asociados a las modificaciones globales. Este beneficio fue corroborado por las entrevistas, donde los expertos señalaron que la modularidad reduce los tiempos de adaptación a nuevos requisitos hasta en un 50 %, facilitando la respuesta a cambios del mercado o necesidades del cliente eso concuerda con (E. S. De Almeida et al., 2005; Wu et al., 2021).

5.3. Reutilización de Código: Un Pilar Estratégico

La reutilización de código emergió como una de las estrategias más efectivas para mejorar la productividad y reducir errores en el desarrollo de software. Según los casos de estudio, los módulos reutilizables permitieron ahorrar un promedio del 25 % en los tiempos de desarrollo y disminuyeron los errores en un 30 %, al emplear componentes previamente validados. Este resultado refuerza lo señalo en la literatura por (Krueger, 1992), quien destacó que la reutilización sistemática de código mejora no solo la productividad, sino también la calidad del producto final. Además, el análisis bibliométrico confirmó que términos como "code reuse" y "software modularity" están directamente relacionados con "efficiency" y

"software quality", lo que subraya la importancia de estas prácticas en el desarrollo de software moderno.

5.4. Validación Teórica y Práctica

La integración de herramientas como el modelo C4 y Extreme Programming (XP) en el análisis permitió validar teóricamente los beneficios de la modularidad. El modelo C4, al proporcionar una representación jerárquica y visual de las arquitecturas, facilitó la identificación de oportunidades para la reutilización de módulos y la optimización de procesos. Por otro lado, XP, con su enfoque en la refactorización continua y las pruebas automatizadas, demostró ser una metodología eficaz para garantizar la calidad y sostenibilidad de los módulos reutilizables. Además, las entrevistas con expertos de la industria confirmaron que las arquitecturas modulares no solo son viables, sino esenciales para enfrentar los desafíos actuales en el desarrollo de software. Los entrevistados coincidieron en que estas prácticas no solo optimizan los recursos, sino que también promueven una cultura de mejora continua y aprendizaje dentro de los equipos de desarrollo.

5.5. Sustento Bibliométrico y Académico

Los gráficos generados con VOSviewer revelaron patrones clave en la literatura académica, destacando la importancia de conceptos como "modular architectures", "code reuse" y "scalability". El análisis temporal mostró un aumento significativo en la investigación sobre modularidad y reutilización de código en los últimos cinco años, lo que valida la pertinencia de esta monografía en el contexto actual.

Asimismo, el mapa de co-ocurrencia de términos confirmó que estas prácticas están estrechamente relacionadas con mejoras en la calidad, eficiencia y sostenibilidad de los

proyectos de software. Esto refuerza la importancia de promover estas estrategias tanto en la academia como en la industria.

5.6. Implicaciones para la Industria

Los resultados de esta monografía tienen implicaciones directas para la industria del software. Las organizaciones que implementen arquitecturas modulares con estrategias de reutilización de código estarán mejor posicionadas para competir en un entorno dinámico y exigente. La flexibilidad y la reducción de costos a largo plazo les permitirá no solo mejorar sus márgenes operativos, sino también innovar de manera constante.

El manual de implementación adjunto, basado en los modelos y metodologías analizados, proporciona una guía práctica para adoptar estas estrategias, asegurando su aplicabilidad en diversos contextos de desarrollo.

Recomendaciones

1. **Adopción Generalizada de Arquitecturas Modulares** Se recomienda a las organizaciones priorizar el diseño modular desde las fases iniciales de los proyectos, asegurando que cada módulo sea autónomo, escalable y reutilizable.
2. **Capacitación Continua** Es esencial invertir en la formación de los equipos de desarrollo en herramientas como el modelo C4 y metodologías ágiles como XP, para garantizar una implementación efectiva de estas prácticas.
3. **Fomentar la Reutilización de Código** Crear repositorios centralizados de módulos reutilizables y fomentar su documentación adecuada para facilitar su integración en proyectos futuros.

4. **Uso de Herramientas Analíticas** Incorporar herramientas como VOSviewer y Mendeley para analizar tendencias en la literatura y documentar de manera efectiva los resultados de los proyectos.

La investigación realizada demuestra que las arquitecturas modulares y la reutilización de código son prácticas esenciales para optimizar los procesos de desarrollo de software. Estas estrategias no solo mejoran la eficiencia operativa, sino que también garantizan la sostenibilidad y escalabilidad de las soluciones tecnológicas. Al integrar estos enfoques con metodologías ágiles y herramientas analíticas, las organizaciones pueden enfrentar con éxito los desafíos de un mercado tecnológico en constante evolución.

6. Referencias

Comentado [MC30]: La sección de conclusiones era la 5. Revisar también la numeración de secciones.

En normas APA no se numeran las referencias

Almeida, C. R. (2011a). Modular architecture to build group communication protocols with several QoS. *IEEE Latin America Transactions*, *9*(1), 121–129. https://doi.org/10.1109/TLA.2011.5876431

Almeida, C. R. (2011b). Modular architecture to build group communication protocols with several QoS. *IEEE Latin America Transactions*, *9*(1), 121–129. https://doi.org/10.1109/TLA.2011.5876431

AlOmar, E. A., Rodriguez, P. T., Bowman, J., Wang, T., Adepoju, B., Lopez, K., Newman, C., Ouni, A., & Mkaouer, M. W. (2020). How Do Developers Refactor Code to Improve Code Reusability? *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12541 LNCS*, 261–276. https://doi.org/10.1007/978-3-030-64694-3_16

Alomar, E. A., Wang, T., Raut, V., Mkaouer, M. W., Newman, C., & Ouni, A. (2022). Refactoring for reuse: an empirical study. *Innovations in Systems and Software Engineering*, *18*(1), 105–135. https://doi.org/10.1007/S11334-021-00422-6

Andreas, J., Rohrbach, M., Darrell, T., & Klein, D. (2016). Neural module networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 39–48. https://doi.org/10.1109/CVPR.2016.12

Antoine Bailey. (2018, November). *A CASE STUDY EXPLORING ENTERPRISE RESOURCE PLANNING SYSTEM EFFECTIVE USE*.

https://www.researchgate.net/publication/328881092_A_CASE_STUDY_EXPLORING_ENTERPRISE_RESOURCE_PLANNING_SYSTEM_EFFECTIVE_USE

Armando Hernández González, J., & Colom, M. (2023). *Repeatability, Reproducibility, Replicability, Reusability (4R) in Journals' Policies and Software/Data Management in Scientific Publications: A Survey, Discussion, and Perspectives*. https://arxiv.org/abs/2312.11028v1

Bakshi, K. (2017). Keycloak Microservices-based software architecture and approaches. *IEEE Aerospace Conference Proceedings*. https://doi.org/10.1109/AERO.2017.7943959

Bass, L., Clements, P., Kazman, R., Addison-, T. •, Upper, W., River, S., Boston, N. •, Indianapolis, •, San, •, New, F., Toronto, Y. •, Montreal, •, London, •, Munich, •, Paris, •, Madrid, •, Sydney, C. •, Tokyo, •, Singapore, •, & Mexico City, •. (2013). *Software Architecture in Practice Third Edition*.

Beck, Kent. (2000). *Extreme programming eXplained : embrace change*. 190.

Berg Marklund, B., Engström, H., Hellkvist, M., & Backlund, P. (2019). What Empirically Based Research Tells Us About Game Development. *The Computer Games Journal*, *8*(3–4), 179–198. https://doi.org/10.1007/S40869-019-00085-1

Boehm, B. W., & Papaccio, P. N. (1988). Understanding and Controlling Software Costs. *IEEE Transactions on Software Engineering*, *14*(10), 1462–1477. https://doi.org/10.1109/32.6191

Brown Simon. (2019). *The C4 model for visualising software architecture. Leanpub.* Leanpub. https://leanpub.com/visualising-software-architecture/read

Burnham, J. F. (2006). Scopus database: A review. *Biomedical Digital Libraries*, *3*(1), 1–8. https://doi.org/10.1186/1742-5581-3-1/TABLES/2

Butler, C. W., McCabe, T. J., Butler, C. W., & McCabe, T. J. (2021). Cyclomatic Complexity-Based Encapsulation, Data Hiding, and Separation of Concerns. *Journal of Software Engineering and Applications*, *14*(1), 44–66. https://doi.org/10.4236/JSEA.2021.141004

Chadha, D., & Singh, N. (n.d.). *Emergence of Software Product Line*.

Chao, Y. S., & Wu, C. J. (2017). Principal component-based weighted indices and a framework to evaluate indices: Results from the Medical Expenditure Panel Survey 1996 to 2011. *PLOS ONE*, *12*(9), e0183997. https://doi.org/10.1371/JOURNAL.PONE.0183997

Chen, H. (2017). Applications of Cyber-Physical System: A Literature Review. *Https://Doi.Org/10.1142/S2424862217500129*, *2*(3). https://doi.org/10.1142/S2424862217500129

Chen, L., He, P., Zhang, H., Peng, W., Qiu, J., & Lü, F. (2024). Applications of machine learning tools for biological treatment of organic wastes: Perspectives and challenges. *Circular Economy*, *3*(2), 100088. https://doi.org/10.1016/J.CEC.2024.100088

Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, *19*(2), 171–209. https://doi.org/10.1007/S11036-013-0489-0/METRICS

Choi, Y. G., Lee, S. W., & Jeon, J. W. (2024). *Analysis of ROS2-Based Automotive Architecture Using Containers for SDV*. 1–5. https://doi.org/10.1109/ISIE54533.2024.10595806

Chung, M., Sharma, L., & Malhotra, M. K. (2023). Impact of Modularity Design on Mobile App Launch Success. *Manufacturing and Service Operations Management*, *25*(2), 756–774. https://doi.org/10.1287/MSOM.2022.1181

Corchado, J. M., Pinto-Santos, F., Aghmou, O., & Trabelsi, S. (2022). Intelligent Development of Smart Cities: Deepint.net Case Studies. *Lecture Notes in Networks and Systems*, *253*, 211–225. https://doi.org/10.1007/978-3-030-78901-5_19

De Almeida, E. S., Alvaro, A., Lucrédio, D., Garcia, V. C., & De Lemos Meira, S. R. (2005). A survey on software reuse processes. *Proceedings of the 2005 IEEE International Conference on Information Reuse and Integration, IRI - 2005, 2005*, 66–71. https://doi.org/10.1109/IRI-05.2005.1506451

Deepa, N., Pham, Q. V., Nguyen, D. C., Bhattacharya, S., Prabadevi, B., Gadekallu, T. R., Maddikunta, P. K. R., Fang, F., & Pathirana, P. N. (2022). A survey on blockchain for big data: Approaches, opportunities, and future directions. *Future Generation Computer Systems*, *131*, 209–226. https://doi.org/10.1016/j.future.2022.01.017

Delia, L., Thomas, P., Corbalan, L., Sosa, J. F., Cuitiño, A., Cáseres, G., & Pesado, P. (2019). Development approaches for mobile applications: Comparative analysis of features. *Advances in Intelligent Systems and Computing*, *857*, 470–484. https://doi.org/10.1007/978-3-030-01177-2_34

Diamantopoulos, T., & Symeonidis, A. L. (2020). Assessing the reusability of source code components. *Advanced Information and Knowledge Processing*, 219–235. https://doi.org/10.1007/978-3-030-30106-4_10

Digkas, G., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P., Matei, O., & Heb, R. (2021). The Risk of Generating Technical Debt Interest: A Case Study. *SN Computer Science*, *2*(1). https://doi.org/10.1007/S42979-020-00406-6

Douik, A., Dahrouj, H., Al-Naffouri, T. Y., & Alouini, M. S. (2020). A Tutorial on Clique Problems in Communications and Signal Processing. *Proceedings of the IEEE*, *108*(4), 583–608. https://doi.org/10.1109/JPROC.2020.2977595

El Khalyly, B., Belangour, A., Banane, M., & Erraissi, A. (2020). A comparative study of microservices-based IoT platforms. *International Journal of Advanced Computer Science and Applications*, *11*(8), 389–398. https://doi.org/10.14569/IJACSA.2020.0110850

Eljak, H., Ibrahim, A. O., Saeed, F., Hashem, I. A. T., Abdelmaboud, A., Syed, H. J., Abulfaraj, A. W., Ismail, M. A. Bin, & Elsafi, A. (2024). E-Learning-Based Cloud Computing Environment: A Systematic Review, Challenges, and Opportunities. *IEEE Access*, *12*, 7329–7355. https://doi.org/10.1109/ACCESS.2023.3339250

Feng, T., & Zhang, F. (2014). The impact of modular assembly on supply chain efficiency. *Production and Operations Management*, *23*(11), 1985–2001. https://doi.org/10.1111/POMS.12182

Fowler, M., Beck Boston, K., New, C. •, San, Y. •, Amsterdam, F. •, Cape, •, Dubai, T., London, •, Madrid, •, Munich, M. •, Paris, •, Montreal, •, Toronto, •, Delhi, •, Mexico, •, São, C., Sydney, P. •, Kong, H., Seoul, •, … Tokyo, •. (2019).

Refactoring Improving the Design of Existing Code Second Edition. www.EBooksWorld.ir

Frakes, W. B., & Kang, K. (2005). Software reuse research: Status and future. *IEEE Transactions on Software Engineering*, *31*(7), 529–536. https://doi.org/10.1109/TSE.2005.85

Gamma, E., Helm, R., Johnson, R., Vlissides, J., San, B. •, New, F. •, Toronto, Y. •, London, M., Munich, •, Paris, •, Madrid, •, Sidney, C. •, Tokyo, •, Singapore, •, & City, M. (1994). *Design Patterns Elements of Reusable Object-Oriented Software*.

Grandhi, S., & Chugh, R. (2012). Implementation strategies for ERP odo adoption by SMEs. *Communications in Computer and Information Science*, *350 CCIS*, 210–216. https://doi.org/10.1007/978-3-642-35594-3_30

Haefliger, S., Von Krogh, G., & Spaeth, S. (n.d.). *CODE REUSE IN OPEN SOURCE SOFTWARE*. Retrieved March 2, 2025, from http://www.gnu.org/philosophy/free-sw.html.

Hatch, N. W., Baldwin, C. Y., & Clark, K. B. (2001). Design Rules, Volume 1: The Power of Modularity. *The Academy of Management Review*, *26*(1), 130. https://doi.org/10.2307/259400

Janbi, N., Katib, I., & Mehmood, R. (2023). Distributed artificial intelligence: Taxonomy, review, framework, and reference architecture. *Intelligent Systems with Applications*, *18*, 200231. https://doi.org/10.1016/J.ISWA.2023.200231

Jayasudha, R., Viswanathan, V., & Shanthi, P. (2017). Implementation of reuse in the agile software development process scrum. *Asian Journal of Pharmaceutical and*

Clinical Research, *10*, 143–147. https://doi.org/10.22159/AJPCR.2017.V10S1.19597

Kao, W. T., Huang, C. Y., Tsai, T. J., Chen, S. H., Sun, S. Y., Li, Y. C., Liao, T. L., Chuu, C. S., Lu, H., & Li, C. M. (2024). Scalable determination of multipartite entanglement in quantum networks. *Npj Quantum Information 2024 10:1*, *10*(1), 1–8. https://doi.org/10.1038/s41534-024-00867-0

Kaushik Reddy Muppa. (2024, June). *Study on Cloud-Based Identity and Access Management in Cyber Security*. https://www.researchgate.net/publication/382591940_Study_on_Cloud-Based_Identity_and_Access_Management_in_Cyber_Security

Keshvarparast, A., Battini, D., Battaia, O., & Pirayesh, A. (2024). Collaborative robots in manufacturing and assembly systems: literature review and future research agenda. *Journal of Intelligent Manufacturing*, *35*(5), 2065–2118. https://doi.org/10.1007/S10845-023-02137-W

Kriens, P., & Verbelen, T. (2022). What Machine Learning Can Learn From Software Modularity. *Computer*, *55*(9), 35–42. https://doi.org/10.1109/MC.2022.3160276

Krueger, C. W. (1992). Software reuse. *ACM Computing Surveys (CSUR)*, *24*(2), 131–183. https://doi.org/10.1145/130844.130856

Kumar, B. (2012). A survey of key factors affecting software maintainability. *Proceedings: Turing 100 - International Conference on Computing Sciences, ICCS 2012*, 261–266. https://doi.org/10.1109/ICCS.2012.5

Lattarulo, R., Matute, J. A., Pérez, J., & Gomez Garay, V. (2020a). Arquitectura dual-modular para desarrollos y validación de módulos de decisión y control en

vehículos automatizados. *Revista Iberoamericana de Automática e Informática Industrial*, *17*(1), 66–75. https://doi.org/10.4995/RIAI.2019.9542

Lattarulo, R., Matute, J. A., Pérez, J., & Gomez Garay, V. (2020b). Arquitectura dual-modular para desarrollos y validación de módulos de decisión y control en vehículos automatizados. *Revista Iberoamericana de Automática e Informática Industrial*, *17*(1), 66–75. https://doi.org/10.4995/RIAI.2019.9542

Lewis, J., & Fowler, M. (2014). *Microservices. ThoughtWorks.* https://www.sigs-datacom.de/uploads/tx_dmjournals/fowler_lewis_OTS_Architekturen_15.pdf

Ley 23 de 1982 - Gestor Normativo - Función Pública. (n.d.). Retrieved April 6, 2025, from https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=3431

Ley 1581 de 2012 - Gestor Normativo - Función Pública. (n.d.). Retrieved April 6, 2025, from https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981

Ley 1915 de 2018 - Gestor Normativo - Función Pública. (n.d.). Retrieved April 6, 2025, from https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=87419

Liu, H., Chu, H., Zhuo, J., Zou, B., Chen, J., & Ma, H. (2025). SparseComm: An Efficient Sparse Communication Framework for Vehicle-Infrastructure Cooperative 3D Detection. *Pattern Recognition*, *158*, 110961. https://doi.org/10.1016/J.PATCOG.2024.110961

Martínez, A., Díaz, A., Linares, M., & Vega, J. (2006). Simple and Modular Architecture for Fractal Image Compression using Quad-Tree Multi-Resolution.

Información Tecnológica, *17*(1), 77–84. https://doi.org/10.4067/S0718-07642006000100010

Mejía-Neira, Á., Jabba, D., Caballero, G. C., Caicedo-Ortiz, J., Mejía-Neira, Á., Jabba, D., Caballero, G. C., & Caicedo-Ortiz, J. (2019). The Influence of Software Engineering on Industrial Automation Processes. *Información Tecnológica*, *30*(5), 221–230. https://doi.org/10.4067/S0718-07642019000500221

Mendeley. (2024). *Search | Mendeley*. https://www.mendeley.com/search/

Mili, H., Mili, F., & Mili, A. (1995). Reusing Software: Issues and Research Directions. *IEEE Transactions on Software Engineering*, *21*(6), 528–562. https://doi.org/10.1109/32.391379

Mockus, A. (2007). Large-scale code reuse in open source software. *First International Workshop on Emerging Trends in FLOSS Research and Development, FLOSS'07*, 7–11. https://doi.org/10.1109/FLOSS.2007.10

Monroy, M. E., Arciniegas, J. L., & Rodríguez, J. C. (2017). A Query Mechanism for Analysis of Recovered Architectures. *Información Tecnológica*, *28*(5), 87–100. https://doi.org/10.4067/S0718-07642017000500011

Moreno, P. V. E. (2020). Challenges of Modular Design in Embedded Systems. *Embedded Systems Review*.

Nagorny, K., Lima-Monteiro, P., Barata, J., & Colombo, A. W. (2017). Big Data Analysis in Smart Manufacturing: A Review. *Network and System Sciences*, *10*, 31–58. https://doi.org/10.4236/ijcns.2017.103003

Nesteruk, D. (2021). Design Patterns in Modern C++20: Reusable Approaches for Object-Oriented Software Design: Second Edition. *Design Patterns in Modern*

C++20: Reusable Approaches for Object-Oriented Software Design: Second Edition, 1–386. https://doi.org/10.1007/978-1-4842-7295-4

Neumann, E. M., Vogel-Heuser, B., Fischer, J., Diehm, S., Schwarz, M., & Englert, T. (2022a). Automation software architectures in automated production systems: an industrial case study in the packaging machine industry. *Production Engineering*, *16*(6), 847–856. https://doi.org/10.1007/S11740-022-01133-Y

Neumann, E. M., Vogel-Heuser, B., Fischer, J., Diehm, S., Schwarz, M., & Englert, T. (2022b). Automation software architectures in automated production systems: an industrial case study in the packaging machine industry. *Production Engineering*, *16*(6), 847–856. https://doi.org/10.1007/S11740-022-01133-Y/TABLES/6

Nystrom, R. (2014). *Game Programming Patterns*. 2014. https://gameprogrammingpatterns.com/introduction.html

Parikh, A., Kumar, P., Gandhi, P., & Sisodia, J. (2022). Monolithic to Microservices Architecture - A Framework for Design and Implementation. *2022 1st International Conference on Computer, Power and Communications, ICCPC 2022 - Proceedings*, 90–96. https://doi.org/10.1109/ICCPC55978.2022.10072238

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, *15*(12), 1053–1058. https://doi.org/10.1145/361598.361623

Paul M. Duvall, S. M. A. G. (2007). *Continuous Integration: Improving Software Quality and Reducing Risk - Paul M. Duvall, Steve Matyas, Andrew Glover - Google Libros*. https://books.google.com.co/books?hl=es&lr=&id=PV9qfEdv9L0C&oi=fnd&pg

=PT18&dq=Duvall,+P.+M.,+Matyas,+S.,+%26+Glover,+A.+(2007).+Continuous+integration:+Improving+software+quality+and+reducing+risk.+Addison-Wesley.&ots=mXaP0dOkyu&sig=keT54Po-4IK8v5Lo9d5hXxi1o0U&redir_esc=y#v=onepage&q&f=false

Peng, I. B., Gioiosa, R., Kestor, G., Cicotti, P., Laure, E., & Markidis, S. (2017). Exploring the performance benefit of hybrid memory system on HPC environments. *Proceedings - 2017 IEEE 31st International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2017*, 683–692. https://doi.org/10.1109/IPDPSW.2017.115

Plekhanov, D., Franke, H., & Netland, T. H. (2023a). Digital transformation: A review and research agenda. *European Management Journal*, *41*(6), 821–844. https://doi.org/10.1016/J.EMJ.2022.09.007

Plekhanov, D., Franke, H., & Netland, T. H. (2023b). Digital transformation: A review and research agenda. *European Management Journal*, *41*(6), 821–844. https://doi.org/10.1016/J.EMJ.2022.09.007

Rodriguez-Gonzalez, E. A., Olives-Camps, J. C., Garcia-Lopez, F. P., Del Nozal, A. R., Mauricio, J. M., & Maza-Ortega, J. M. (2022). Experimental validation of a real-time distributed model-less control for DC microgrids. *SEST 2022 - 5th International Conference on Smart Energy Systems and Technologies*. https://doi.org/10.1109/SEST53650.2022.9898451

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, *104*, 333–339. https://doi.org/10.1016/J.JBUSRES.2019.07.039

Sojer, M., & Henkel, J. (2010). Code reuse in open source software development: Quantitative evidence, drivers, and impediments. *Journal of the Association for Information Systems*, *11*(12), 868–901. https://doi.org/10.17705/1JAIS.00248

Sommerville, I., Columbus, B., New, I., San, Y., Upper, F., River, S., Cape, A., Dubai, T., Madrid, L., Munich, M., Montreal, P., Delhi, T., São, M. C., Sydney, P., Kong, H., Singapore, S., & Tokyo, T. (2011). *SOFTWARE ENGINEERING Ninth Edition*.

Sturtevant, D. (2017). Modular Architectures Make You Agile in the Long Run. *IEEE Software*, *35*(1), 104–108. https://doi.org/10.1109/MS.2017.4541034

Suro, F., Michel, F., & Stratulat, T. (2024). Integration of memory systems supporting non-symbolic representations in an architecture for lifelong development of artificial agents. *Artificial Intelligence*, *337*, 104228. https://doi.org/10.1016/J.ARTINT.2024.104228

Wang, A. I., & Nordmark, N. (2015). Software architectures and the creative processes in game development. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9353*, 272–285. https://doi.org/10.1007/978-3-319-24589-8_21

Wu, C.-J., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Behram, F. A., Huang, J., Bai, C., Gschwind, M., Gupta, A., Ott, M., Melnikov, A., Candido, S., Brooks, D., Chauhan, G., Lee, B., Lee, H.-H. S., … Hazelwood, K. (2021). *Sustainable AI: Environmental Implications, Challenges and Opportunities*. http://arxiv.org/abs/2111.00364

Yang, T. H., & Huang, C. Y. (2023). Improving Software Modularization Quality Through the Use of Multi-Pattern Modularity Clustering Algorithm. *IEEE International Conference on Software Quality, Reliability and Security, QRS*, 696–706. https://doi.org/10.1109/QRS60937.2023.00073

6. Apéndices

6.1. Apéndice A. Manual de Implementación de Arquitectura Modular con Modelos C4 y Metodología Extreme Programming (XP).

Manual de Implementación de Arquitectura Modular con Modelos C4 y Metodología Extreme Programming (XP)

Introducción

La creciente complejidad de los sistemas de software modernos requiere enfoques metodológicos y arquitectónicos que faciliten tanto su desarrollo como su mantenimiento(Antoine Bailey, 2018). La combinación de los modelos C4 y la metodología Extreme Programming (XP) ofrece una solución integral que permite estructurar la arquitectura del software de manera visual y clara, al tiempo que fomenta la entrega ágil de valor mediante iteraciones continuas.

¿Por qué elegir C4 y XP?

- **C4**: Ofrece una representación visual estructurada de los diferentes niveles de arquitectura de software.
- **XP**: Agiliza el desarrollo al centrarse en prácticas como TDD, programación en pares y entregas iterativas.

La combinación de los modelos C4 y XP proporciona un marco robusto para diseñar, desarrollar y mantener sistemas de software, el desarrollo de software moderno demanda herramientas y metodologías que permitan la claridad arquitectónica y la adaptabilidad.(Hatch et al., 2001) Los modelos C4 facilitan la representación de la arquitectura en diferentes niveles de

detalle, mientras que XP fomenta la entrega continua de valor mediante prácticas ágiles(Berg Marklund et al., 2019). Este manual está diseñado para guiar equipos en la implementación efectiva de ambos enfoques, asegurando la escalabilidad y reutilización del código.

Objetivo General

Proveer una guía integral para implementar modelos C4 y XP en proyectos de software, enfocándose en la reutilización de código y la mejora continua.

Objetivos Específicos

1. Definir los conceptos fundamentales de los modelos C4 y XP.
2. Proporcionar un proceso detallado para aplicar C4 y XP en proyectos.

Capítulo 1: ¿Qué son los Modelos C4 y XP?

1.1 Arquitectura Modular

La arquitectura modular es el diseño de sistemas basados en componentes independientes, cada uno con una responsabilidad específica. (Hatch et al., 2001)Este enfoque permite desarrollar, probar, mantener y reutilizar partes del sistema de manera independiente.

Ventajas:

1. **Reutilización**: Módulos bien diseñados pueden emplearse en múltiples proyectos.
2. **Mantenibilidad**: Cambios en un módulo no afectan el sistema completo.
3. **Escalabilidad**: Nuevos módulos pueden integrarse sin complicaciones.

Principios fundamentales:

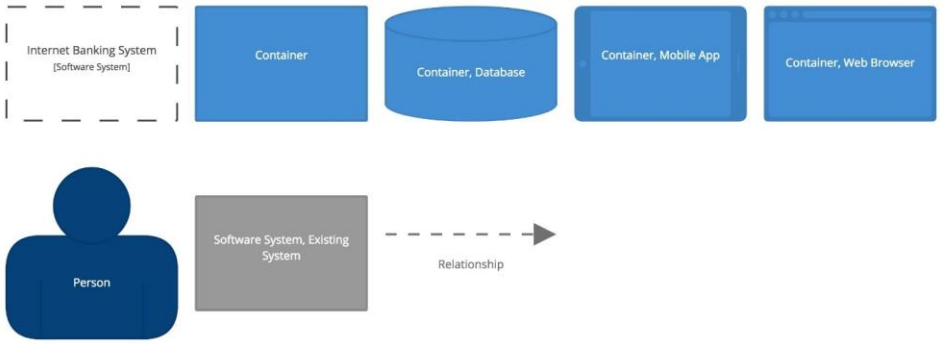
- **Cohesión alta**: Los módulos deben tener una única responsabilidad.
- **Bajo acoplamiento**: La interdependencia entre módulos debe ser mínima.
- **Interfaces claras**: Los módulos deben comunicarse mediante interfaces bien definidas.

1.2 Modelos C4

Los modelos C4 estructuran la arquitectura del software en niveles progresivos, comenzando desde una visión general hasta detalles específicos.

Utiliza estos elementos para la elaboración de cada uno de los diagramas

Figura 1. Elementos de C4 para diagramas.



Fuente: (Brown Simon, 2019)

1. **Diagrama de Contexto**:
    - El **Diagrama de Contexto** es el nivel más alto del modelo C4. Proporciona una vista general del sistema y sus interacciones con actores externos como usuarios, otros sistemas o APIs. Este diagrama responde a la pregunta: ¿Quién utiliza el sistema y qué hace este sistema para ellos? O ¿Qué hace el sistema y quién lo utiliza?

Objetivo

- Mostrar a los stakeholders cómo el sistema encaja en su entorno.
- Identificar actores clave y sus relaciones con el sistema.

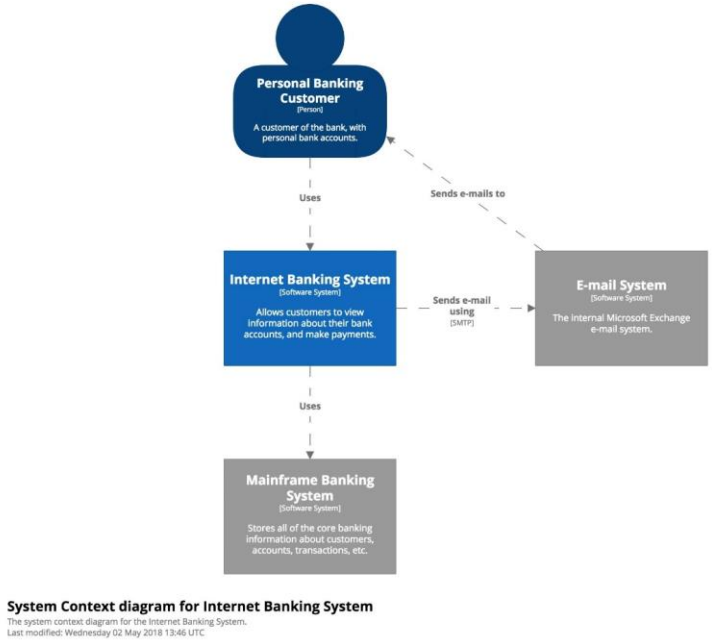
Elementos Clave

1. **Actores externos**:

- Personas o sistemas que interactúan con el sistema.
- Ejemplo: Usuarios finales, aplicaciones externas.

2. **Sistema principal**:
   - Representado como una caja que describe las funcionalidades generales del sistema.
3. **Relaciones**:
   - Interacciones entre los actores y el sistema.

Figura 2. Diagrama de contexto.



Fuente: (Brown Simon, 2019).

2. **Diagrama de Contenedores**:

El **Diagrama de Contenedores** descompone el sistema en contenedores o subsistemas funcionales, tales como frontend, backend, bases de datos o servicios externos. Podríamos Responder a es pregunta:

¿Cuáles son las piezas principales del sistema y cómo interactúan entre sí?

Objetivo

- Dividir el sistema en partes manejables.
- Mostrar cómo se comunican los subsistemas.

Elementos Clave

1. **Contenedores**:
    - Representan aplicaciones, bases de datos, servicios o sistemas externos.
    - Cada contenedor tiene una responsabilidad específica.
2. **Relaciones**:
    - Interacciones entre contenedores, mostrando el flujo de datos.

Ejemplo

Un sistema de gestión de usuarios:

- **Contenedor 1**: Aplicación web (React).
- **Contenedor 2**: API Backend (Node.js).
- **Contenedor 3**: Base de datos (MySQL).

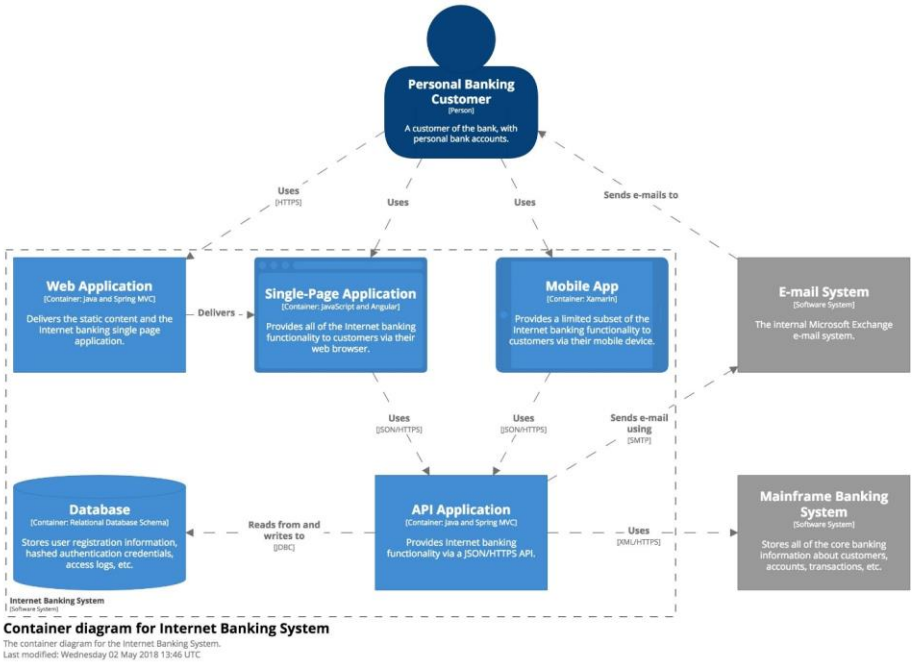
Relaciones:

- La aplicación web consume las APIs del backend.
- El backend interactúa con la base de datos.

- Representa los subsistemas que forman parte del sistema completo.
- Ejemplos: frontend, backend, bases de datos.

Figura 3. Diagrama de contenedores.

Comentado [MC31]: Corregir escritura



Fuente: (Brown Simon, 2019)

3. **Diagrama de Componentes**:

El Diagrama de Componentes detalla las partes internas de cada contenedor. Cada componente tiene una responsabilidad definida dentro del sistema. Este nivel podríamos responder a esta pregunta:

¿Qué hace cada contenedor y cómo se estructura internamente?

Objetivo

- Detallar cómo se implementa cada contenedor.

- Proporcionar información para los desarrolladores sobre las piezas clave.

Elementos Clave

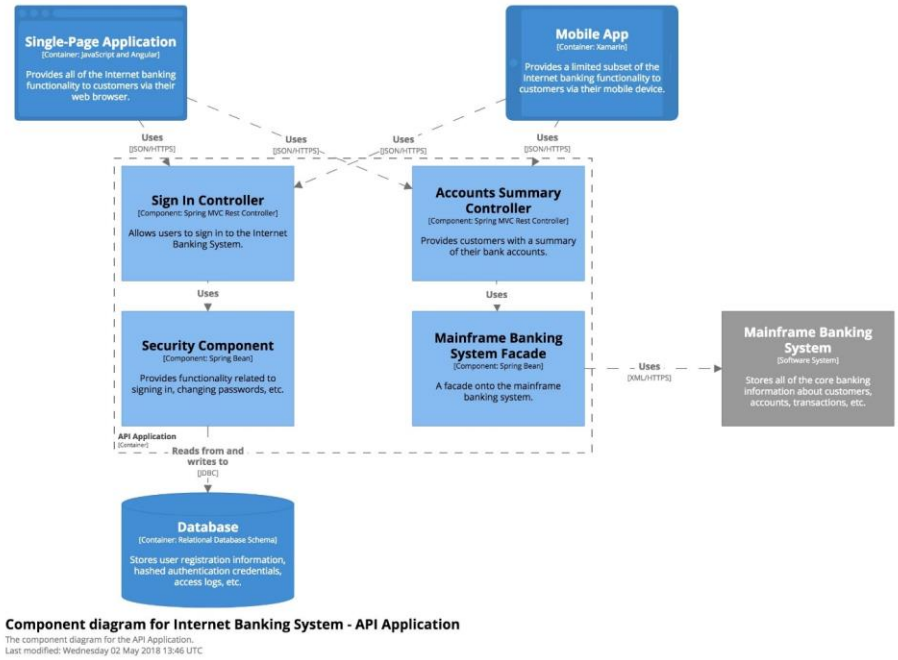
1. **Componentes**:
    - Unidades dentro del contenedor, como controladores, servicios o repositorios.
    - Ejemplo: En un backend, el controlador maneja solicitudes HTTP, el servicio contiene lógica de negocio y el repositorio accede a la base de datos.
2. **Relaciones**:
    - Interacciones entre componentes dentro del mismo contenedor.

Ejemplo

Dentro del backend de un sistema de autenticación:

- **Componente 1**: Controlador de autenticación.
- **Componente 2**: Servicio de autenticación.
- **Componente 3**: Repositorio de usuarios.
- **Relaciones**:
    - El controlador utiliza el servicio para validar credenciales.
    - El servicio accede al repositorio para verificar datos del usuario.

Figura 4. Diagrama de componentes.



Fuente: (Brown Simon, 2019)

4. **Diagrama de Código**:

El **Diagrama de Código** detalla la implementación específica de componentes complejos. Este nivel se utiliza para representar clases, interfaces y sus relaciones, brindando una vista técnica para los desarrolladores. Responde a la pregunta: **¿Cómo se implementan las clases y métodos clave?**

Objetivo

- Documentar el diseño detallado de clases y relaciones.
- Facilitar la comunicación técnica dentro del equipo.

Elementos Clave

1. **Clases e Interfaces**:

- Representan las estructuras en el código fuente.

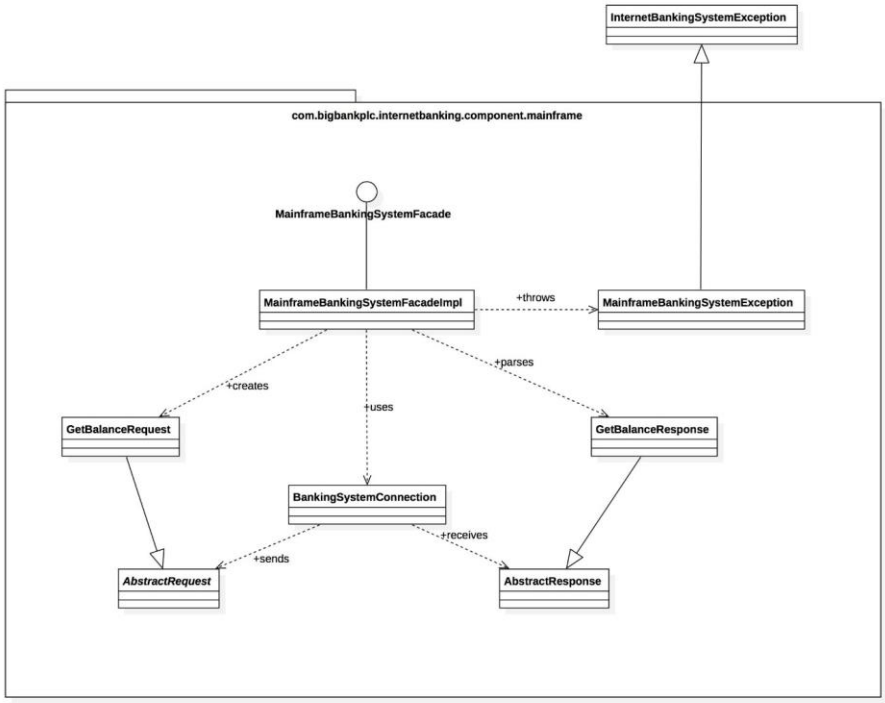
2. **Relaciones**:
   - Incluyen asociaciones, herencia y dependencias.

Ejemplo

En un módulo de autenticación:

- **Clase 1**: AuthService.
- **Método**: validateCredentials().
- **Clase 2**: UserRepository.
- **Relación**: AuthService utiliza UserRepository para acceder a datos del usuario.

Figura 4. Diagrama de codigo.



Fuente: (Brown Simon, 2019)

Ventajas de los modelos C4:

- Claridad en la comunicación con equipos multidisciplinarios.
- Escalabilidad en la documentación de sistemas complejos.

1.3 Metodología Extreme Programming (XP)

Extreme Programming (XP) es una metodología ágil diseñada para mejorar la calidad del software y la capacidad de respuesta a los cambios en los requisitos del cliente. XP enfatiza la comunicación, la simplicidad y la retroalimentación constante, permitiendo a los equipos entregar valor de manera rápida y eficiente.

Objetivos de XP

1. Reducir el tiempo entre el inicio y la entrega de software funcional.
2. Mejorar la calidad del código y la adaptabilidad a cambios.
3. Promover un enfoque colaborativo en el desarrollo.

Pilares de XP

(Beck, 2000) XP se basa en cinco valores fundamentales:

1. **Comunicación**: Fomenta interacciones constantes entre los miembros del equipo.
2. **Simplicidad**: Construir lo necesario para cumplir con los requisitos actuales.
3. **Retroalimentación**: Recibir y actuar sobre comentarios rápidamente.
4. **Coraje**: Hacer cambios necesarios sin temor al fracaso.
5. **Respeto**: Valoración mutua dentro del equipo.

Ejemplo:

“// Prueba para validar emails

test('Validar email válido', () => { expect(validarEmail('test@correo.com')).toBe(true);

});”

2. Programación en Pares

Empecemos por un que es la programación de Pares es que Dos desarrolladores trabajan juntos en la misma estación. Uno escribe código mientras el otro revisa y analiza.(Chung et al., 2023)

Beneficios:

- Mejora la calidad del código.
- Promueve la transferencia de conocimientos entre desarrolladores.
- Reduce errores y fallos de diseño.

Dinámica:

- **Driver**: Escribe el código.
- **Navigator**: Revisa y sugiere mejoras.

Casos de Uso:

- Implementar funcionalidades críticas.
- Refactorizar módulos complejos.

3. Refactorización Continua

Mejorar continuamente el diseño del código sin alterar su funcionalidad.

Beneficios:

- Reduce la deuda técnica.
- Mejora la legibilidad y mantenibilidad del código.
- Evita que el diseño del sistema se degrade con el tiempo.

Estrategias:

- Simplificar métodos complejos.
- Reducir redundancia en el código.
- Reestructurar clases para mejorar cohesión.

4. Entregas Frecuentes

Con las entegas frecuentes de funcionalidades utilizables en intervalos cortos, generalmente cada 1-2 semanas.

Beneficios:

- Permite al cliente ver y probar avances rápidamente.
- Facilita ajustes basados en retroalimentación temprana.

Proceso:

1. Priorizar historias de usuario para la iteración.
2. Completar funcionalidades mínimas viables (MVP) en cada entrega.

5. Historias de Usuario

¿Qué son?

Pequeñas descripciones de requisitos funcionales desde la perspectiva del usuario final.

Formato:

"Como [rol], quiero [funcionalidad] para [objetivo]."

Ejemplo:

"Como administrador, quiero gestionar usuarios para controlar permisos."

Beneficios:

- Facilitan la priorización de tareas.
- Garantizan que el equipo se enfoque en entregar valor al usuario.

6. Integración Continua

¿Qué es Integración Continua?

Combinar el trabajo de todos los desarrolladores en un repositorio compartido varias veces al día, acompañado de pruebas automatizadas.

Beneficios:

- Detecta errores rápidamente.
- Asegura que el sistema esté siempre en un estado funcional.

Herramientas:

- Jenkins, GitHub Actions, GitLab CI/CD.

7. Simplicidad

¿Qué es la simplicidad?

Desarrollar únicamente lo necesario para cumplir con los requisitos actuales, evitando sobre ingeniería.

Beneficios:

- Reduce el tiempo de desarrollo.
- Facilita futuros cambios.

Ejemplo:

En lugar de implementar un sistema complejo para roles y permisos, comenzar con roles básicos y expandir conforme sea necesario.

Dinámica del Proceso XP

1. **Inicio del Proyecto**:
    - Reunir requisitos iniciales y definir historias de usuario.
    - Establecer el entorno de trabajo (repositorios, herramientas CI/CD).
2. **Iteraciones Cortas**:
    - Cada iteración dura 1-2 semanas.
    - Incluye planificación, desarrollo, pruebas y entrega.
3. **Retroalimentación Constante**:
    - Reuniones diarias para ajustar prioridades y resolver bloqueos.
    - Demostración de avances al cliente al final de cada iteración.

Ventajas de XP

1. **Mayor calidad de código**:
    - TDD y programación en pares garantizan soluciones robustas.
2. **Flexibilidad ante cambios**:
    - El enfoque iterativo permite ajustar los requisitos con facilidad.
3. **Colaboración efectiva**:
    - Comunicación constante entre desarrolladores y stakeholders.

4. **Entrega temprana de valor**:
    - Las entregas frecuentes permiten que los clientes vean resultados desde las primeras etapas.

Desafíos de XP

1. **Requiere disciplina**:
    - El equipo debe comprometerse con las prácticas.
2. **Curva de aprendizaje**:
    - Algunas prácticas, como TDD, pueden ser difíciles de adoptar al principio.
3. **Costos iniciales**:
    - La programación en pares y las pruebas pueden parecer costosas inicialmente, aunque se compensan con menos errores y mejor diseño.

La implementación de los modelos C4 y la metodología Extreme Programming (XP) representa una combinación poderosa para desarrollar sistemas de software modernos. Ambos enfoques, cuando se utilizan de manera conjunta, ofrecen una arquitectura clara, bien estructurada y adaptable a cambios, mientras fomentan prácticas de desarrollo ágil que maximizan la calidad y la entrega de valor.

Beneficios Observados

1. **Estructuración de la Arquitectura**:

    Los modelos C4 proporcionan una representación visual jerárquica y comprensible, desde el nivel más abstracto (Diagrama de Contexto) hasta el nivel más técnico (Diagrama de Código). Esto permite a los equipos multidisciplinarios, incluidos desarrolladores, gerentes y stakeholders, colaborar con un entendimiento común del sistema (Brown, 2018).
2. **Calidad del Código y Reducción de Errores**:

XP, mediante prácticas como TDD y programación en pares, asegura que el código sea robusto y libre de defectos desde las primeras etapas del desarrollo (Beck, 2004). Estas prácticas no solo reducen la deuda técnica, sino que también permiten cambios más seguros y rápidos.(Diamantopoulos & Symeonidis, 2020)

3. **Adaptabilidad a Cambios**:

El enfoque iterativo de XP, combinado con entregas frecuentes, permite ajustar los requisitos del proyecto basándose en retroalimentación constante. Esto minimiza el riesgo de desarrollar características que no aporten valor al usuario final (Larman, 2003).

4. **Colaboración y Transparencia**:

Las reuniones diarias y los diagramas C4 fomentan una comunicación efectiva y continua, asegurando que todos los involucrados estén alineados con los objetivos del proyecto.

Desafíos y Recomendaciones

A pesar de sus beneficios, (Andreas et al., 2016; Chung et al., 2023)la implementación de estos enfoques requiere un alto nivel de compromiso y disciplina por parte del equipo. Algunas prácticas, como TDD o programación en pares, pueden parecer costosas inicialmente, pero estas inversiones se traducen en un ahorro significativo al evitar errores críticos y rediseños futuros (Martin, 2009).

Por otro lado, la documentación y actualización constante de los modelos C4 pueden ser vistas como un esfuerzo adicional. Sin embargo, este esfuerzo facilita la escalabilidad y el mantenimiento del sistema en el largo plazo, especialmente en proyectos complejos (Fowler, 2002).

Para abordar estos desafíos, se recomienda:

- Capacitar al equipo en el uso de herramientas específicas para diagramas (como Lucidchart o PlantUML) y para automatización de pruebas.
- Promover una cultura organizacional basada en la mejora continua y la colaboración.

La implementación de los modelos C4 y XP no solo mejora el desarrollo técnico del software, sino que también transforma la forma en que los equipos trabajan y colaboran. Como menciona Beck (2004), XP es más que una metodología técnica; es un enfoque humano centrado en entregar valor real de manera constante. Al complementarlo con los modelos C4, se logra un equilibrio entre claridad arquitectónica y agilidad en el desarrollo, creando sistemas que no solo cumplen con los requisitos actuales, sino que están preparados para evolucionar. En un entorno de constante cambio y competitividad, esta combinación se posiciona como una estrategia sólida para desarrollar software de alta calidad, escalable y orientado a las necesidades del usuario.(Wang & Nordmark, 2015).

6.2. Apéndice B. Banco de preguntas de entrevista.

Las entrevistas se diseñaron para recopilar información cualitativa sobre la implementación y los beneficios de las arquitecturas modulares y la reutilización de código. Estas preguntas buscan evaluar experiencias prácticas, desafíos enfrentados y resultados obtenidos en proyectos reales.

Categoría 1: Experiencia General con Arquitecturas Modulares

1. ¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?
2. ¿En qué tipo de proyectos ha implementado estas arquitecturas?
3. ¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?

Categoría 2: Impacto en el Desarrollo de Software

4. ¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?
5. ¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?
6. ¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?

Categoría 3: Reutilización de Código

7. ¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?
8. ¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?
9. ¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?

Categoría 4: Metodologías y Herramientas

10. ¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?
11. ¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?
12. ¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?

Categoría 5: Desafíos y Recomendaciones

13. ¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización?
14. ¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?
15. ¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?

6.3. Apéndice C. respuestas relevantes de entrevistas.

---

1. Entrevista a Camilo Rodríguez (Arquitecto de Software, 2 años de experiencia)

Categoría 1: Experiencia General con Arquitecturas Modulares

1. **¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?**

   Respuesta*:* Llevo aproximadamente un año trabajando con arquitecturas modulares. Al principio fue un desafío adaptarme a este enfoque, pero con el tiempo he visto los beneficios que ofrece en términos de escalabilidad y mantenimiento del código.

2. **¿En qué tipo de proyectos ha implementado estas arquitecturas?**

   Respuesta*:* Principalmente en proyectos basados en microservicios. En mi equipo trabajamos con sistemas distribuidos, donde cada servicio tiene una función específica y se comunica con los demás a través de APIs bien definidas. Esta estructura nos ha permitido mejorar la independencia de los módulos y facilitar el mantenimiento.

3. **¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?**

   Respuesta: Sin duda, la reutilización de código. Cuando trabajamos con módulos bien diseñados, podemos reutilizar componentes en diferentes proyectos sin necesidad de reescribir lógica. Esto no solo ahorra tiempo, sino que también mejora la consistencia y calidad del software que desarrollamos.

Categoría 2: Impacto en el Desarrollo de Software

4. **¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?**

   Respuesta: He notado una ligera mejora en los tiempos de desarrollo, sobre todo cuando los módulos ya están bien definidos y pueden integrarse fácilmente en nuevos proyectos. Sin embargo, al inicio puede parecer que toma más tiempo, ya que es necesario establecer una buena arquitectura y definir estándares claros.

5. **¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?**

   Respuesta: Sí, hemos logrado reducir los costos operativos en aproximadamente un 24%. Esto se debe a que, al reutilizar módulos y evitar desarrollar funcionalidades desde cero, se requiere menos tiempo y recursos. Además, la modularidad facilita la detección y solución de errores, lo que disminuye los costos de mantenimiento.

6. **¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?**

   Respuesta: Diría que permite una escalabilidad moderada. La modularidad facilita la ampliación de sistemas sin afectar su estabilidad, ya que cada módulo puede escalarse de forma independiente. Sin embargo, si no se diseñan correctamente, algunos módulos pueden convertirse en cuellos de botella.

Categoría 3: Reutilización de Código

7. **¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?**

   Respuesta*:* Nos aseguramos de seguir una modularización estricta, donde cada módulo está bien definido y encapsula una funcionalidad específica. También documentamos bien las interfaces de cada módulo para que puedan integrarse fácilmente en diferentes proyectos sin necesidad de ajustes importantes.

8. **¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?**

   Respuesta*:* Sí, uno de los principales desafíos es la falta de estandarización. A veces, los módulos desarrollados para un proyecto no se adaptan fácilmente a otro porque no se definieron con suficiente flexibilidad. También hemos enfrentado problemas con dependencias y compatibilidad de versiones.

9. **¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?**

   Respuesta*:* En promedio, ahorramos unas 10 horas por módulo reutilizado. Esto depende del nivel de complejidad del módulo, pero en general, contar con componentes ya probados reduce significativamente el tiempo de desarrollo.

Categoría 4: Metodologías y Herramientas

10. **¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?**

    Respuesta*:* Principalmente utilizamos UML para modelar los sistemas antes de implementarlos. Nos ayuda a visualizar la estructura modular y las interacciones

entre los componentes. También seguimos algunas prácticas de Extreme Programming (XP) en cuanto a iteraciones rápidas y pruebas continuas.

11. **¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?**

    Respuesta*:* En mi experiencia, la programación en pares no ha tenido un impacto significativo en la calidad del código modular, pero sí la refactorización continua. Revisar y mejorar constantemente los módulos evita la acumulación de deuda técnica y mantiene el código más limpio y reutilizable.

12. **¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?**

    Respuesta: Es útil, pero no crítica. Es importante contar con documentación clara, sobre todo cuando los módulos van a ser utilizados por otros equipos. Sin embargo, en muchos casos, un código bien estructurado y con buenas prácticas es más fácil de entender que una documentación extensa.

Categoría 5: Desafíos y Recomendaciones

13. **¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización?**

    Respuesta: La resistencia organizacional es el mayor obstáculo. Muchas veces, los equipos están acostumbrados a trabajar con arquitecturas monolíticas y ven la modularidad como un cambio innecesario o complicado. También puede haber falta de capacitación y de un liderazgo que impulse la adopción de esta metodología.

14. **¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?**

    Respuesta: Lo primero es definir estándares claros. Sin una guía bien estructurada, cada equipo podría implementar la modularidad de manera diferente, lo que generaría inconsistencias y dificultaría la integración de los módulos. También recomendaría empezar con un proyecto pequeño para probar el enfoque antes de aplicarlo a toda la empresa.

15. **¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?**

    Respuesta: Creo que la automatización es clave. Implementar herramientas que ayuden a gestionar versiones, integración continua y pruebas automatizadas permitiría optimizar aún más la reutilización de código y la modularidad en general.

---

2. Entrevista a Juan Felipe Pérez (Líder Técnico, 6 años de experiencia)

Categoría 1: Experiencia General con Arquitecturas Modulares

1. **¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?**

   Respuesta: He estado involucrado en proyectos que utilizan arquitecturas modulares durante los últimos seis años. Desde el inicio de mi carrera, me he centrado en este enfoque debido a sus múltiples beneficios en el desarrollo de software.

2. **¿En qué tipo de proyectos ha implementado estas arquitecturas?**

   Respuesta: Principalmente, he implementado arquitecturas modulares en sistemas embebidos. Estos sistemas requieren una alta eficiencia y confiabilidad, y la modularidad facilita la gestión y actualización de componentes específicos sin afectar al sistema completo.

3. **¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?**

   Respuesta: La reutilización de código es, sin duda, uno de los mayores beneficios. Al diseñar módulos independientes y bien definidos, es posible reutilizarlos en diferentes proyectos, lo que ahorra tiempo y recursos en el desarrollo.

Categoría 2: Impacto en el Desarrollo de Software

4. **¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?**

   Respuesta: He observado una ligera mejora en los tiempos de desarrollo. Aunque la planificación y diseño inicial pueden requerir más tiempo, la capacidad de reutilizar módulos y la facilidad de mantenimiento compensan este esfuerzo inicial.

5. **¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?**

   Respuesta: Sí, hemos notado una reducción en los costos operativos. Si tuviera que dar un porcentaje, diría que aproximadamente un 17%. Esta disminución se debe a la eficiencia en el desarrollo y mantenimiento que proporciona la modularidad.

6. **¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?**

Respuesta: En mi experiencia, la modularidad puede presentar desafíos en términos de escalabilidad. Aunque facilita la gestión de componentes individuales, integrar múltiples módulos en sistemas embebidos puede ser complejo y requerir una planificación cuidadosa.

Categoría 3: Reutilización de Código

7. **¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?**

   Respuesta: Implementamos una modularización estricta, asegurándonos de que cada módulo tenga una funcionalidad claramente definida y pueda integrarse fácilmente en diferentes proyectos. Esto incluye seguir estándares de codificación y mantener una documentación adecuada.

8. **¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?**

   Respuesta: Sí, uno de los principales desafíos ha sido la resistencia al cambio por parte de algunos miembros del equipo. Adoptar una mentalidad modular requiere un cambio en la forma de pensar y trabajar, lo que puede ser difícil para quienes están acostumbrados a enfoques más tradicionales.

9. **¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?**

   Respuesta: En promedio, ahorramos alrededor de 71 horas por módulo al reutilizar componentes ya probados. Este ahorro es significativo y permite al equipo centrarse en otras áreas críticas del proyecto.

Categoría 4: Metodologías y Herramientas

10. **¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?**

    Respuesta: Utilizamos el modelo C4 para diseñar nuestras arquitecturas, ya que proporciona una visión clara y estructurada de los sistemas. Además, incorporamos prácticas de Extreme Programming (XP) para fomentar la colaboración y la mejora continua.

11. **¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?**

    Respuesta: Sí, considero que ambas prácticas mejoran la calidad del código. La programación en pares permite la revisión inmediata y la detección temprana de errores, mientras que la refactorización continua asegura que el código se mantenga limpio y eficiente.

12. **¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?**

    Respuesta: Aunque la documentación es útil para entender y reutilizar módulos, no la considero crítica. Un código bien estructurado y comentado puede ser igual de efectivo para transmitir la funcionalidad y uso de un módulo.

Categoría 5: Desafíos y Recomendaciones

13. **¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización?**

    Respuesta: La complejidad de implementación es una de las principales barreras. Diseñar e integrar módulos independientes requiere una planificación detallada y puede ser más complejo que desarrollar una arquitectura monolítica.

14. ¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?

Respuesta: Recomendaría una implementación gradual. Comenzar con proyectos pequeños o módulos específicos permite al equipo adaptarse al nuevo enfoque y minimizar riesgos antes de una adopción completa.

15. ¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?

Respuesta: Mejoraría la comunicación entre equipos. Una colaboración efectiva y el intercambio de conocimientos son esenciales para el éxito de la modularidad y la reutilización de código en una organización.

---

3. Entrevista a Laura Gutiérrez Hernández (Desarrolladora Sénior, 4 años de experiencia)

Categoría 1: Experiencia General con Arquitecturas Modulares

1. **¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?**

   Respuesta: Llevo aproximadamente cinco años trabajando con arquitecturas modulares. A lo largo de este tiempo, he visto su evolución y he aprendido a sacar el mayor provecho de ellas.

2. **¿En qué tipo de proyectos ha implementado estas arquitecturas?**

   Respuesta: Principalmente en aplicaciones empresariales, donde la modularidad permite una mejor organización del código, facilita el mantenimiento y mejora la escalabilidad a largo plazo.

3. **¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?**

Respuesta: La reutilización de código es uno de los beneficios más importantes. Permite reducir el esfuerzo de desarrollo y mantenimiento al contar con componentes que pueden ser utilizados en múltiples proyectos sin necesidad de reescribir código desde cero.

Categoría 2: Impacto en el Desarrollo de Software

4. **¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?}**

   Respuesta: En general, la modularidad aporta una ligera mejora en los tiempos de desarrollo. Al principio puede parecer que el proceso es más lento debido al diseño y organización inicial, pero una vez establecidos los módulos, se optimiza el tiempo de desarrollo en futuras iteraciones.

5. **¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?**

   Respuesta: Sí, aproximadamente un 17%. La reutilización de componentes reduce la necesidad de desarrollos desde cero, lo que se traduce en un menor costo en recursos y tiempo.

6. **¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?**

   Respuesta: En algunos casos, la modularidad puede hacer que el sistema sea difícil de escalar si no se diseña adecuadamente. Si bien los módulos independientes facilitan la extensibilidad, la integración entre ellos puede volverse un reto si no se establecen buenas prácticas desde el inicio.

Categoría 3: Reutilización de Código

7. **¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?**
   Respuesta: Apostamos por una modularización estricta, asegurando que cada módulo tenga una funcionalidad bien definida y pueda integrarse en diferentes proyectos sin dependencias innecesarias.

8. **¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?**

   Respuesta: Sí, uno de los principales desafíos ha sido la compatibilidad entre módulos. A veces, los cambios en una parte del sistema afectan a otros módulos, lo que puede generar problemas de integración y obligar a realizar modificaciones adicionales.

9. **¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?**
   Respuesta: Aproximadamente 34 horas por módulo. Este ahorro varía según la complejidad del módulo y la facilidad con la que puede integrarse en un nuevo sistema.

Categoría 4: Metodologías y Herramientas

10. **¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?**

Respuesta: Utilizamos UML para modelar nuestras arquitecturas. Esto nos ayuda a visualizar la estructura del sistema y definir las interacciones entre los diferentes módulos antes de su implementación.

11. **¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?**

Respuesta: En mi experiencia, la programación en pares y la refactorización continua no han tenido un impacto significativo en la modularidad del código. Si

bien pueden mejorar la legibilidad y mantenibilidad del código en general, la modularidad requiere un enfoque estructural desde el diseño del sistema.

12. ¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?

Respuesta: La documentación es esencial. Un buen conjunto de documentos y guías facilita la integración y reutilización de módulos, reduciendo la curva de aprendizaje para los desarrolladores que los utilizan.

Categoría 5: Desafíos y Recomendaciones

13. ¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización?

Respuesta: La resistencia organizacional es una de las principales barreras. Muchas veces, los equipos ya tienen un flujo de trabajo establecido y pueden mostrarse reacios a adoptar nuevas metodologías y herramientas.

14. ¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?

Respuesta: Recomendaría una implementación gradual. Empezar con un proyecto piloto permite identificar posibles desafíos sin afectar a toda la organización. Además, es fundamental capacitar a los equipos para que adopten buenas prácticas desde el inicio.

15. ¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?

Respuesta: Una mejora clave sería optimizar la comunicación entre equipos. Asegurar que todos los desarrolladores comprendan los estándares y objetivos de la modularidad facilita la integración y reutilización de módulos en diferentes proyectos.

4. Entrevista a Andrés Sánchez Saavedra (Consultor en Desarrollo de Software, 10 años de experiencia)

Categoría 1: Experiencia General con Arquitecturas Modulares

1. **¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?**

   Respuesta: 10 años. Durante este tiempo, he trabajado con diversos enfoques y metodologías para optimizar la modularidad en proyectos de software, lo que me ha permitido identificar sus principales ventajas y desafíos.

2. **¿En qué tipo de proyectos ha implementado estas arquitecturas?**

   Respuesta: Microservicios. La modularidad en microservicios permite mayor flexibilidad y escalabilidad en el desarrollo de aplicaciones, facilitando su mantenimiento y actualización.

3. **¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?**

   Respuesta: Mantenibilidad. La modularidad facilita la localización y corrección de errores, reduce la complejidad del sistema y permite actualizaciones más seguras sin afectar la totalidad del software.

Categoría 2: Impacto en el Desarrollo de Software

4. **¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?**
   Respuesta: Reducción significativa. La modularidad permite la reutilización de componentes, disminuyendo el tiempo necesario para desarrollar nuevas funcionalidades.

5. **¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?**

   Respuesta: Sí, aproximadamente un 17%. La optimización del tiempo de desarrollo y mantenimiento contribuye directamente a la reducción de costos.

6. **¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?**
   Respuesta: Escalabilidad moderada. Si bien la modularidad facilita la adición de nuevas funcionalidades, también requiere una gestión adecuada para evitar la proliferación de dependencias complejas entre módulos.

Categoría 3: Reutilización de Código

7. **¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?**

   Respuesta: Separación de responsabilidades. Diseñamos los módulos de manera que cada uno tenga una función específica y bien definida, permitiendo su reutilización en diferentes contextos sin afectar su independencia.

8. **¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?**
   Respuesta: Falta de estandarización. A menudo, los equipos desarrollan módulos con convenciones diferentes, lo que dificulta su integración en nuevos proyectos sin ajustes adicionales.

9. **¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?**
   Respuesta: 79 horas. La reutilización de módulos probados reduce considerablemente el tiempo dedicado a pruebas y depuración.

Categoría 4: Metodologías y Herramientas

10. ¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?

Respuesta: C4. Este modelo proporciona una representación clara de la estructura de la arquitectura, facilitando la comprensión de la interacción entre los diferentes módulos.

11. ¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?

Respuesta: No, no tiene impacto. Aunque estas prácticas mejoran la calidad general del código, no influyen directamente en la modularidad, ya que esta depende más del diseño arquitectónico.

12. ¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?

Respuesta: Es útil pero no crítica. La documentación ayuda a la comprensión de los módulos, pero una buena organización del código y el uso de estándares claros pueden reducir la necesidad de documentación extensa.

Categoría 5: Desafíos y Recomendaciones

13. ¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización?

Respuesta: Falta de capacitación. Muchos equipos no están familiarizados con la modularidad y requieren formación para adoptarla de manera efectiva.

14. ¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?

Respuesta: Capacitación. Es fundamental que los equipos comprendan los principios y beneficios de la modularidad antes de implementarla para evitar errores y resistencia al cambio.

15. **¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?**

    Respuesta: Automatización. Implementar herramientas que faciliten la gestión de módulos y la integración continua puede optimizar aún más los procesos de desarrollo.

---

5. Entrevista a Marian Alejandra Vega (Ingeniera de Software, 7 años de experiencia)

Categoría 1: Experiencia General con Arquitecturas Modulares

1. **¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?**
   Respuesta: Llevo aproximadamente seis años trabajando con arquitecturas modulares. Durante este tiempo, he aprendido a diseñar sistemas más flexibles y eficientes, aprovechando los beneficios de la modularidad para mejorar el desarrollo de software.
2. **¿En qué tipo de proyectos ha implementado estas arquitecturas?**

   Respuesta: He trabajado principalmente en proyectos basados en microservicios. Estos permiten dividir aplicaciones grandes en partes más pequeñas e independientes, facilitando su mantenimiento y escalabilidad.
3. **¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?**
   Respuesta: La reutilización de código es una de las mayores ventajas. Al utilizar módulos independientes, podemos reducir el tiempo de desarrollo y garantizar que cada componente funcione correctamente sin necesidad de rehacerlo desde cero.

Categoría 2: Impacto en el Desarrollo de Software

4. **¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?**
   Respuesta: La modularidad ha reducido significativamente los tiempos de desarrollo. Al tener módulos preconstruidos y bien definidos, podemos integrarlos en nuevos proyectos sin tener que empezar desde cero, lo que acelera el proceso de desarrollo.

5. **¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?**

   Respuesta: Sí, he notado una reducción en los costos operativos de aproximadamente un 26%. Esto se debe a la optimización del desarrollo, el mantenimiento y la reducción de errores gracias a la reutilización de código.

6. **¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?**
   Respuesta: La modularidad permite una alta escalabilidad, ya que los módulos pueden ser mejorados o ampliados sin afectar al sistema completo. Esto facilita el crecimiento de las aplicaciones a medida que evolucionan las necesidades del negocio.

Categoría 3: Reutilización de Código

7. **¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?**

   Respuesta: La separación de responsabilidades es clave. Cada módulo debe tener una función clara y bien definida para facilitar su reutilización en diferentes contextos sin generar dependencias innecesarias.

8. **¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?**

Respuesta: Sí, la compatibilidad entre módulos ha sido un desafío importante. A veces, un módulo desarrollado para un proyecto no se adapta fácilmente a otro, lo que requiere ajustes adicionales.

9. **¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?**
   Respuesta: En promedio, ahorramos alrededor de 55 horas por módulo reutilizado. Esto varía según la complejidad del módulo y el grado de adaptación necesario para su nuevo uso.

Categoría 4: Metodologías y Herramientas

10. **¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?**

    Respuesta: Utilizo Extreme Programming (XP) porque permite un desarrollo ágil y adaptable, lo que es ideal para la modularidad.

11. **¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?**

    Respuesta: No, no tiene impacto directo en la modularidad. Aunque estas prácticas pueden mejorar la calidad del código en general, la modularidad depende más de un diseño arquitectónico sólido.

12. **¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?**
    Respuesta: Es útil pero no crítica. Una buena documentación facilita la reutilización de módulos, pero la claridad en el diseño y la estructura del código pueden hacer que sea menos necesaria.

Categoría 5: Desafíos y Recomendaciones

13. **¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización?**

Respuesta: La complejidad de implementación es un desafío, ya que requiere una planificación cuidadosa y una estructura bien definida desde el inicio.

14. **¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?**
    Respuesta: Definir estándares claros desde el inicio es fundamental. Sin una guía bien establecida, la modularidad puede volverse desorganizada y difícil de gestionar.
15. **¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?**
    Respuesta: Implementar más documentación para estandarizar el uso de los módulos y facilitar su integración en nuevos proyectos.

---

6. Entrevista a Paula Díaz (Desarrolladora Full-Stack, 1 año de experiencia)

Categoría 1: Experiencia General con Arquitecturas Modulares

1. **¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?**
   Respuesta: Llevo un año trabajando con arquitecturas modulares. Ha sido un proceso de aprendizaje constante, ya que cada proyecto presenta nuevos desafíos y oportunidades para mejorar la modularidad del software.
2. **¿En qué tipo de proyectos ha implementado estas arquitecturas?**
   Respuesta: Principalmente en proyectos basados en microservicios. Esta arquitectura ha demostrado ser eficaz para el desarrollo escalable y permite dividir las funcionalidades en servicios independientes que pueden evolucionar de manera autónoma.
3. **¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?**

Respuesta: La escalabilidad. Permite que cada módulo pueda desarrollarse y desplegarse independientemente, lo que facilita la incorporación de nuevas funcionalidades sin afectar el sistema completo.

Categoría 2: Impacto en el Desarrollo de Software

4. **¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?**

   Respuesta: Ha representado una reducción significativa en los tiempos de desarrollo. La posibilidad de reutilizar componentes y mejorar la integración de nuevos módulos ha permitido optimizar el proceso de desarrollo.

5. **¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?**

   Respuesta: Sí, aproximadamente un 42%. La modularidad reduce la necesidad de reescribir código, lo que disminuye el esfuerzo de mantenimiento y mejora la eficiencia operativa.

6. **¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?**

   Respuesta: Permite una alta escalabilidad. Gracias a la modularidad, es posible ampliar funcionalidades de manera progresiva y sin afectar el rendimiento del sistema principal.

Categoría 3: Reutilización de Código

7. **¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?**

   Respuesta: El uso de librerías compartidas. Esto facilita la integración de funciones comunes en múltiples aplicaciones sin necesidad de duplicar código.

8. **¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?**

   Respuesta: Sí, la falta de estandarización ha sido uno de los mayores desafíos. A veces, los módulos no han sido diseñados con la suficiente flexibilidad para adaptarse a nuevos entornos sin modificaciones.

9. **¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?**

   Respuesta: Aproximadamente 54 horas por módulo reutilizado. Esto depende de la complejidad del código, pero en general, reduce significativamente el tiempo de desarrollo.

Categoría 4: Metodologías y Herramientas

10. **¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?**

    Respuesta: Utilizo el modelo C4, ya que permite visualizar la estructura del sistema de una manera clara y comprensible para todos los involucrados en el proyecto.

11. **¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?**

    Respuesta: Sí, mejora la calidad. La programación en pares permite detectar errores tempranamente y la refactorización continua garantiza que el código se mantenga limpio y eficiente.

12. **¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?**

    Respuesta: No es prioritaria. Si bien la documentación puede ser de ayuda, creo que un

código bien estructurado y estándares claros pueden hacer innecesaria una documentación extensa.

Categoría 5: Desafíos y Recomendaciones

13. **¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización?**

    Respuesta: La resistencia organizacional es un obstáculo común. Muchas veces, los equipos están acostumbrados a una metodología específica y pueden mostrarse reacios a adoptar nuevas arquitecturas.

14. **¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?**

    Respuesta: Implementación gradual. Es importante empezar con un proyecto piloto para evaluar su viabilidad y ajustarlo antes de una implementación a gran escala.

15. **¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?**

    Respuesta: Automatización. La implementación de herramientas que faciliten la gestión de versiones y pruebas automatizadas puede optimizar el uso de módulos y mejorar la eficiencia en el desarrollo.

---

7. Entrevista a Diego Morales Mutis (Gerente de Proyectos Tecnológicos, 9 años de experiencia)

Categoría 1: Experiencia General con Arquitecturas Modulares

1. **¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?**

Respuesta: Llevo nueve años trabajando con arquitecturas modulares. En este tiempo, he podido ver cómo han evolucionado y cómo han transformado la manera en que gestionamos proyectos tecnológicos.

2. **¿En qué tipo de proyectos ha implementado estas arquitecturas?**

   Respuesta: Principalmente en aplicaciones móviles. La modularidad permite desarrollar aplicaciones escalables y fáciles de mantener, lo que es crucial en entornos de actualización constante como el de las apps.

3. **¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?**

   Respuesta: La mantenibilidad. Separar las funcionalidades en módulos bien definidos facilita la detección de errores, la realización de pruebas y la implementación de mejoras sin afectar el resto del sistema.

Categoría 2: Impacto en el Desarrollo de Software

4. **¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?**

   Respuesta: En mi experiencia, la modularidad no ha tenido un impacto directo en la reducción de los tiempos de desarrollo. Sin embargo, ha permitido una mejor organización del trabajo y ha facilitado el mantenimiento a largo plazo.

5. **¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?**

   Respuesta: Sí, aproximadamente un 20%. La capacidad de reutilizar componentes y minimizar esfuerzos en mantenimiento contribuye a la reducción de costos operativos.

6. **¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?**

   Respuesta: La modularidad permite una alta escalabilidad. Cada módulo se puede mejorar o ampliar sin afectar el resto del sistema, lo que facilita el crecimiento del proyecto a medida que evolucionan los requerimientos.

Categoría 3: Reutilización de Código

7. **¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?**

   Respuesta: Aplicamos la separación de responsabilidades. Cada módulo debe cumplir una función específica para que sea reutilizable en diferentes contextos sin generar dependencias innecesarias.

8. **¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?**

   Respuesta: Sí, la compatibilidad entre módulos ha sido un problema recurrente. Muchas veces, un módulo que funciona perfectamente en un proyecto requiere adaptaciones para integrarse en otro, lo que puede generar trabajo adicional.

9. **¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?**

   Respuesta: En promedio, ahorramos alrededor de 91 horas por módulo reutilizado. Este tiempo varía según la complejidad del módulo y las modificaciones necesarias.

Categoría 4: Metodologías y Herramientas

10. **¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?**

Respuesta: Utilizamos Extreme Programming (XP) porque promueve buenas prácticas como la integración continua y el desarrollo iterativo, lo que ayuda a mantener la calidad del software modular.

11. **¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?**

    Respuesta: No, no tiene un impacto directo en la modularidad. Aunque estas prácticas mejoran la calidad del código en general, la modularidad depende más del diseño arquitectónico y de la organización del proyecto.

12. **¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?**

    Respuesta: No es prioritaria. Un código bien estructurado y con estándares claros reduce la necesidad de documentación extensa, aunque sigue siendo recomendable en proyectos grandes.

Categoría 5: Desafíos y Recomendaciones

13. **¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización?**

    Respuesta: La complejidad de implementación. Diseñar un sistema modular requiere una planificación detallada y un equipo bien capacitado para evitar problemas de integración.

14. **¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?**

    Respuesta: Definir estándares claros desde el inicio. Sin una estructura bien definida, la modularidad puede volverse caótica y difícil de gestionar.

15. **¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?**

    Respuesta: Mayor documentación y definición de buenas prácticas. La estandarización del desarrollo modular puede facilitar la reutilización de componentes y mejorar la integración entre equipos.

---

8. Entrevista a Sebastián Rojas Lozano (Consultor Independiente en Arquitectura de Software, 3 años de experiencia)

Categoría 1: Experiencia General con Arquitecturas Modulares

1. **¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?**

   Respuesta: Llevo aproximadamente dos años trabajando con arquitecturas modulares. Ha sido un proceso de constante aprendizaje, donde he podido ver la evolución de los sistemas y la forma en que la modularidad facilita su gestión y mantenimiento.

2. **¿En qué tipo de proyectos ha implementado estas arquitecturas?**

   Respuesta: Principalmente en proyectos de microservicios. Esta arquitectura permite desarrollar aplicaciones escalables, flexibles y fáciles de mantener, lo que es clave para entornos donde la disponibilidad y el rendimiento son prioritarios.

3. **¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?**

   Respuesta: La mantenibilidad. Los sistemas modulares permiten realizar cambios en componentes individuales sin afectar el resto del sistema, lo que facilita el mantenimiento, reduce el tiempo de inactividad y mejora la eficiencia del equipo de desarrollo.

Categoría 2: Impacto en el Desarrollo de Software

4. **¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?**

   Respuesta: No he visto un impacto significativo en la reducción del tiempo de desarrollo. Si bien al inicio puede parecer que la modularidad agrega complejidad, con el tiempo se compensa gracias a la facilidad de mantenimiento y la reutilización de componentes.

5. **¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?**

   Respuesta: Sí, aunque en mi experiencia, el impacto no ha sido tan alto como en otros casos. Podría decir que los costos operativos se han reducido aproximadamente en un 3%. La modularidad ayuda a minimizar los errores y facilita la implementación de nuevas funcionalidades sin afectar la estabilidad del sistema.

6. **¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?**

   Respuesta: Permite una alta escalabilidad. Diseñar sistemas en módulos independientes hace que sea más sencillo agregar nuevas funcionalidades y optimizar el rendimiento según las necesidades del negocio.

Categoría 3: Reutilización de Código

7. **¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?**

   Respuesta: Aplicamos una modularización estricta. Diseñamos los módulos de manera que sean lo más independientes posible, permitiendo su reutilización en diferentes contextos sin generar dependencias innecesarias.

8. **¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?**

   Respuesta: Sí, la compatibilidad entre módulos es un reto común. A veces, las versiones o configuraciones difieren entre proyectos, lo que requiere ajustes adicionales para integrar los módulos sin afectar la funcionalidad.

9. **¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?**

   Respuesta: En promedio, ahorramos alrededor de 26 horas por módulo reutilizado. Aunque esto depende del nivel de personalización que se requiera, en general, disminuye significativamente el tiempo de desarrollo.

Categoría 4: Metodologías y Herramientas

10. **¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?**

    Respuesta: Utilizo Extreme Programming (XP) porque promueve el desarrollo iterativo y colaborativo, lo que facilita la adaptación a cambios y mejora la calidad del código modular.

11. **¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?**

    Respuesta: No, no tiene un impacto directo en la modularidad. Aunque estas prácticas mejoran la calidad del código en general, la modularidad depende más de una buena planificación arquitectónica y de un diseño adecuado de los módulos.

12. **¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?**

    Respuesta: Es útil pero no crítica. Una buena documentación ayuda a la integración y

reutilización de módulos, pero si el código está bien estructurado y sigue estándares claros, la documentación extensa puede ser innecesaria.

Categoría 5: Desafíos y Recomendaciones

13. **¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización? Respuesta:** La falta de capacitación es un gran obstáculo. No todos los desarrolladores están familiarizados con las mejores prácticas en modularidad, lo que puede dificultar su adopción y generar inconsistencias en el diseño.
14. **¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?**

    Respuesta: Es fundamental invertir en capacitación para el equipo. Comprender los principios de la modularidad desde el inicio evitará errores y hará que la transición sea más fluida.
15. **¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?**

    Respuesta: La automatización de procesos de pruebas y despliegue ayudaría a optimizar la modularidad y reducir errores. Contar con herramientas de integración continua puede facilitar la reutilización y la gestión eficiente de los módulos.

---

9. Entrevista a Diego Rojas Robles (Arquitecto de Software, 8 años de experiencia)

Categoría 1: Experiencia General con Arquitecturas Modulares

1. **¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?**

Respuesta: Llevo aproximadamente cinco años trabajando con arquitecturas modulares. Desde que comencé a utilizarlas, he podido notar la diferencia en la organización del código y la facilidad para mantener y escalar sistemas complejos.

2. **¿En qué tipo de proyectos ha implementado estas arquitecturas?**

   Respuesta: He trabajado principalmente en aplicaciones empresariales. La modularidad ha sido clave para gestionar grandes volúmenes de datos, permitir la interoperabilidad entre sistemas y mejorar la mantenibilidad del software.

3. **¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?**

   Respuesta: La reutilización de código. Los módulos bien diseñados pueden utilizarse en distintos proyectos, lo que reduce el tiempo de desarrollo y mejora la eficiencia del equipo.

Categoría 2: Impacto en el Desarrollo de Software

4. **¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?**

   Respuesta: Ha habido una ligera mejora en los tiempos de desarrollo. Aunque la implementación inicial puede ser más lenta debido a la necesidad de diseñar correctamente los módulos, a largo plazo se traduce en ahorros significativos de tiempo y esfuerzo.

5. **¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?**

   Respuesta: Sí, aproximadamente un 4%. Aunque el porcentaje puede parecer bajo, la modularidad permite reducir costos de mantenimiento y actualización del software, evitando gastos innecesarios en correcciones o rediseños.

6. **¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?**

Respuesta: En algunos casos, la escalabilidad es un desafío. Si bien los módulos independientes facilitan la adición de nuevas funcionalidades, la integración y coordinación entre ellos puede volverse compleja si no se diseña adecuadamente desde el principio.

Categoría 3: Reutilización de Código

7. **¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?**

   Respuesta: Hacemos un fuerte uso de librerías compartidas. Estas permiten estandarizar funcionalidades comunes y reducir la duplicación de código, lo que facilita la integración y mejora la eficiencia del desarrollo.

8. **¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?**

   Respuesta: Sí, la resistencia al cambio ha sido uno de los mayores desafíos. Algunos desarrolladores prefieren crear soluciones desde cero en lugar de adaptar módulos existentes, lo que puede dificultar la adopción de un enfoque modular.

9. **¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?**

   Respuesta: En promedio, ahorramos alrededor de 39 horas por módulo reutilizado. Este tiempo puede variar según la complejidad de la funcionalidad y la necesidad de adaptación a cada nuevo proyecto.

Categoría 4: Metodologías y Herramientas

10. **¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?**

Respuesta: Utilizo Extreme Programming (XP) porque fomenta la colaboración, la refactorización continua y la entrega incremental, lo que encaja muy bien con la modularidad.

11. **¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?**

    Respuesta: Sí, mejora la calidad del código. La programación en pares permite detectar errores y mejorar la estructura del código, mientras que la refactorización continua evita que el código modular se vuelva obsoleto o desordenado.

12. **¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?**

    Respuesta: No es prioritaria, pero sigue siendo necesaria. Un código bien escrito y estructurado puede ser suficiente para comprender su uso, aunque contar con documentación ayuda a nuevos desarrolladores a integrarse más rápidamente.

Categoría 5: Desafíos y Recomendaciones

13. **¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización?**

    Respuesta: La complejidad de implementación. Diseñar una arquitectura modular eficiente requiere una planificación cuidadosa y un equipo bien capacitado para evitar problemas de integración.

14. **¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?**

    Respuesta: La capacitación del equipo es fundamental. Sin un conocimiento sólido sobre modularidad, la implementación puede volverse desordenada y generar más problemas que soluciones.

15. ¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?

Respuesta: La automatización de pruebas e integración continua sería clave. Esto permitiría garantizar que los módulos sigan funcionando correctamente a medida que evolucionan los proyectos.

---

10. Entrevista a Raúl Torres Martínez (Consultor Independiente en Arquitectura de Software, 9 años de experiencia)

Categoría 1: Experiencia General con Arquitecturas Modulares

1. **¿Cuánto tiempo lleva trabajando con arquitecturas modulares en proyectos de software?**

   Respuesta: Llevo nueve años trabajando con arquitecturas modulares. Durante este tiempo, he podido ver su evolución y el impacto positivo que tiene en la eficiencia y mantenibilidad de los sistemas.

2. **¿En qué tipo de proyectos ha implementado estas arquitecturas?**

   Respuesta: Principalmente en aplicaciones móviles. La modularidad en este tipo de proyectos permite una mejor organización del código y facilita la actualización y mantenimiento sin afectar toda la aplicación.

3. **¿Cuál considera que es el mayor beneficio de las arquitecturas modulares frente a las monolíticas?**

   Respuesta: La mantenibilidad. Al dividir el sistema en módulos bien definidos, es mucho más fácil realizar mejoras, corregir errores y escalar funcionalidades sin afectar la estabilidad del sistema.

Categoría 2: Impacto en el Desarrollo de Software

4. **¿Cómo describiría el impacto de la modularidad en los tiempos de desarrollo?**

   Respuesta: No he notado un impacto significativo en la reducción de tiempos de desarrollo. Si bien la modularidad facilita la reutilización de componentes, su implementación inicial puede ser más compleja y requerir una mayor planificación.

5. **¿Ha notado una reducción en los costos operativos gracias a la modularidad? ¿En qué porcentaje, aproximadamente?**

   Respuesta: Sí, aproximadamente un 20%. La modularidad permite optimizar los recursos, reduciendo costos en mantenimiento, soporte y tiempo de desarrollo de nuevas funcionalidades.

6. **¿Cómo afecta la modularidad a la escalabilidad de los sistemas en los que trabaja?**

   Respuesta: La modularidad permite una alta escalabilidad. Al separar las funcionalidades en módulos independientes, es posible mejorar y expandir el sistema sin afectar el resto de la aplicación.

Categoría 3: Reutilización de Código

7. **¿Qué estrategias utiliza para fomentar la reutilización de código en sus proyectos?**

   Respuesta: Implementamos una separación clara de responsabilidades. Cada módulo está diseñado para cumplir una función específica y puede integrarse fácilmente en otros proyectos sin modificaciones mayores.

8. **¿Ha enfrentado desafíos al intentar reutilizar módulos en diferentes proyectos? ¿Cuáles?**

Respuesta: Sí, la compatibilidad entre módulos ha sido un desafío. Diferencias en dependencias y versiones pueden hacer que un módulo reutilizado requiera ajustes adicionales para su correcta integración.

9. **¿Cuánto tiempo, en promedio, ahorra al reutilizar módulos ya probados en lugar de desarrollarlos desde cero?**

   Respuesta: Aproximadamente 91 horas por módulo. El ahorro de tiempo depende de la complejidad del módulo y del grado de adaptación necesario para su implementación.

Categoría 4: Metodologías y Herramientas

10. **¿Qué herramientas y metodologías utiliza para diseñar y gestionar arquitecturas modulares (por ejemplo, C4, XP)?**

    Respuesta: Utilizo Extreme Programming (XP) porque fomenta la refactorización y el desarrollo iterativo, lo que permite mejorar continuamente los módulos y mantener un código limpio y eficiente.

11. **¿Considera que la programación en pares y la refactorización continua contribuyen a la calidad del código modular? ¿Por qué?**

    Respuesta: No, no tiene un impacto directo en la modularidad. Si bien estas prácticas mejoran la calidad general del código, la modularidad depende más del diseño arquitectónico y la planificación del sistema.

12. **¿Qué importancia le da a la documentación en el diseño y la reutilización de módulos?**

    Respuesta: No es prioritaria. Un código bien estructurado y con estándares claros puede ser suficiente para entender su funcionamiento sin necesidad de documentación extensa.

Categoría 5: Desafíos y Recomendaciones

13. **¿Cuáles son las principales barreras para la implementación de arquitecturas modulares en su organización?**

    Respuesta: La complejidad de implementación. Adoptar una arquitectura modular requiere una planificación detallada y un equipo capacitado para evitar problemas de integración y compatibilidad.

14. **¿Qué recomendaría a una empresa que quiere adoptar arquitecturas modulares por primera vez?**

    Respuesta: Definir estándares desde el inicio. Contar con reglas claras para la modularización ayuda a evitar inconsistencias y facilita la adopción de buenas prácticas.

15. **¿Qué mejoras incluiría en sus estrategias actuales de modularidad y reutilización de código?**

    Respuesta: Mayor documentación y estandarización. Esto facilitaría la integración de módulos en diferentes proyectos y reduciría la curva de aprendizaje para nuevos desarrolladores.