**20 resúmenes**

**Nombre:**

**Johan Calderón Perdomo**

**Instructor:**

**Jesús Ariel González**

**Servicio Nacional de Aprendizaje SENA Sede Industria**

**Neiva – Huila**

**14-09-2024**

Johan Calderón Perdomo



Hola, mi nombre es Johan Calderón Perdomo, tengo 18 y soy un apasionado por la tecnología y el deporte. Me gusta estar investigando sobre nuevas tecnologías, lo que me ha llevado a enfocarme en el desarrollo de software. Soy técnico en desarrollo de software y actualmente estudio un tecnólogo en análisis y desarrollo de software, así como ingeniería de sistemas. Además, formo parte del equipo de fútbol de salón del Sena, donde soy arquero. Desde muy joven, he sentido curiosidad por cómo funcionan las cosas, lo que me motivó a estudiar programación y a seguir aprendiendo constantemente. Mi meta es seguir creciendo en el campo de la tecnología y contribuir con soluciones innovadoras a problemas cotidianos, al mismo tiempo que sigo practicando y disfrutando del deporte. Aparte de mi pasión por el desarrollo de software, disfruto mucho del tiempo libre estando con mi familia jugando videos juegos y comiendo hamburguesas. Además de mis estudios y mi participación en el equipo de fútbol de salón, he desarrollado habilidades en el manejo de diferentes lenguajes de programación como C#, Java, Kotlin, y frameworks como Angular, lo que me ha permitido trabajar en proyectos tanto personales como académicos.

**Revisión de elementos conceptuales para la representación de las arquitecturas de referencias de software**

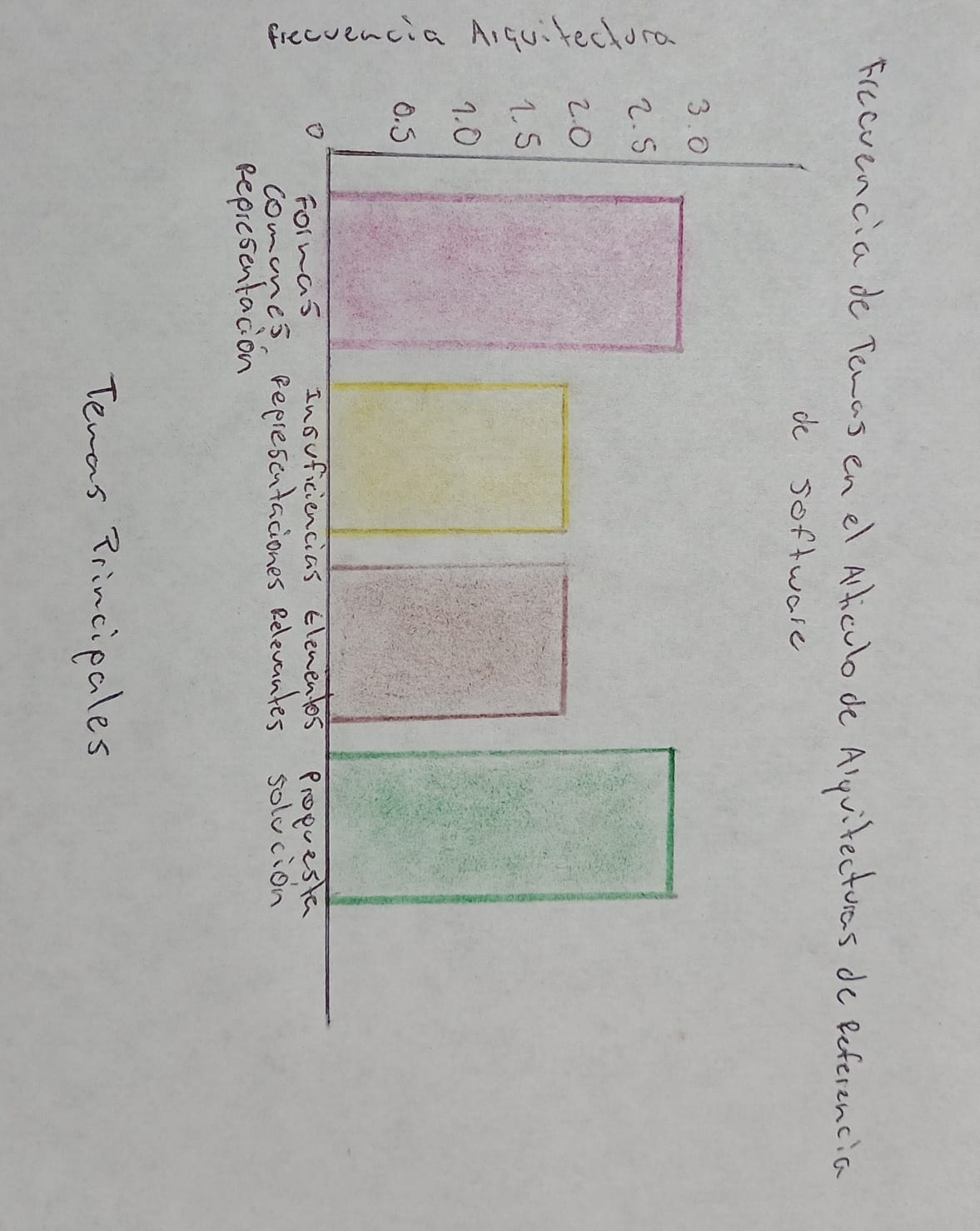
**Resumen**

La Arquitectura de Software organiza un sistema a nivel alto, definiendo elementos y sus relaciones. Facilita el desarrollo al reutilizar componentes mediante arquitecturas de referencia. Ha evolucionado a través de enfoques estructurales, patrones y escenarios, guiando el diseño y la comprensión de sistemas complejos.

**Reflexión**

La Arquitectura de Software organiza sistemas definiendo sus elementos y relaciones, facilitando el desarrollo mediante la reutilización de componentes. Su evolución, basada en enfoques estructurales y patrones, permite diseñar sistemas complejos de manera más eficiente y comprensible.

[**Enlace Artículo**](http://scielo.sld.cu/scielo.php?pid=S2227-18992019000100143&script=sci_arttext)



**Análisis comparativo de Patrones de Diseño de Software**

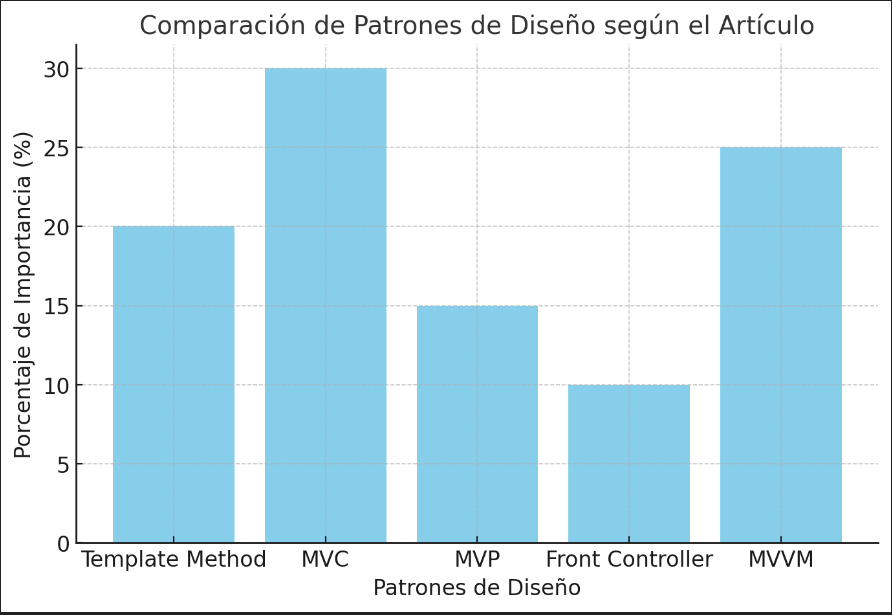
**Resumen**

Los patrones de diseño evitan duplicación de código y mejoran la calidad del software. El artículo compara varios patrones, concluyendo que no hay uno superior, ya que cada uno tiene un propósito específico, facilitando el mantenimiento y el modularidad del sistema.

**Reflexión**

Los patrones de diseño mejoran la calidad del software al evitar la duplicación de código y facilitar su mantenimiento. Cada patrón tiene un propósito único, lo que hace que no exista uno superior a otro. La comparación muestra que la elección del patrón depende del contexto y las necesidades del sistema, promoviendo el modularidad.

[**Enlace Artículo**](https://dialnet.unirioja.es/servlet/articulo?codigo=9042927)



**Análisis de secuencia discretas para la detección de Patrones de Diseño de Software**

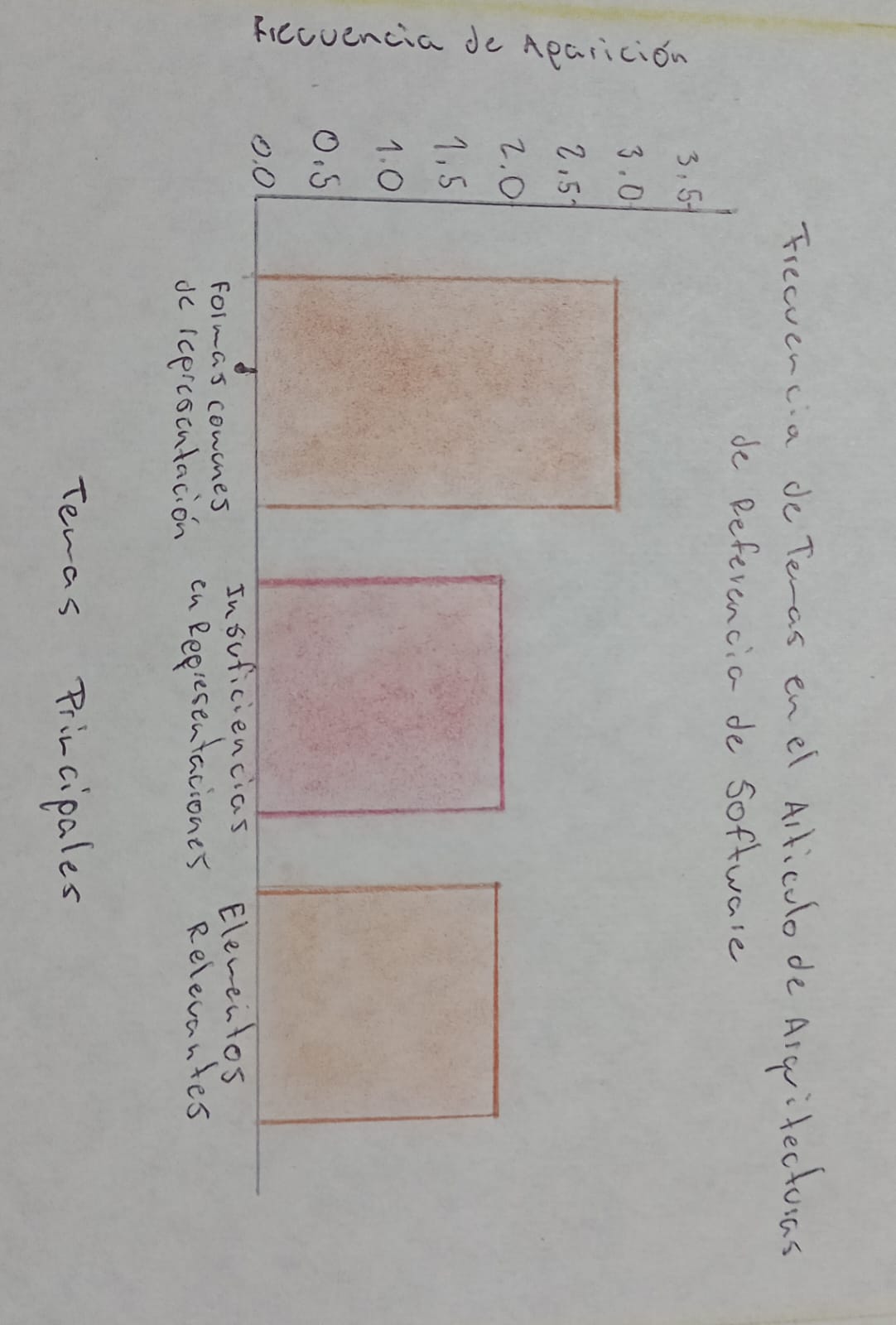
**Resumen**

Los Agentes de Interfaz ofrecen asistencia personalizada en software. Este trabajo propone usar Modelos de Márkov para detectar patrones de diseño en herramientas CASE, mejorando la precisión y tiempos de respuesta, y generando recomendaciones basadas en las acciones del usuario.

**Reflexión**

El uso de Modelos de Márkov para detectar patrones de diseño en herramientas CASE representa un avance significativo en la personalización del software. Al mejorar la precisión y los tiempos de respuesta, los Agentes de Interfaz pueden ofrecer recomendaciones más efectivas basadas en las acciones del usuario, optimizando así la experiencia de interacción con el software.

[**Enlace Artículo**](https://sedici.unlp.edu.ar/bitstream/handle/10915/152663/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y)



**Patrones de Diseño**

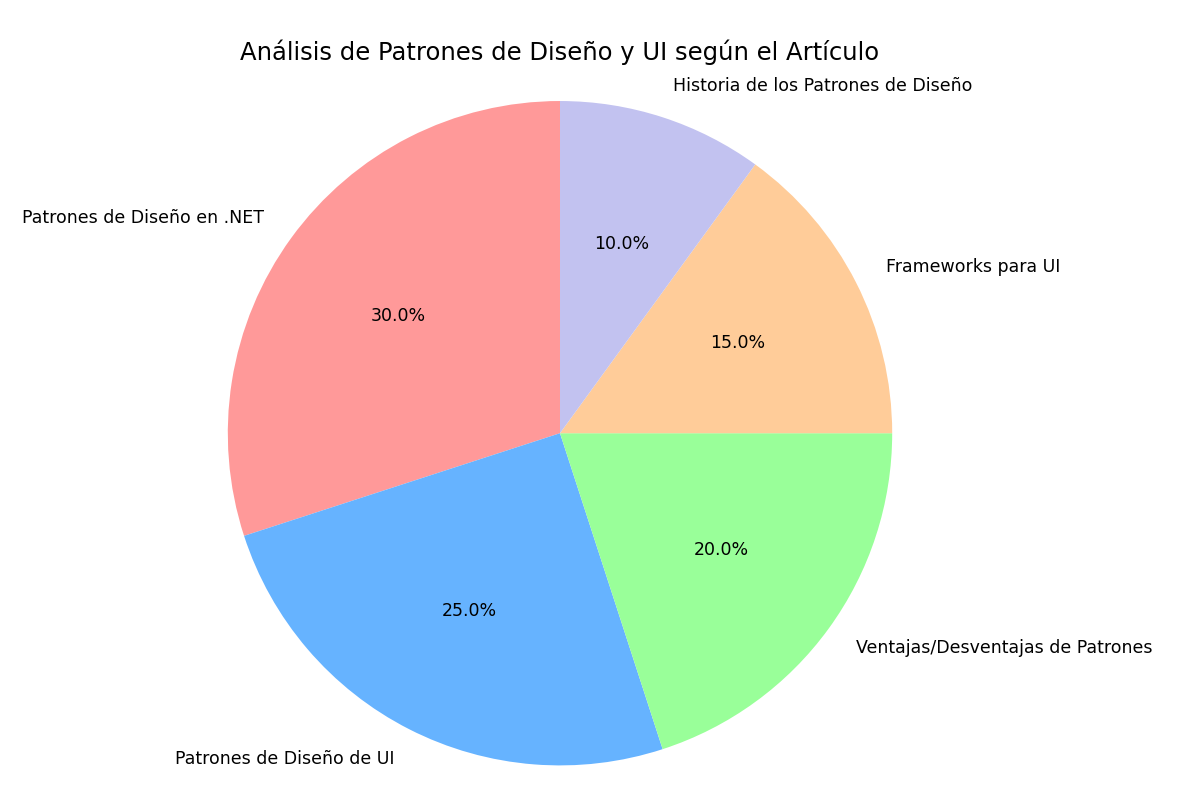
**Resumen**

Los patrones de diseño, popularizados por el libro Design Patterns en 1994, proporcionan soluciones reutilizables para crear software escalable y flexible. Este documento aborda dos tipos de patrones: los de diseño para la implementación en código, explicando seis patrones usados en .NET, y los de diseño de interfaz de usuario, enfocados en mejorar la experiencia del usuario en sitios web, con ejemplos y frameworks recomendados para facilitar el desarrollo.

**Reflexión**

Los patrones de diseño, popularizados en 1994, son fundamentales para crear software escalable y flexible. Al abordar tanto patrones de implementación en código como de diseño de interfaz, se proporciona un enfoque integral que mejora la calidad del desarrollo y la experiencia del usuario, facilitando así la adopción de soluciones más efectivas.

[**Enlace Artículo**](https://www.academia.edu/download/57338244/Patrones.pdf)



**Patrones de Diseño: Mejores Experiencias y Productos Fácilmente Escalables**

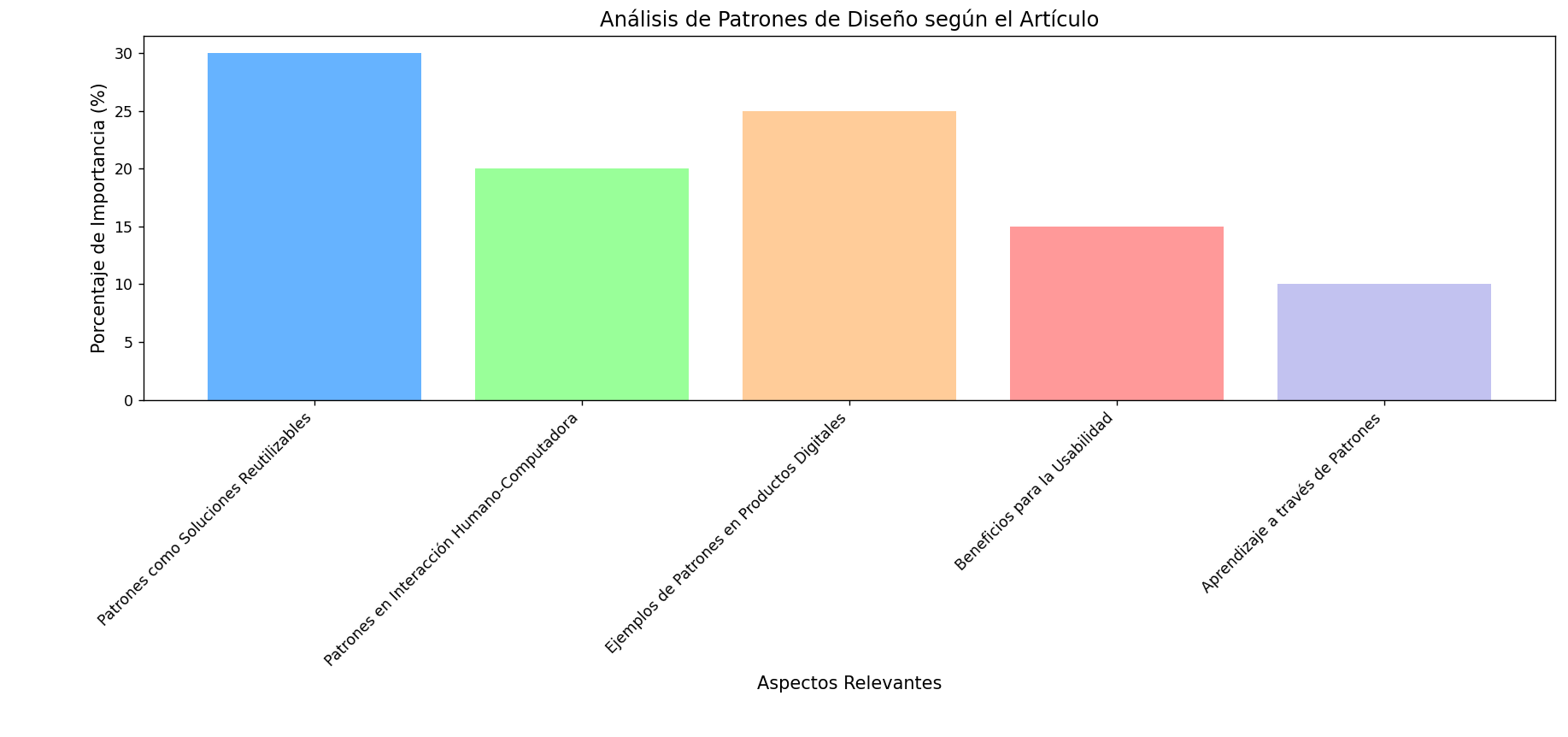
**Resumen**

Los patrones de diseño en experiencias digitales facilitan el uso para los usuarios, quienes ya están familiarizados con ciertos estándares. Siguiendo estos patrones, se mejora la usabilidad y se evita una curva de aprendizaje empinada, lo que ayuda a que los usuarios se concentren en el contenido del sitio en lugar de en cómo usarlo. Esto también aumenta la eficiencia del equipo de desarrollo.

**Reflexión**

Los patrones de diseño en experiencias digitales son clave para mejorar la usabilidad, permitiendo que los usuarios se sientan cómodos con las interfaces familiares. Esto no solo reduce la curva de aprendizaje, sino que también permite a los equipos de desarrollo concentrarse en crear contenido valioso, aumentando así la eficiencia en el desarrollo de productos.

[**Enlace Artículo**](https://es.modyo.com/blog/patrones-de-diseno-mejores-experiencias-y-productos-facilmente-escalables)



**Software Architecture**

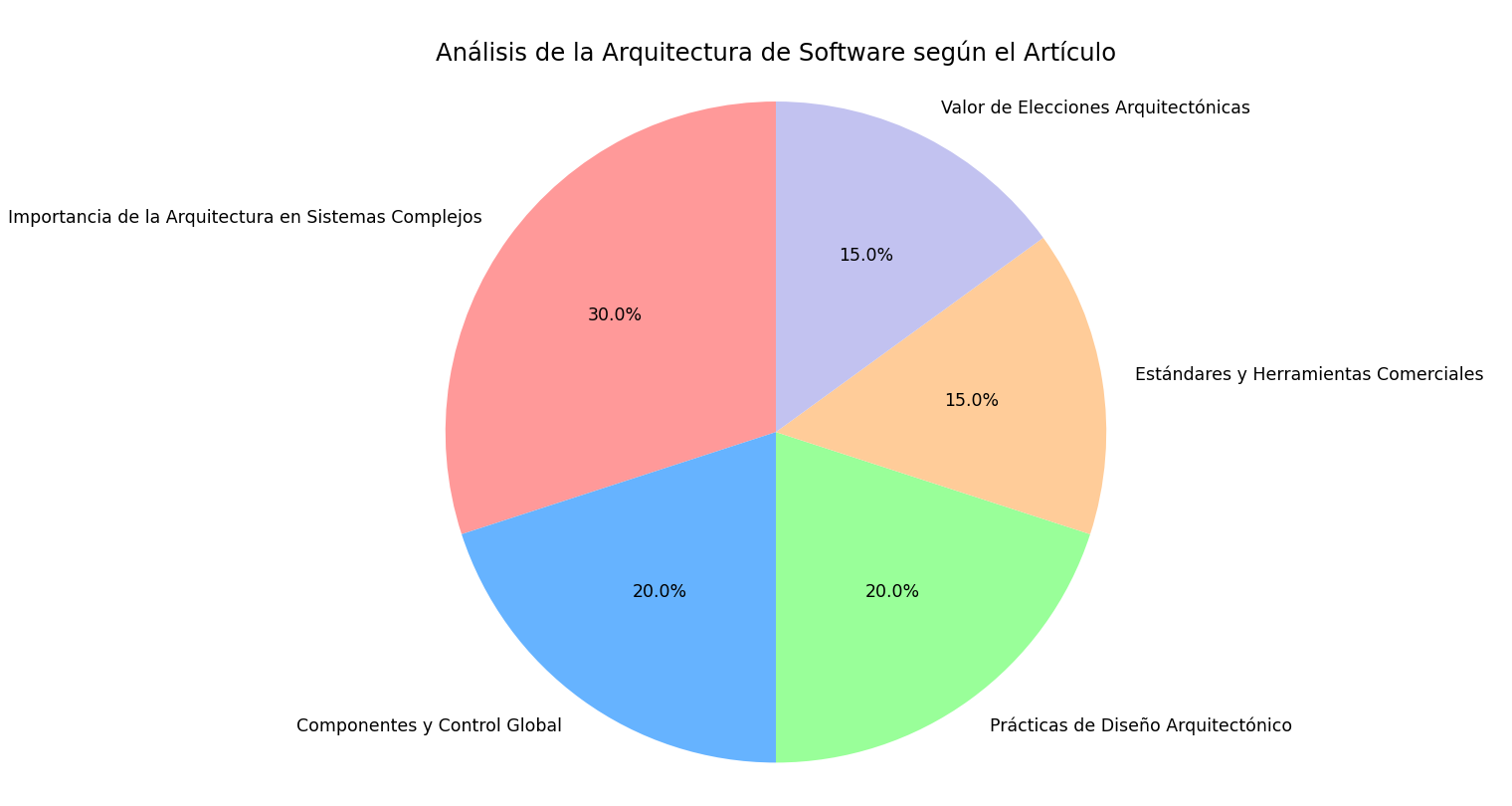
**Resumen**

El diseño arquitectónico en software es crucial para manejar la creciente complejidad de los sistemas. Una buena arquitectura garantiza requisitos clave como rendimiento, escalabilidad y confiabilidad. En la última década, ha ganado importancia con la creación de libros, herramientas, cursos y estándares formales que ayudan a reutilizar y mejorar los diseños arquitectónicos.

**Reflexión**

El diseño arquitectónico es esencial para gestionar la complejidad creciente de los sistemas de software. Una buena arquitectura no solo asegura rendimiento y escalabilidad, sino que también ha evolucionado con recursos formales que promueven la reutilización, lo que es vital para adaptarse a las demandas cambiantes del entorno tecnológico.

[**Enlace Artículo**](https://kilthub.cmu.edu/articles/journal_contribution/Software_Architecture/6609593/files/12101711.pdf)



**A Systematic Review on Software Architectures for IoT Systems and Future Direction to the Adoption of Microservices Architecture**

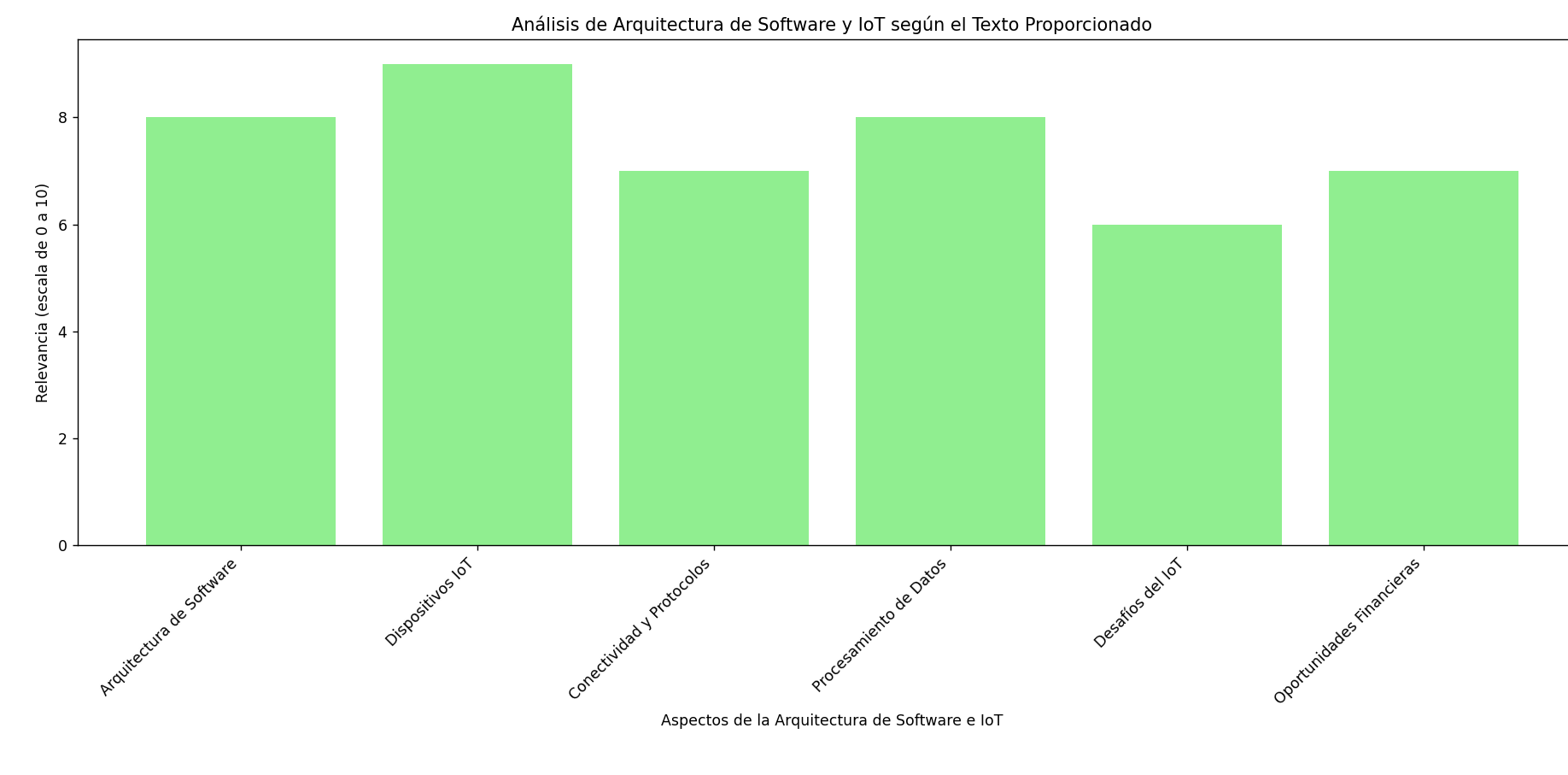
**Resumen**

Este estudio revisa la investigación sobre la adopción de la Arquitectura de Microservicios (MSA) en sistemas de Internet de las Cosas (IoT). Analiza 140 artículos para identificar patrones arquitectónicos, desafíos y soluciones en IoT. Destaca trece patrones de MSA, clasifica arquitecturas en nueve categorías, y aborda veintitrés desafíos principales. Los hallazgos ofrecen recomendaciones para mejorar la arquitectura y el desarrollo de software de IoT, beneficiando a la industria y a la comunidad académica.

**Reflexión**

El estudio sobre la Arquitectura de Microservicios en sistemas IoT resalta la importancia de identificar patrones y desafíos en este contexto emergente. Al analizar 140 artículos, se ofrecen recomendaciones prácticas que no solo benefician a la industria, sino que también enriquecen la comunidad académica, fomentando un desarrollo más eficiente y robusto de software en el ámbito IoT.

[**Enlace Artículo**](https://www.proquest.com/docview/2933155446/E860FCE7DDB74622PQ/23?accountid=31491&sourcetype=Scholarly%20Journals)



**Software Architectures - Present and Visions**

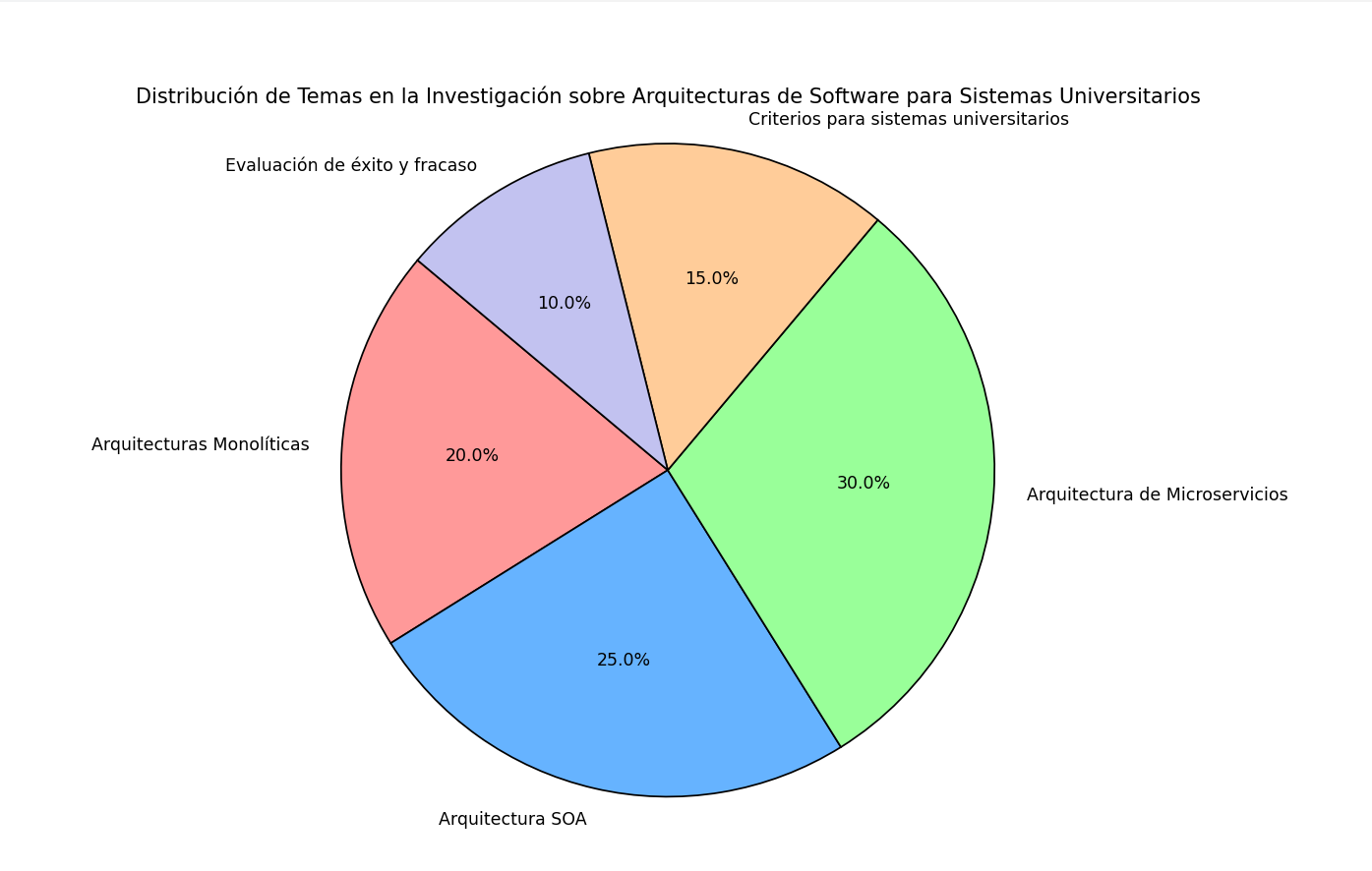
**Resumen**

El artículo analiza diferentes tipos de arquitecturas de sistemas de software, como monolíticas, SOA y microservicios, y evalúa cuál es más adecuada para sistemas universitarios. Examina desafíos específicos, factores de éxito y causas de fracaso, y ofrece una recomendación objetiva para la implementación en el ámbito académico.

**Reflexión**

El análisis de diferentes arquitecturas de software, como monolíticas, SOA y microservicios, es esencial para seleccionar la más adecuada en contextos universitarios. Al evaluar desafíos y factores de éxito, el artículo proporciona recomendaciones objetivas que pueden guiar la implementación efectiva de sistemas en el ámbito académico, maximizando su rendimiento y sostenibilidad.

[**Enlace Artículo**](https://www.proquest.com/docview/1758012314/C6BB69EB2E54D0EPQ/1?accountid=31491&sourcetype=Scholarly%20Journals)



**A scalable architecture for automated monitoring of microservices**

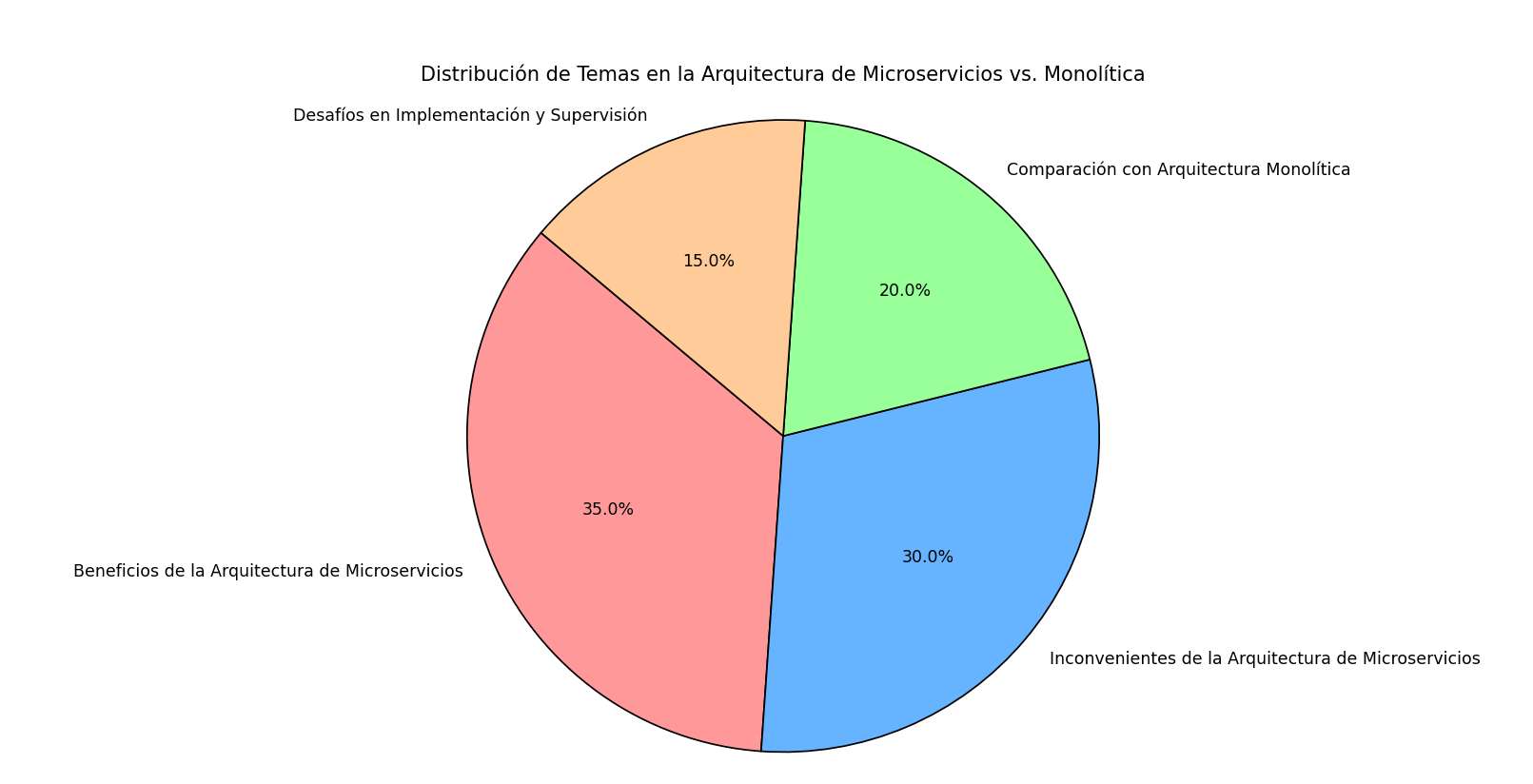
**Resumen**

El artículo propone una arquitectura para monitorear microservicios, registrando métricas clave de rendimiento y utilizando algoritmos avanzados para clasificar eventos y automatizar decisiones. Emplea Prometheus para métricas a corto plazo, OpenTSDB para almacenamiento a largo plazo y RabbitMQ para la transmisión de mensajes.

**Reflexión**

La propuesta de una arquitectura para el monitoreo automatizado de microservicios destaca la importancia de registrar métricas clave y utilizar algoritmos avanzados para la toma de decisiones. Al integrar herramientas como Prometheus y RabbitMQ, se logra un sistema que no solo optimiza el rendimiento, sino que también facilita la gestión eficiente de eventos en entornos de microservicios.

[**Enlace Artículo**](https://www.proquest.com/docview/2172010739/38B8A9878E894A8APQ/1?accountid=31491&sourcetype=Scholarly%20Journals)



**Implementation of a Universal Framework Using Design Patterns for Application Development on Microcontrollers**

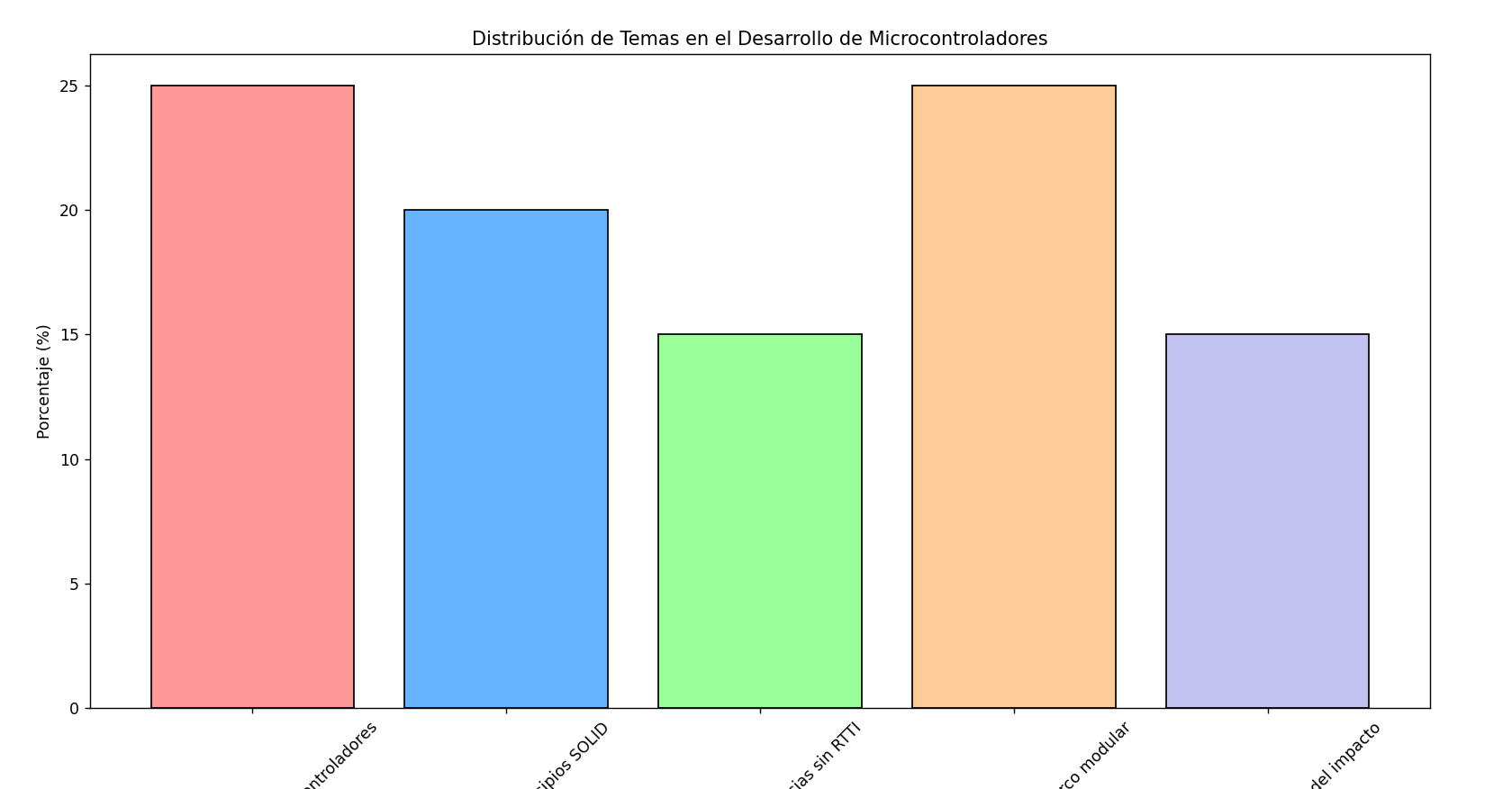
**Resumen**

El artículo presenta un marco modular para microcontroladores que aplica principios modernos y patrones de diseño para mejorar la calidad del software en sistemas integrados e IoT. Este enfoque permite la comunicación independiente entre módulos, integración flexible de hardware y adaptabilidad a diferentes entornos de desarrollo, mejorando la sostenibilidad y escalabilidad a largo plazo.

**Reflexión**

La implementación de un marco modular para microcontroladores que utiliza patrones de diseño modernos representa un avance significativo en la calidad del software en sistemas IoT. Este enfoque promueve la comunicación independiente entre módulos y la adaptabilidad a diversos entornos de desarrollo, lo que garantiza una mayor sostenibilidad y escalabilidad a largo plazo.

[**Enlace Artículo**](https://www.proquest.com/docview/3059721054/339A7C7B154647BAPQ/4?accountid=31491&sourcetype=Scholarly%20Journals)



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

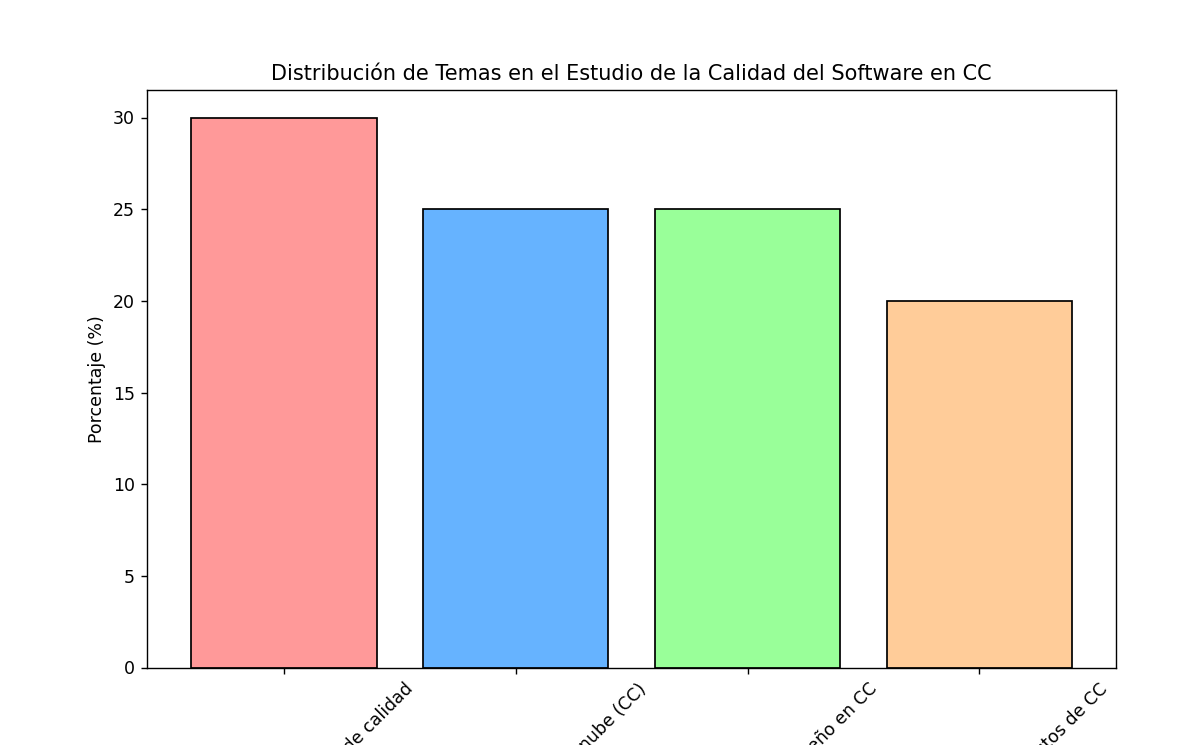
**Resumen**

El trabajo presenta una herramienta gráfica para diseñar arquitecturas de software en aplicaciones web, basada en patrones de diseño. Se enfoca en cómo las funcionalidades del software se relacionan con los atributos de calidad y utiliza el paradigma de computación en la nube para ofrecer acceso a recursos configurables de manera eficiente.

**Reflexión**

La herramienta gráfica presentada para modelar arquitecturas de software en la nube destaca la relación entre funcionalidades y atributos de calidad. Al emplear patrones de diseño, se facilita el acceso eficiente a recursos configurables, lo que optimiza el desarrollo de aplicaciones web y maximiza su rendimiento en entornos de computación en la nube.

[**Enlace Artículo**](https://www.proquest.com/docview/2343015882/90C25C73A9E343F0PQ/6?accountid=31491&sourcetype=Scholarly%20Journals)



**Incidencia de los Patrones de Diseño de Software en la Seguridad de Aplicaciones Web**

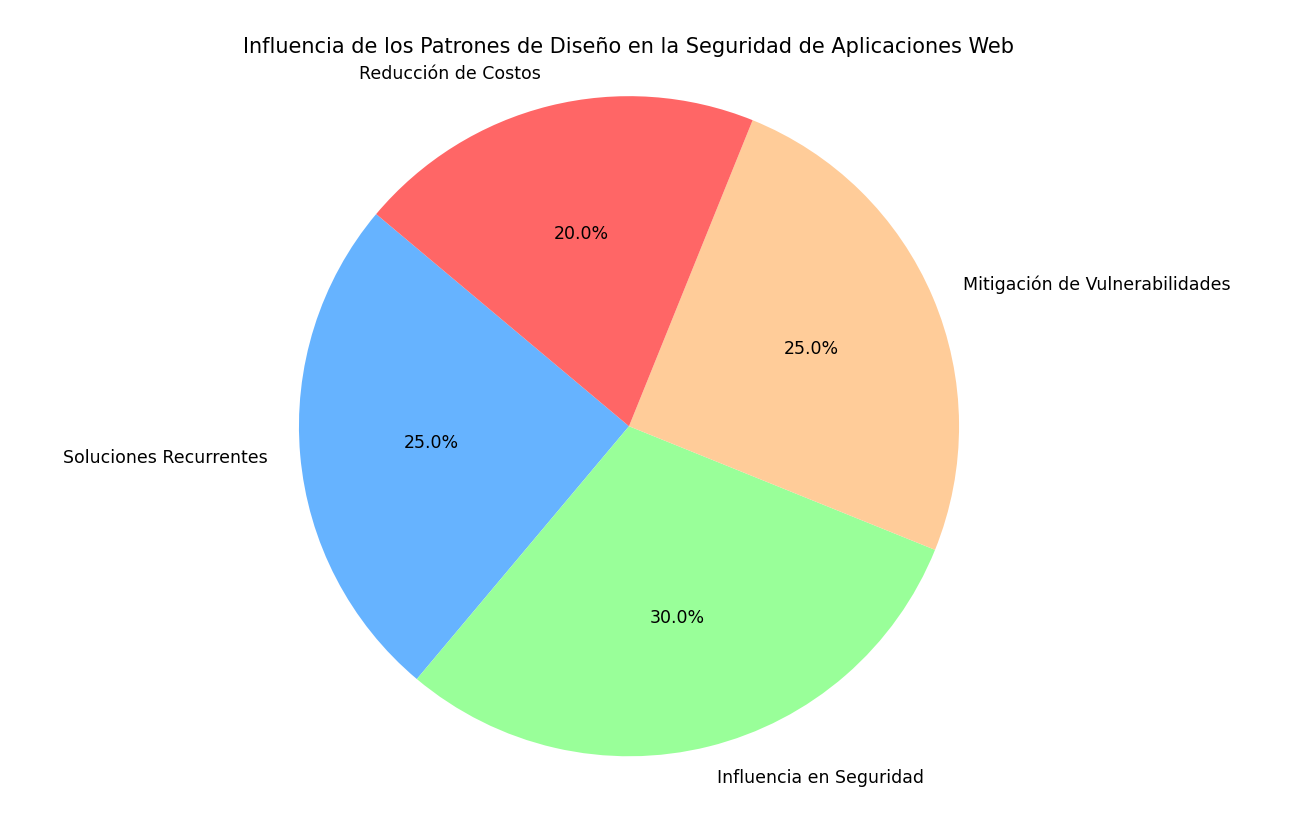
**Resumen**

Los patrones de diseño ayudan a estructurar y organizar el código de manera eficiente, y pueden mejorar la seguridad de aplicaciones web al abordar problemas como validación de datos y autenticación. La investigación busca determinar cómo estos patrones ayudan a mitigar vulnerabilidades, reduciendo costos y mejorando la mantenibilidad del software.

**Reflexión**

El estudio sobre la incidencia de patrones de diseño en la seguridad de aplicaciones web revela su papel crucial en la organización del código y en la mitigación de vulnerabilidades. Al abordar problemas como la validación de datos, estos patrones no solo mejoran la seguridad, sino que también contribuyen a la sostenibilidad y mantenibilidad del software, lo que se traduce en una reducción de costos a largo plazo.

[**Enlace Artículo**](https://www.investigarmqr.com/ojs/index.php/mqr/article/view/886)



**Integración de patrones de seguridad y patrones de diseño J2EE**

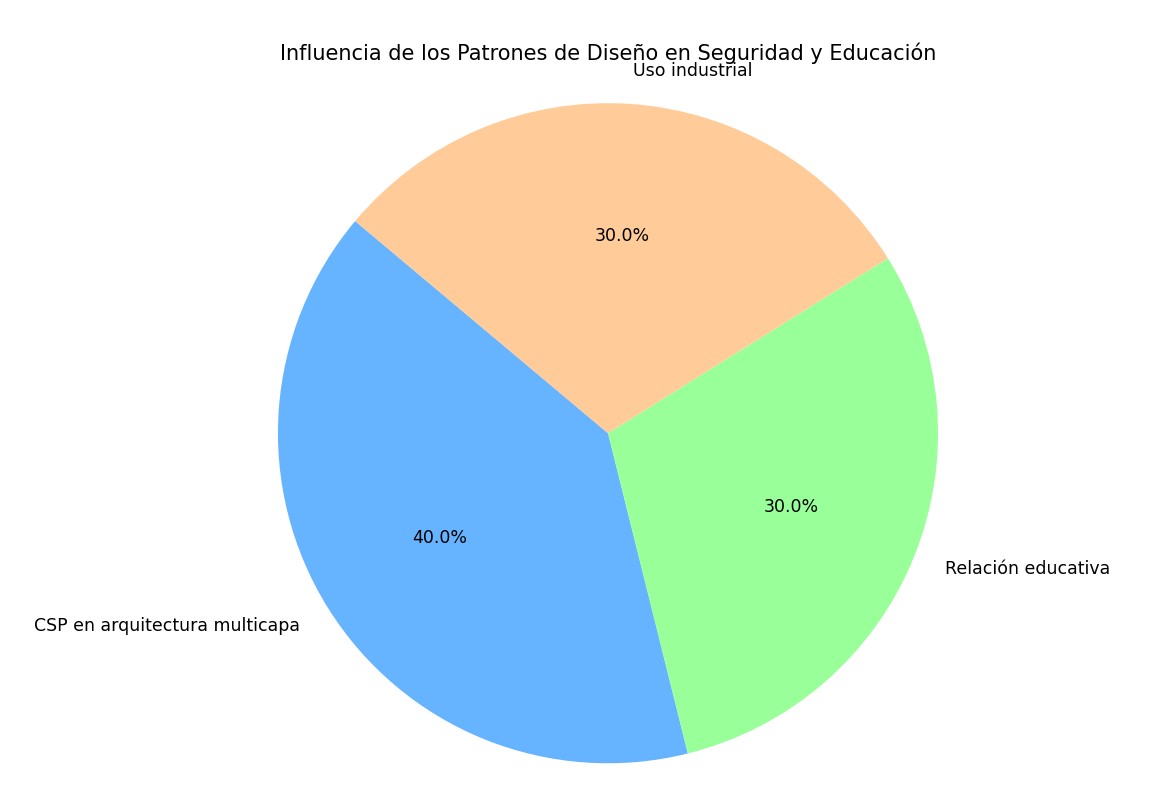
**Resumen**

El trabajo analiza dos catálogos populares de patrones de diseño de software de Sun Microsystems (Oracle), define sus relaciones para un uso coherente y presenta casos prácticos de aplicaciones que integran patrones de ambos catálogos.

**Reflexión**

El análisis de la integración de patrones de seguridad y de diseño J2EE es crucial para lograr un desarrollo de software más coherente y seguro. Al definir relaciones y presentar casos prácticos, el trabajo proporciona un marco útil que permite a los desarrolladores implementar soluciones efectivas, optimizando tanto la funcionalidad como la protección de las aplicaciones.

[**Enlace Articulo**](https://docta.ucm.es/entities/publication/14b254d6-2bc6-481b-abd8-460469bf2dd0)



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

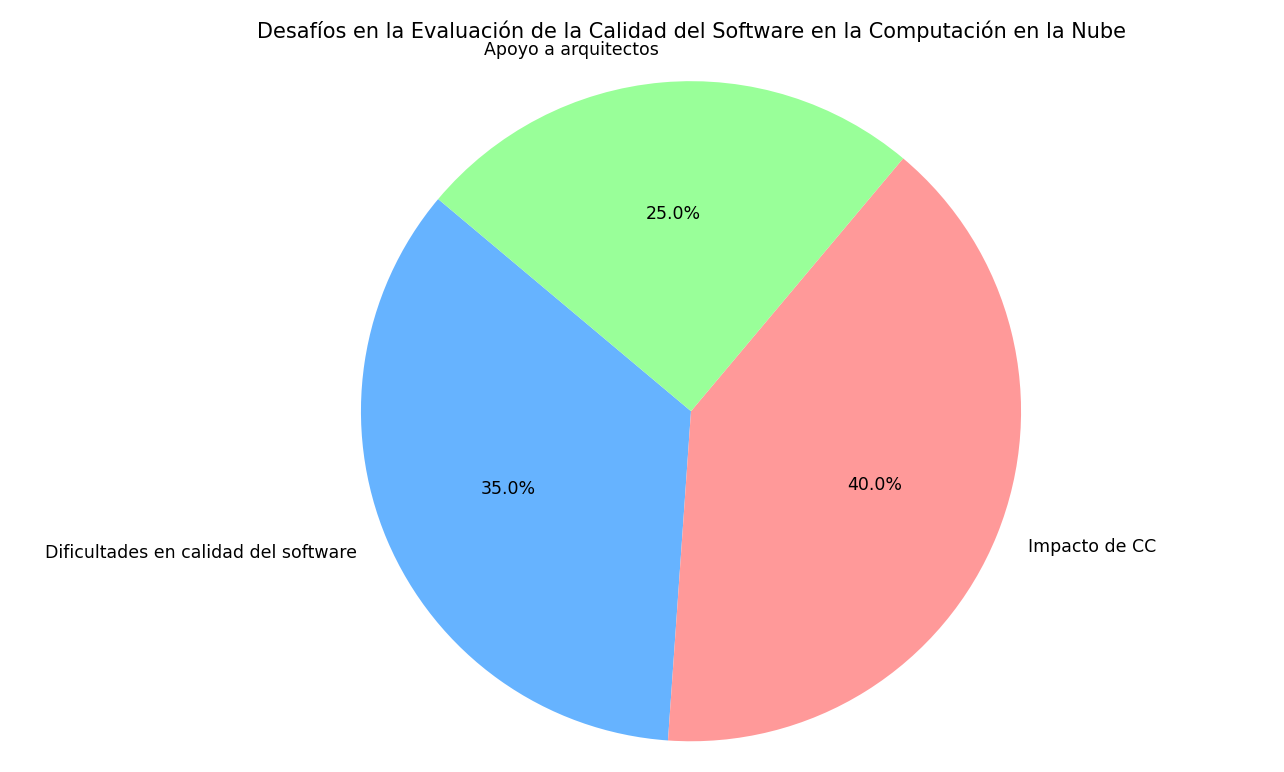
**Resumen**

El trabajo presenta una herramienta gráfica para diseñar arquitecturas de software en aplicaciones web, utilizando un modelo de soporte. Destaca que las funcionalidades del software se separan de los atributos de calidad y que la computación en la nube facilita el acceso a recursos configurables de manera eficiente.

**Reflexión**

La presentación de una herramienta gráfica para modelar arquitecturas de software en la nube subraya la importancia de separar las funcionalidades de los atributos de calidad. Este enfoque no solo facilita el diseño de aplicaciones web, sino que también aprovecha los recursos configurables de la computación en la nube, mejorando así la eficiencia y la flexibilidad en el desarrollo.

[**Enlace Articulo**](https://www.proquest.com/docview/2343015882/82B908C636404259PQ/4?accountid=31491&sourcetype=Scholarly%20Journals)



**Application of Design Patterns in Service Oriented Architecture**

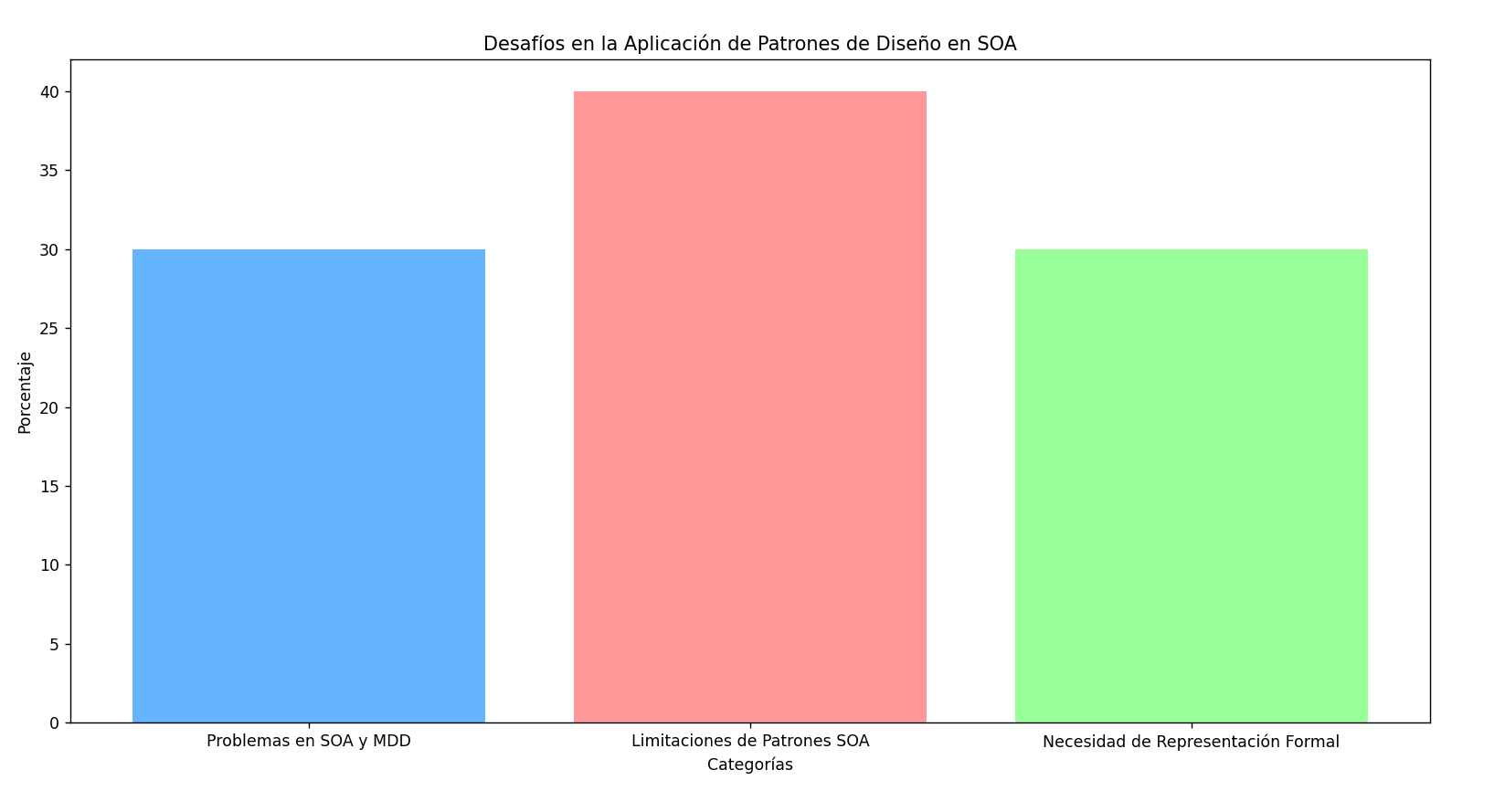
**Resumen**

El artículo investiga cómo los patrones de diseño informales de Arquitectura Orientada a Servicios (SOA) pueden resolver problemas en el desarrollo de software basado en modelos. Utiliza análisis orientado a objetos y teorías de categorías para formalizar estos patrones, mejorando su aplicación en el modelado de sistemas.

**Reflexión**

El artículo sobre la aplicación de patrones de diseño en Arquitectura Orientada a Servicios (SOA) destaca cómo estos patrones pueden abordar problemas comunes en el desarrollo de software. Al formalizar patrones informales mediante análisis orientado a objetos y teorías de categorías, se potencia su aplicabilidad en el modelado de sistemas, lo que mejora la calidad y la eficiencia del proceso de desarrollo.

[**Enlace Articulo**](https://www.proquest.com/docview/2162719666/579077592A2F4496PQ/102?accountid=31491)



**Design and Implementation of a Java-Based Client-Server Application**

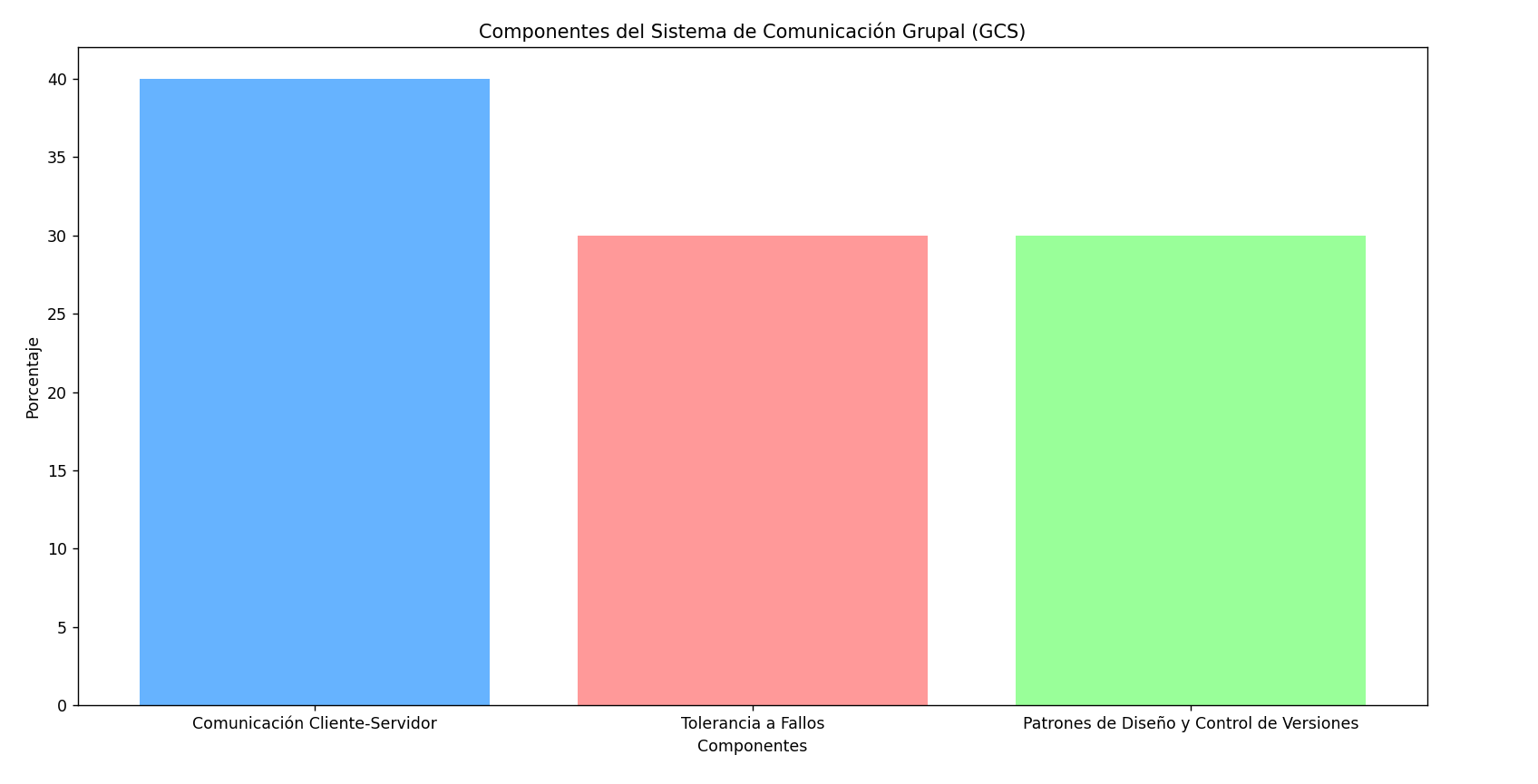
**Resumen**

El informe describe el desarrollo del Sistema de Comunicación Grupal (GCS) en Java, que utiliza sockets y protocolos de comunicación para gestionar grupos a través de una interfaz de línea de comandos. Destaca la tolerancia a errores, patrones de diseño y uso de control de versiones, proporcionando una visión completa de su arquitectura e implementación.

**Reflexión**

El informe sobre el Sistema de Comunicación Grupal (GCS) en Java ofrece una visión integral del desarrollo de aplicaciones cliente-servidor. Al destacar la tolerancia a errores, patrones de diseño y el control de versiones, se demuestra cómo una arquitectura bien implementada puede facilitar la gestión de grupos y mejorar la experiencia del usuario, asegurando la robustez del sistema.

[**Enlace Artículo**](https://www.proquest.com/docview/3040141605/579077592A2F4496PQ/111?accountid=31491&sourcetype=Working%20Papers)



**Modeling Design Patterns for Semi-Automatic Reuse in System Design**

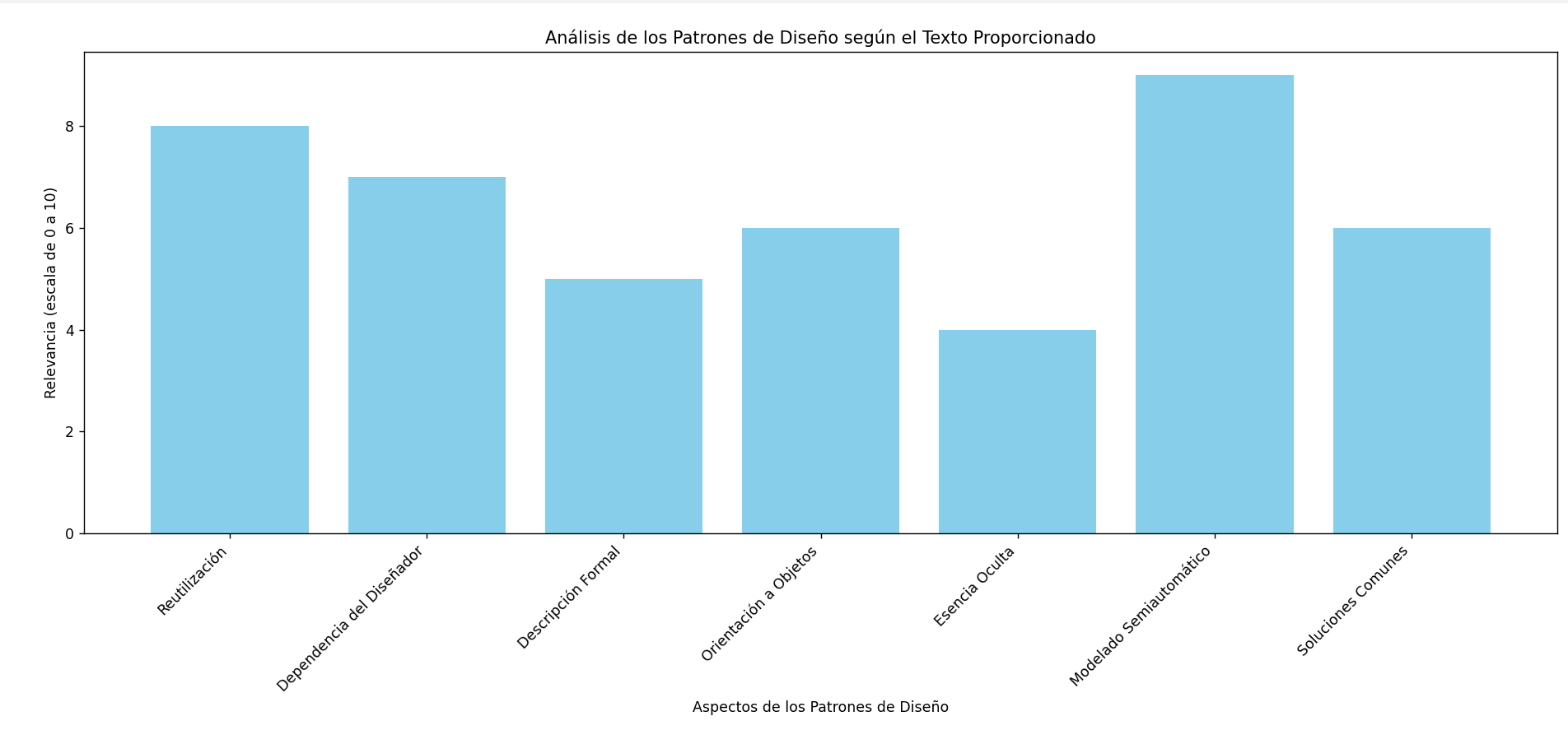
**Resumen**

Los patrones de diseño ofrecen soluciones reutilizables para problemas comunes en el desarrollo de software, pero su eficacia depende del diseñador. A menudo, se describen de manera informal o demasiado orientada a objetos. El artículo propone un enfoque para modelar estos patrones y utilizarlos semiautomáticamente para mejorar el diseño del software, mostrando que este método ayuda a identificar problemas clave en aplicaciones prácticas.

**Reflexión**

La propuesta de un enfoque semiautomático para modelar patrones de diseño resalta la necesidad de mejorar la reutilización en el desarrollo de software. Al abordar la informalidad y la falta de estructuración en la descripción de estos patrones, el artículo evidencia cómo este método puede ayudar a identificar problemas clave y optimizar el diseño, beneficiando así a los desarrolladores en la creación de aplicaciones más efectivas.

[**Enlace Artículo**](https://www.proquest.com/docview/2954642945/abstract/D88747EC066945C7PQ/2?accountid=31491&sourcetype=Scholarly%20Journals)



**Convertir aplicaciones de escritorio en aplicaciones móviles con Java**

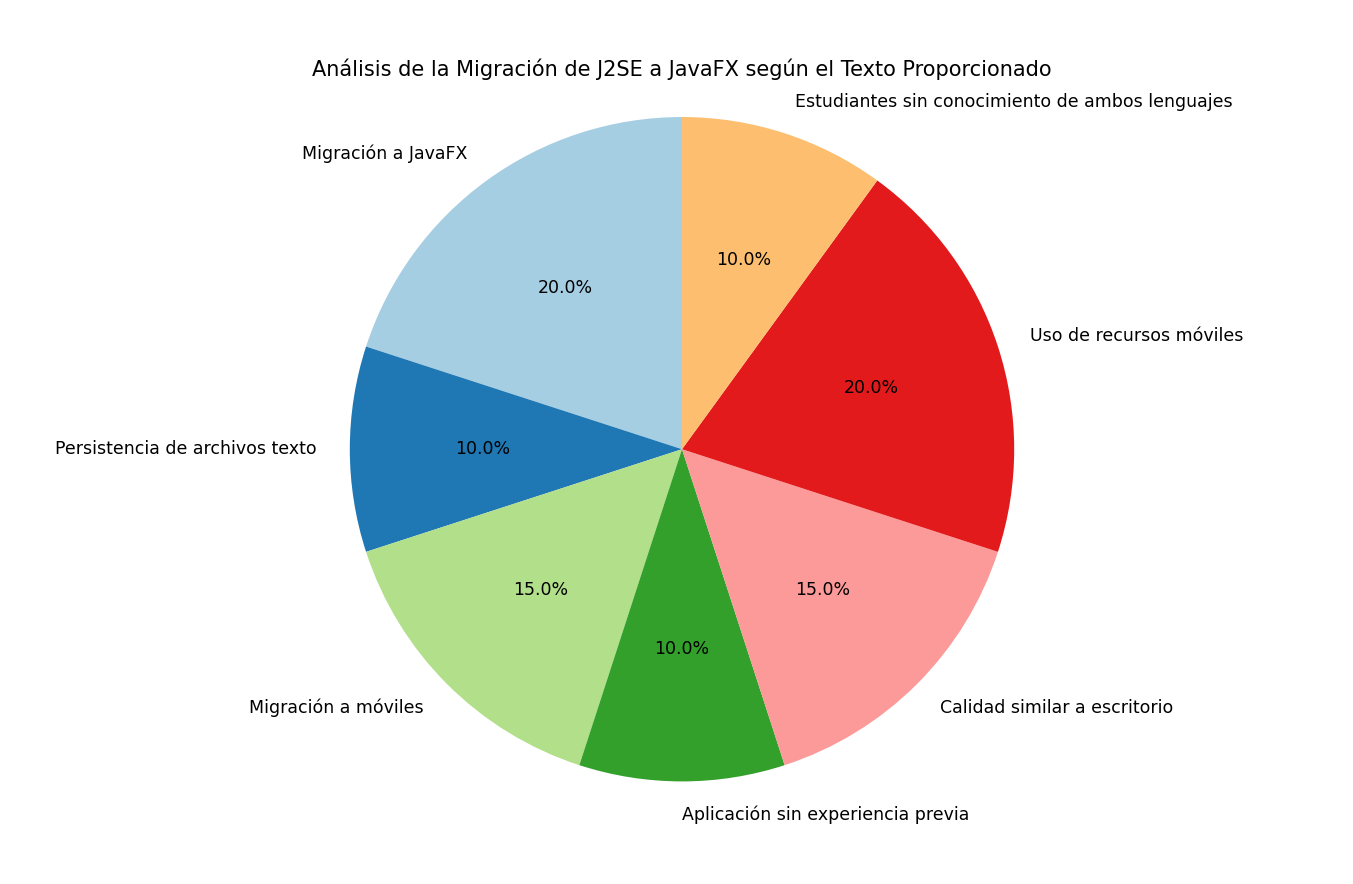
**Resumen**

El artículo presenta un proceso para convertir aplicaciones J2SE a JavaFX, enfocándose en aplicaciones locales que gestionan archivos de texto y se desean migrar a móviles. Proporciona pautas para crear aplicaciones JavaFX desde cero, manteniendo la calidad y funcionalidad de las versiones de escritorio, y se aplicó en un curso avanzado en UNIMINUTO, logrando una conversión exitosa sin necesidad de experiencia previa en JavaFX.

**Reflexión**

El proceso de convertir aplicaciones de escritorio en móviles utilizando JavaFX representa un avance significativo en la accesibilidad del software. Al proporcionar pautas claras y aplicarse con éxito en un curso avanzado, se demuestra que es posible migrar aplicaciones sin experiencia previa, lo que facilita a los desarrolladores expandir sus habilidades y adaptar sus productos a nuevos entornos.

[**Enlace Artículo**](https://www.proquest.com/docview/2018732382/abstract/A0AF1FD47215498BPQ/8?accountid=31491&sourcetype=Scholarly%20Journals)



**Inclusión de patrones de diseño en un plan de estudios de Ingeniería Técnica en Informática de Gestión**

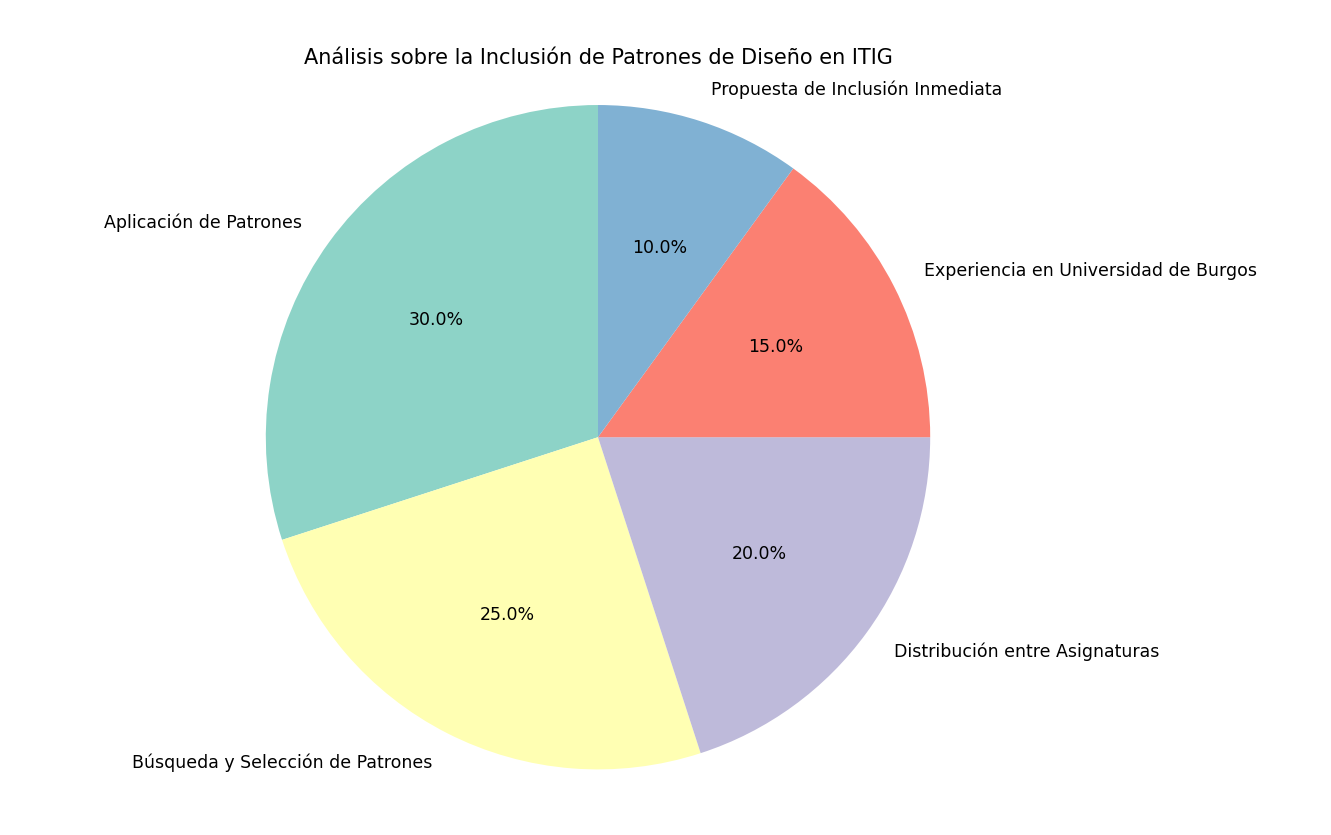
**Resumen**

El trabajo propone incluir patrones de diseño en el currículo de Ingeniería Técnica en Informática de Gestión (ITIG), diferenciando entre la aplicación y la selección de patrones. Se basa en la experiencia previa de la Universidad de Burgos y sugiere una integración adecuada de estos conceptos en las asignaturas del plan de estudios.

**Reflexión**

La inclusión de patrones de diseño en el currículo de Ingeniería Técnica en Informática de Gestión es fundamental para formar profesionales competentes. Al diferenciar entre aplicación y selección de patrones, se propone una educación más estructurada que fomente la comprensión profunda de estos conceptos, preparando a los estudiantes para enfrentar desafíos reales en el desarrollo de software.

[**Enlace Artículo**](https://rua.ua.es/dspace/handle/10045/127856)



**A Systematic Review on Software Architectures for IoT Systems and Future Direction to the Adoption of Microservices Architecture**

**Resumen**

Este estudio revisa la investigación sobre la adopción de la Arquitectura de Microservicios (MSA) en sistemas de Internet de las Cosas (IoT). Analiza 140 artículos para identificar patrones arquitectónicos, desafíos y soluciones en IoT. Destaca trece patrones de MSA, clasifica arquitecturas en nueve categorías, y aborda veintitrés desafíos principales. Los hallazgos ofrecen recomendaciones para mejorar la arquitectura y el desarrollo de software de IoT, beneficiando a la industria y a la comunidad académica.

**Reflexión**

El estudio sobre la adopción de la Arquitectura de Microservicios en sistemas IoT resalta la importancia de identificar patrones arquitectónicos y sus desafíos. Al analizar 140 artículos, se ofrecen recomendaciones valiosas que no solo benefician a la industria, sino que también contribuyen al desarrollo académico, promoviendo una mejor comprensión y aplicación de las arquitecturas en entornos de Internet de las Cosas.

[**Enlace Artículo**](https://www.proquest.com/docview/2933155446/E860FCE7DDB74622PQ/23?accountid=31491&sourcetype=Scholarly%20Journals)

