

Arquitectura de software con programación orientada a objetos

Resumen:

El artículo define como se organiza y estructura un programa para cumplir con sus requisitos técnicos y comerciales. Busca satisfacer necesidades funcionales (tareas) y no funcionales (calidad y desempeño). La programación orientada a objetos facilita la creación y mantenimiento de sistemas complejos a través del uso de clases y objetos. Lenguajes como Java y C++ son destacados en este enfoque, evolucionando y mejorando en el desarrollo de software.

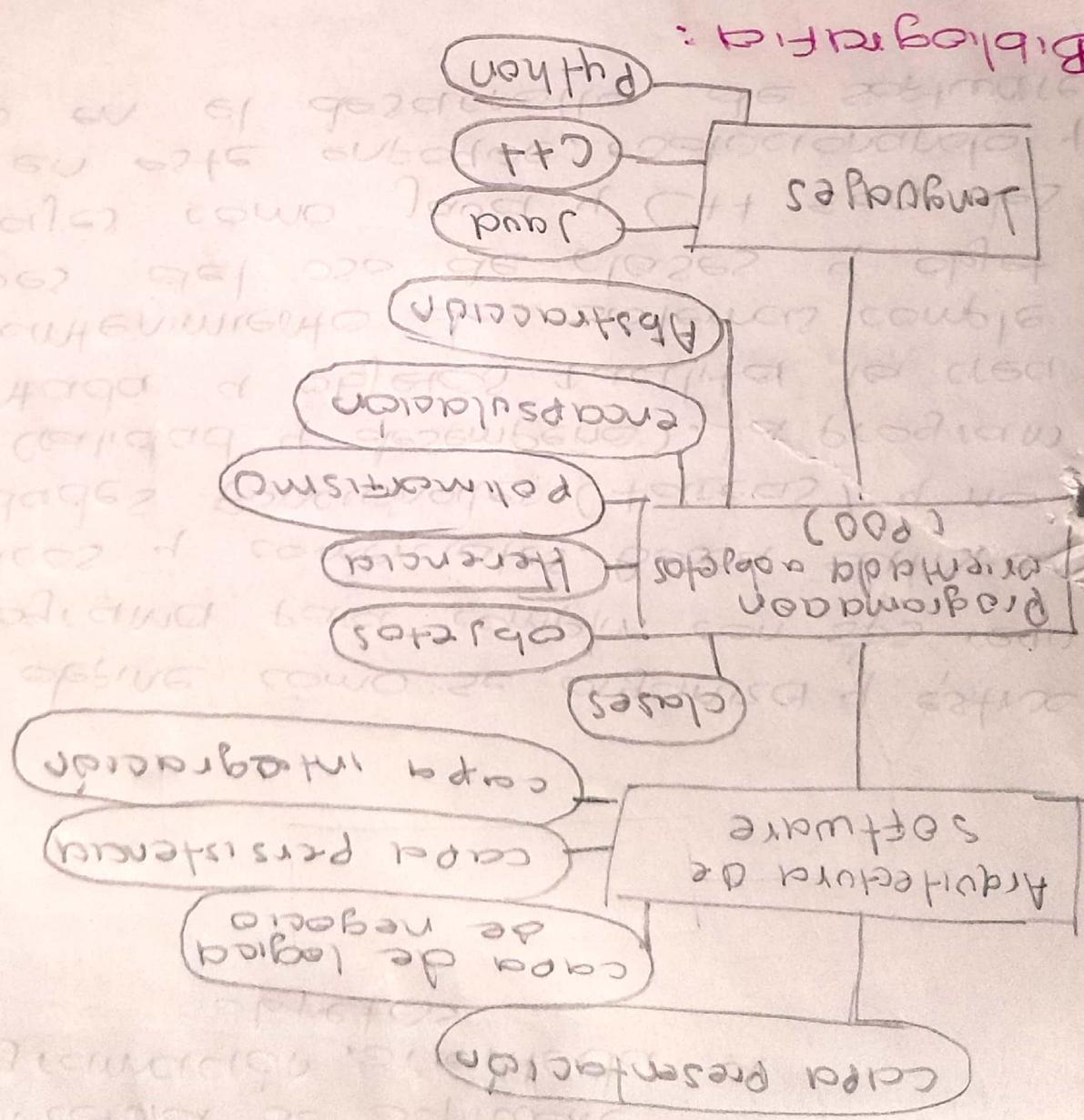
Reflexión:

Es como un mapa que está hecho para seguir en la construcción de los programas y a medida que se está construyendo ir encajando las partes haciendo que este funcione el que tenga clases y objetos ayuda mucho a entender y facilitar el desarrollo del sistema teniendo respaldos claros a la hora de hacer una entrega de cualquier programa.

T. Belénos, H., & Carranqueros, G. (2014) Análisis de asimilación (acumular) de la programación en los estudiantes de la materia Aplicaciones en ambientes propietarios de la materia Aspects (POA) en los estudiantes de la materia Aplicaciones en ambientes propietarios de sistemas.

2. Capítulo Sección, R. (2022). Tema 4. Concepto de arquitectura Software y principios de diseño. Búzco Dígital.

3. Cristián, H. (2021). Una teoría additiva de software. FCEIA UNNE.

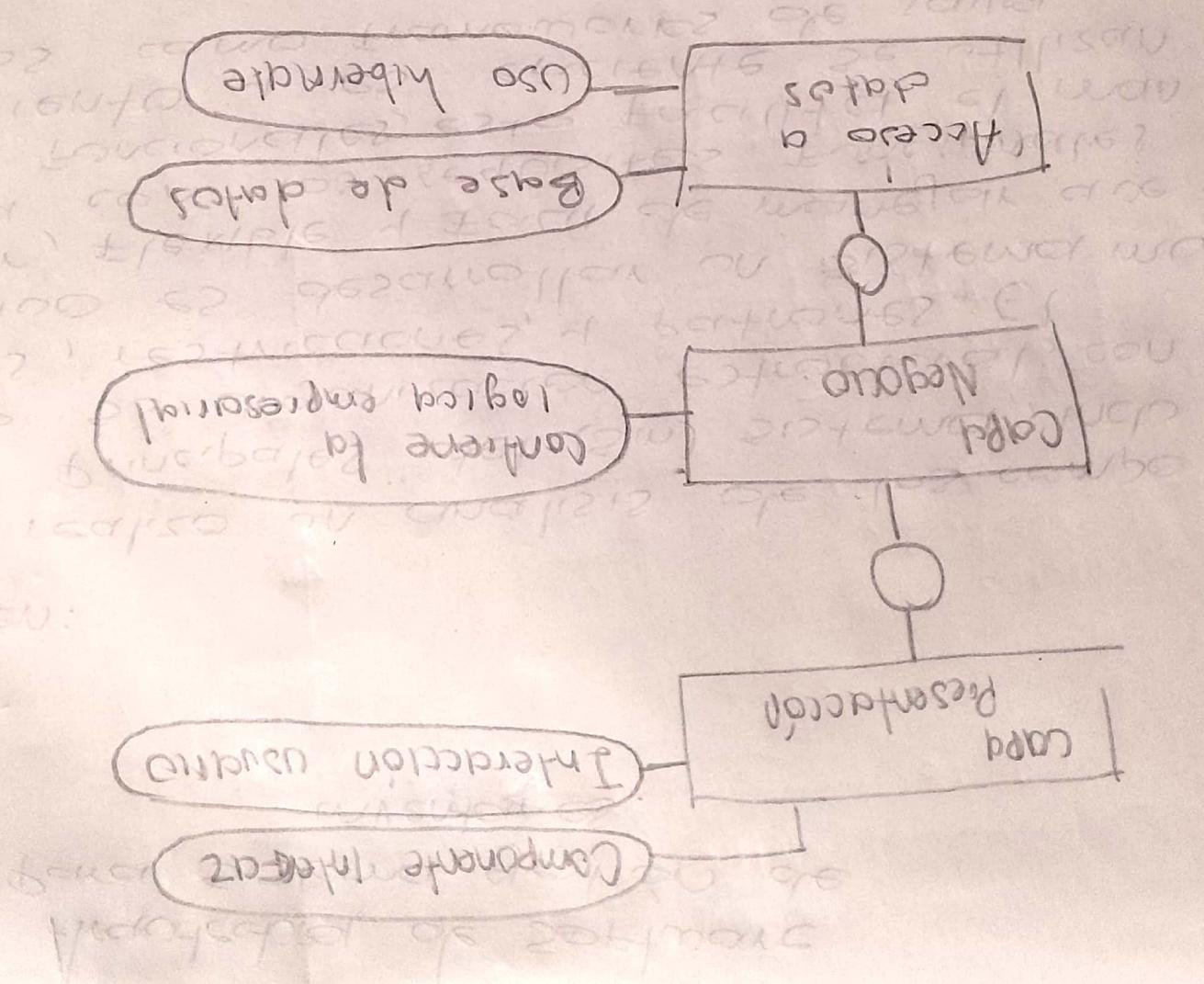


Resumen:

• **Características.**

Para sistema gestión de
Almacenamiento de software

Bibliografias:



Es una idea muy útil para mejorar como las empresas de tecnologías resuelven problemas en su servicio de calidad. Como las empresas de tecnologías tienen que hacer para mejorar sus servicios de calidad. Es necesario tener en cuenta factores como el tiempo de respuesta, la calidad del servicio y la satisfacción del cliente. Para ello, se deben establecer objetivos claros y medibles, así como un plan de acción para alcanzarlos. Los objetivos deben ser realistas y alcanzables, y deben ser revisados periódicamente para evaluar su progreso. La retroalimentación es clave para mejorar continuamente. Una vez establecidos los objetivos, se deben implementar estrategias para lograrlos. Esto implica la identificación de las principales causas de problemas y la implementación de soluciones adecuadas. La retroalimentación es fundamental para evaluar la efectividad de las estrategias y ajustarlas si es necesario. La mejora continua es un proceso constante que requiere compromiso y dedicación por parte de todos los miembros de la organización.

Definition:

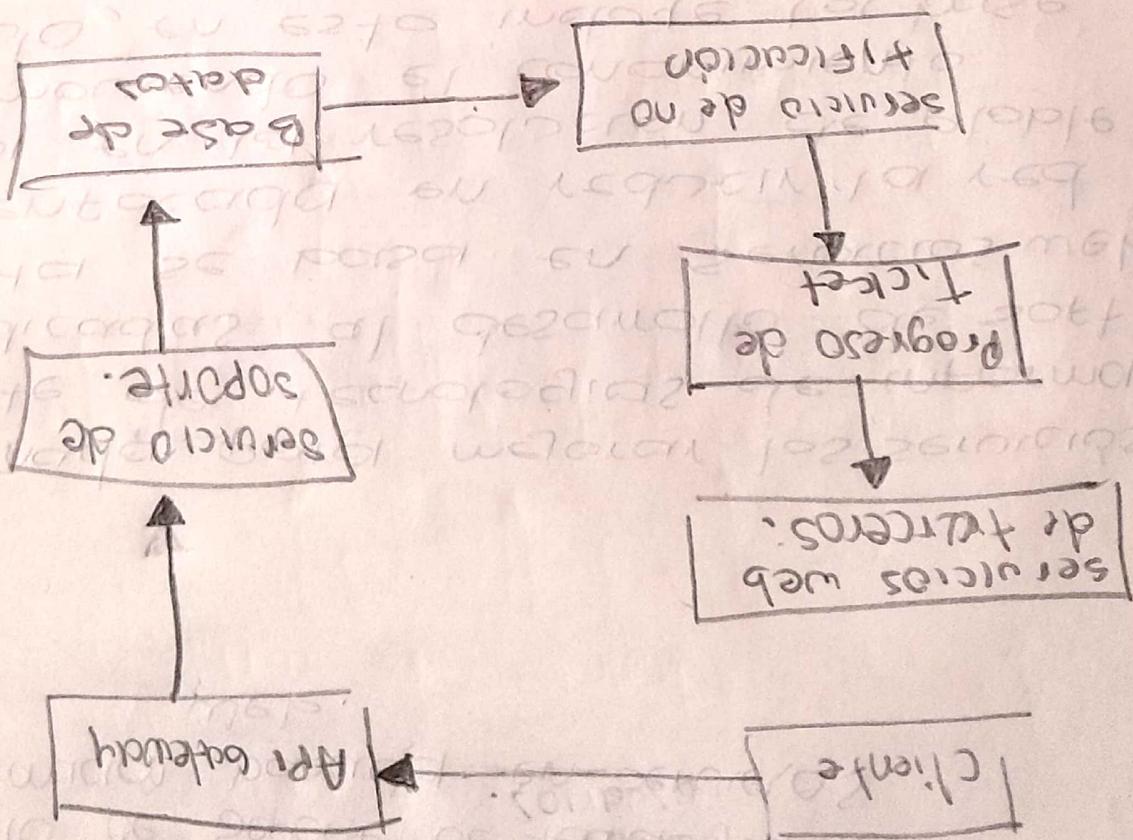
Estos diseños para mejorar los servicios de soporte de tecnologías de información
cifran dedicados al desarrollo de soft
ware. Están basados en servidores web
y están enfocados en reducir la red
undanicia en la resolución de problemas,
promoviendo el conocimiento
comunitario, en este caso de base
de datos, que incluye, como
una colección de información
orientada a ser reutilizada.

Desumeau

Alquiler de software para el servicio de soporte de tecnología de información basada en servicios web.

1. Bell, A. (2008) Service-oriented modeling - John Wiley & Sons, Inc.
2. Erl, T. (2009) SOA Principles of Service Design. Prentice Hall.
3. Jassoth, M. (2009) SOA in practice. The art of distributed system design.
4. Pulles, E. (2006) Understanding Enterprise SOA.

Bibliografía:



Reflexión: Nos (el) da a entender que es tener una buena arquitectura de software, especializada en la implementación de sistemas que permiten manejar el inventario de la empresa, así como proveer información para que el usuario pueda acceder a través de su terminal móvil.

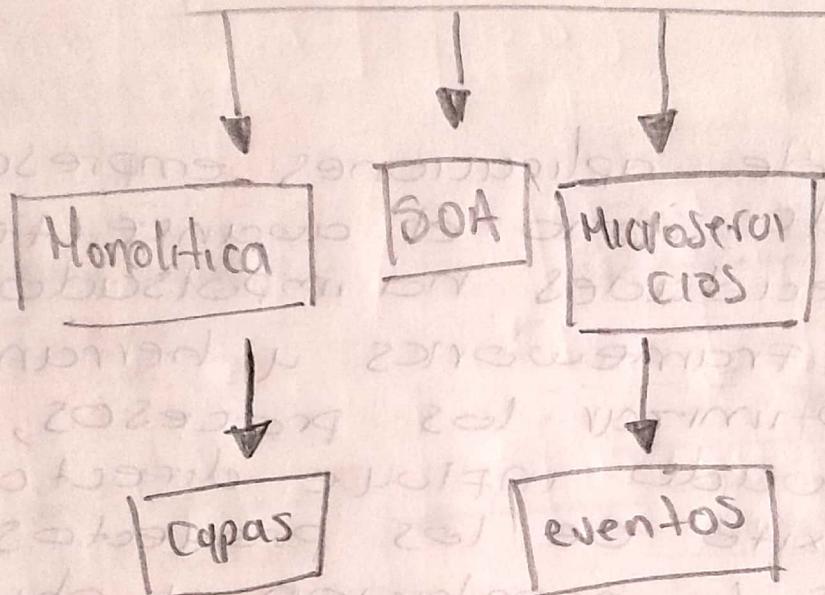
בְּנֵי יִשְׂרָאֵל

El desarrollo de aplicaciones empresariales, desatendiendo el consumo de los usuarios y necesidades logísticas, tiene como resultado la implementación de sistemas que no cumplen con las expectativas de los usuarios. Los sistemas que se crean para optimizar los procesos, su diseño adecuado incluye directamente en el éxito de los proyectos empresariales (a través de diferentes transformaciones (los métodos de desarrollo de software que se subraya) la importancia de la creación de lenguajes y patrones en la creación.

Desmen;

Documentación y análisis de los patrón palés frameworks de arquitectura de software en aplicaciones embebidas

Arquitectura Empresarial.



Bibliografía:

1. (Informatika) W.-A. (21 de 06 de 2014) obtenido de http://es.wikipedia.org/wiki/Arquitectura_en_pipeline (INFORMATICA)
2. Blass Len, C.P. (1998). Software Architecture in Practice.
3. Miller, J. M. (2001). Model Driven Architecture. Technology Committee and Architecture Board.
4. Guglielmetti, M. (s.f.). <http://www.mastermagazine.info/termino13916.php>.

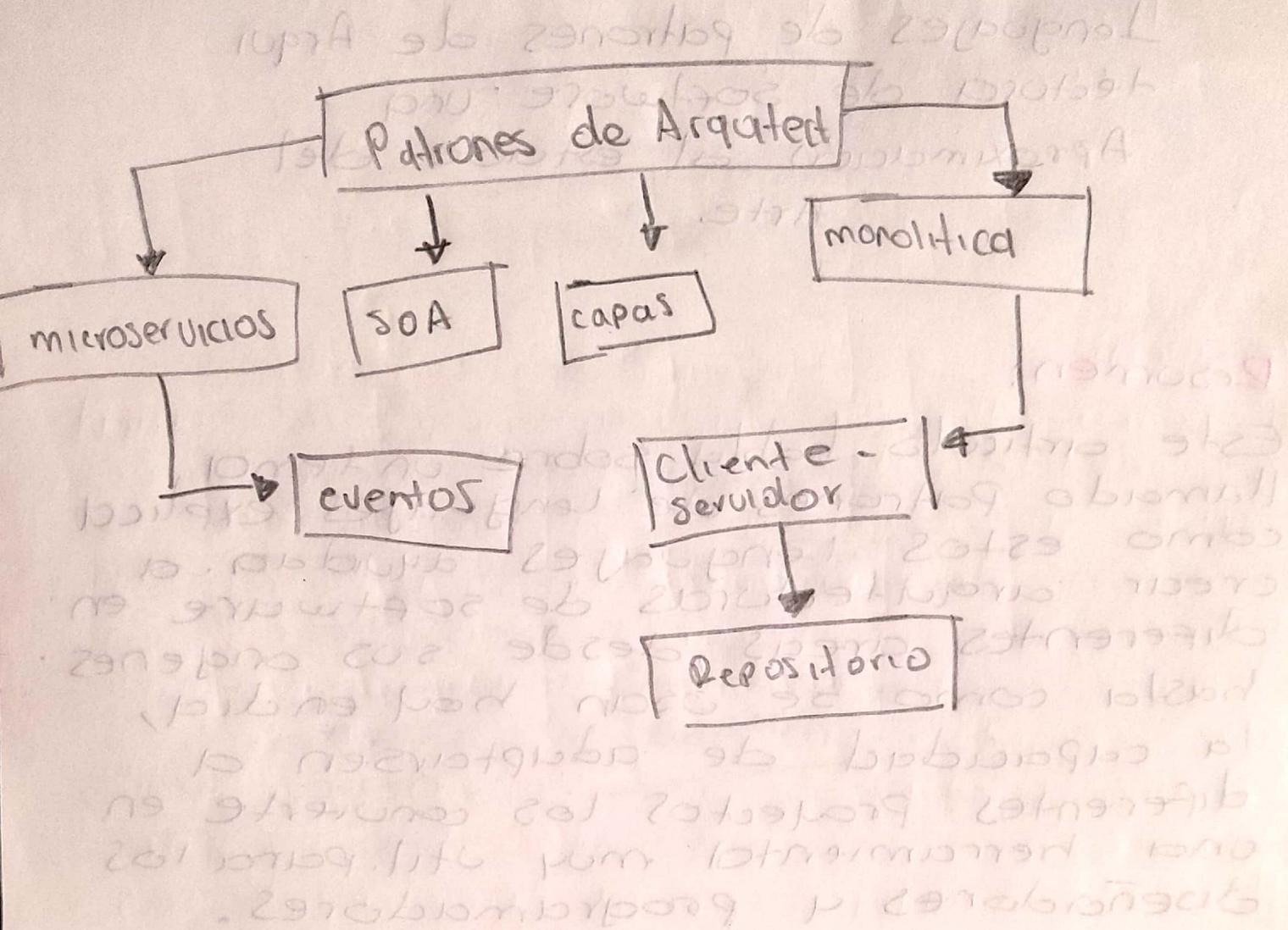
Lenguajes de patrones de Arquitectura de software: Una Aproximación al estudio del Arte.

Resumen:

Este artículo habla sobre un tema llamado Patrones de lenguajes. Explica como estos lenguajes ayudan a crear arquitecturas de software en diferentes áreas, desde sus orígenes hasta como se usan hoy en día, la capacidad de adaptarse a diferentes proyectos los convierte en una herramienta muy útil para los diseñadores y programadores.

Reflexión:

Ha evolucionado mucho desde que empecé hasta el día de hoy los lenguajes de patrones son unas herramientas clave para mejorar la forma en que diseñamos y hacemos software. Ayudan a los programadores a lidiar con los problemas y desafíos que surgen cuando se crean sistemas que son extremadamente extensos y pueden aplicarse a diferentes tipos de sistema.



Bibliografica:

1. Fairbank, George: Just enough Software Architecture: A risk Driven Approach. Boulder: Marshall & Brainerd, 2010 15 P.
2. ALEXANDER, Christopher: Notes on the Synthesis of Form. 6. (1964).
3. LAZARO, María y Marcos, Esperanza, Op. cit. p. 6.
4. DEARDEN, A. M. and FINLAY, J: Op. cit. p. 5.

3. Daniel Le Metayer, Software Architecture Styles as Graph Grammars. Lrisa / Inria, 1996.

November 1996.

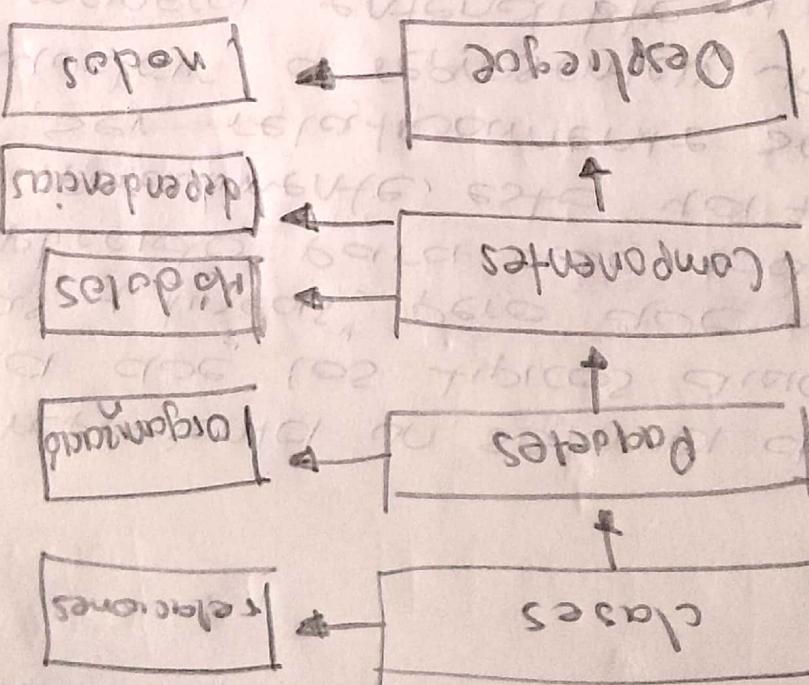
2. Mavraten Boasson, The Artistry of Software Architecture. IEEE Software. Vol. 12, No. 6

discipline. Prentice Hall. 1996.

Architecture Perspectives on an Emerging In

1. Harry Shaw y David Garlan, Software

Bibliografía:



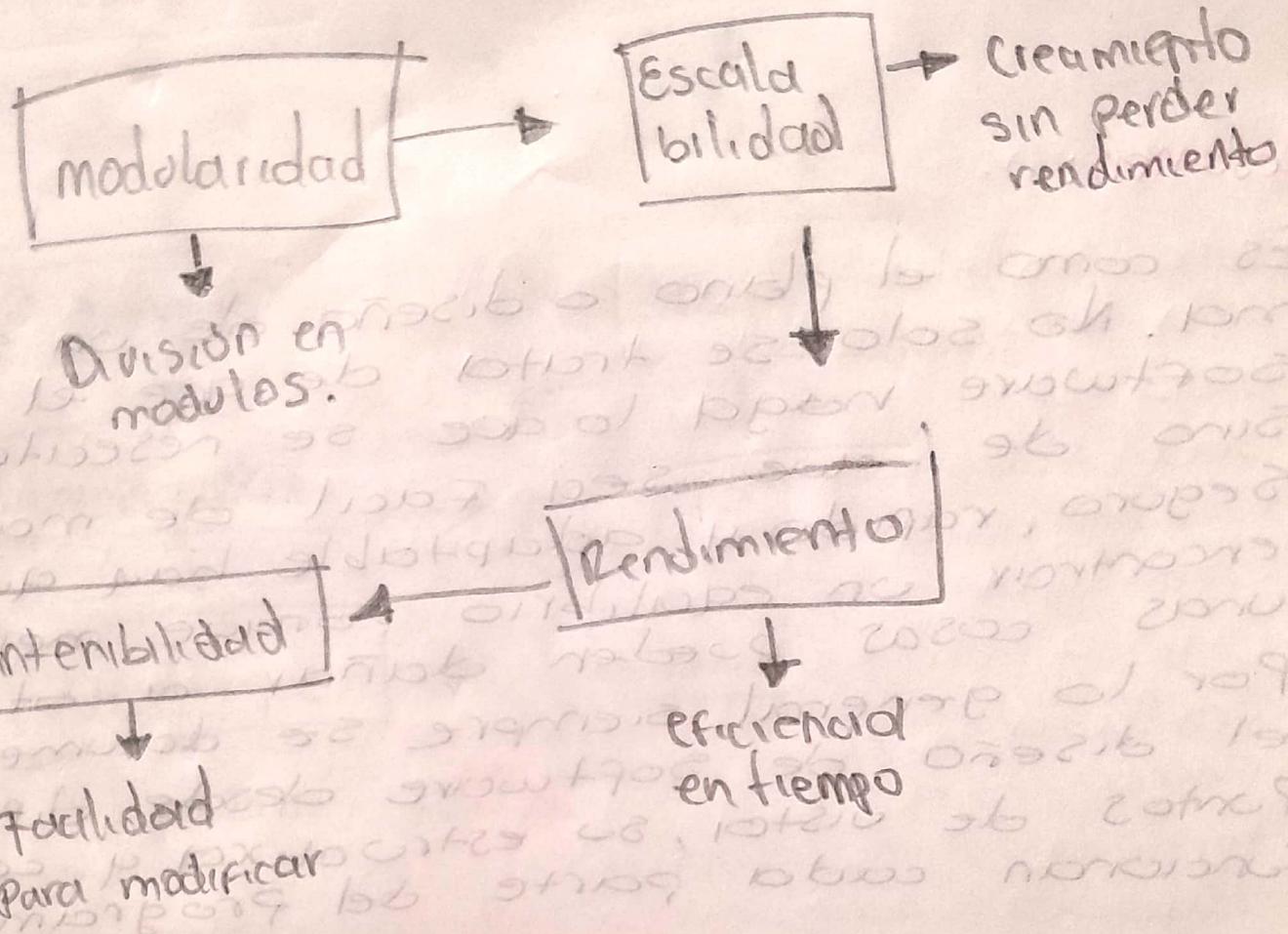
muchas variables y permitir bien.
cambiar el ambiente tener en cuenta
para que el software funcione corriente
una buena arquitectura es fundamental
fíjese durante el proceso, el punto es que
proceso de diseño y requisitos que se
es muy importante documentar todo el
es un trabajo de equilibrio constante
y opiniones de cada integrante de la equipo
algo fíjese y todo depende de decisiones
crear la estructura de un software no es

Definición:

funciona cada parte del programa.
función de usar, su estructura y como
el diseño de software desde diferentes
por lo general siempre se documenta
unas cosas pueden darse a veces.
encuentrar un equilibrio porque mejorar
seguro, rápido y adaptable hoy que
sino de que sea fácil de manejar
software nega lo que se necesita
más. No solo se trata de que el
es como el piano o diseño de un sistema

Resumen:

Algoritmos de calidad y arquitectura
del software.



Bibliografías:

1. Len Bass, Paul Clements, and Rick Kazman. Software architecture in practice. SEI series in software Engineering. Addison-Wesley, 2 edition, 2003.
2. Frank Buschmann, Len Bass, David Kühn, James Meister, Reed Little, Oriented Software Architecture: A system OF Patterns. John Wiley & Sons Ltd... August 1996.

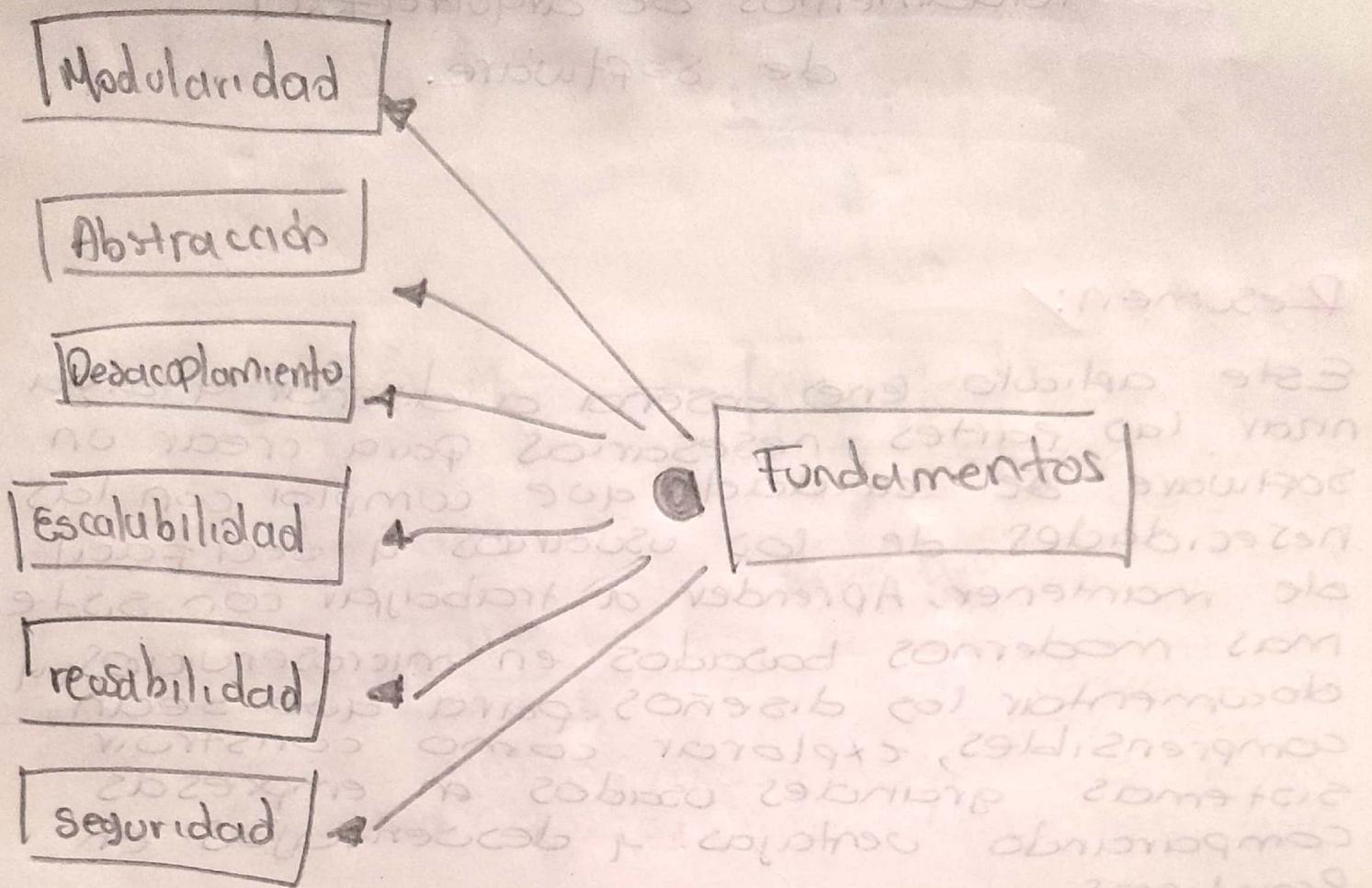
Fundamentos de arquitectura de software.

Resumen:

Este artículo (en) enseña a diseñar y organizar las partes necesarias para crear un software de calidad que cumple con las necesidades de los usuarios y sea fácil de mantener. Aprender a trabajar con sistemas modernos basados en microservicios y documentar los diseños para que sean comprensibles, explorar como construir sistemas grandes usados en empresas comparando ventajas y desventajas en prácticas.

Reflexión:

Muestra que para crear un buen software nos solo se necesita saber programar, sino también pensar en como organizar y conectar, aprender a usar nuevos métodos y herramientas que nos ayude a facilitar el aprendizaje para hacer sistemas que sean útiles y fáciles y que cumplan con las necesidades de las personas que los van a usar, también nos enseña a estar actualizados constantemente a nuevas tecnologías que sean de nuestro apoyo hacia el aprendizaje.



Bibliografías:

1. Vernon, V. (2016) Domain-Driven Design Distilled 11e (1st edition), Addison-Wesley Professional
2. Christudas, Binildas (2019) Practical microservices Architectural Patterns. Apress I.P.
3. Davis, C. (2019) Cloud Native Patterns. Manning, Publications.
4. Soh, Julian, Copeland, Marshall, Puca, Antony, Harris (2010) Microsoft Azure. Apress I.P.

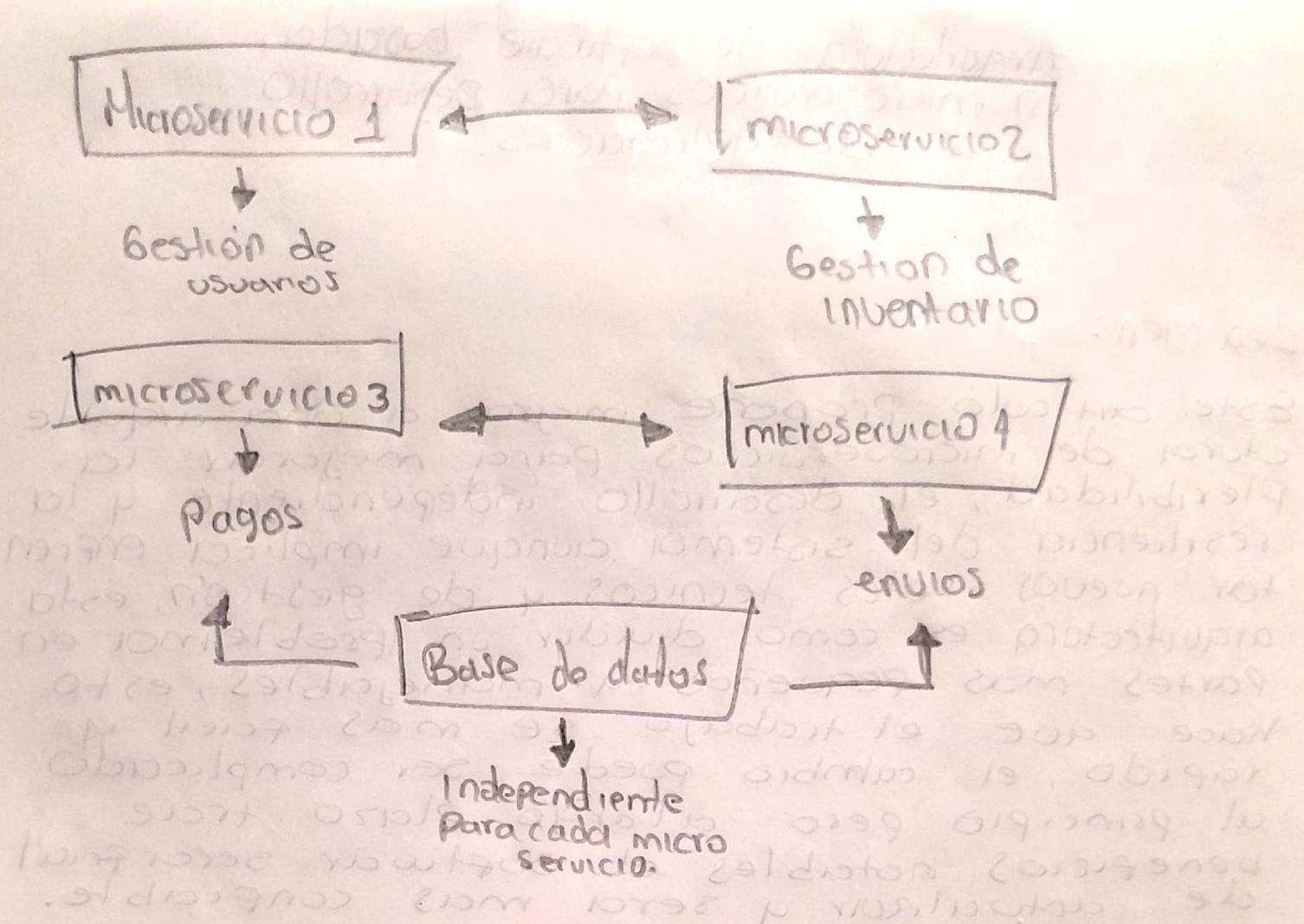
Arquitectura de software basada en microservicios para Desarrollo de aplicaciones.

Resumen:

Este artículo propone migrar a una arquitectura de microservicios para mejorar la flexibilidad, el desarrollo independiente y la resiliencia del sistema aunque implica enfrentar nuevos retos técnicos y de gestión, esta arquitectura es como dividir un problema en partes más pequeñas y manejables, esto hace que el trabajo sea más fácil y rápido, el cambio puede ser complicado al principio pero al largo plazo trae beneficios notables el software será fácil de actualizar y será más confiable.

Reflexión:

Me parece que la evolución de esta arquitectura como lo es microservicios se ve la necesidad de adaptarse a un entorno de una nueva tecnología, las prácticas continuas son las que se investigan y se aprende con facilidad y eficiencia buena comunicación esto ofrece una visión estratégica y ofrecer soluciones asegurando la continuidad y calidad del servicio en organizaciones modernas.



Bibliografía:

1. Carneiro, C.; Schmelmer, T.: *Microservices from Day one*. Apress. Berkeley. CA, (2016).
2. Fowler, M.; Lewis, J., *Microservices*, Vittattu, (2014)
3. Newman, S. *Building Microservices*. O'Reilly media inc, (2015).
4. Richards, M., Garcés, O., Castro, H., Verano, Salamanca, I., Casallas, R. y Gil, S.