Biografía Patricia del Rocio Sarmiento Ruiz

Pantalla de video juego de una persona

Descripción generada automáticamente con confianza media  
Mi nombre completo es Patricia del Rocio Sarmiento Ruiz, tengo 18 años, cumplo el 3 de abril, me encuentro actualmente cursando un tecnólogo en Análisis y Desarrollo de Software , en lo que a mí respecta, siento que no tengo las habilidades o la lógica suficiente para continuar estudiando software, por lo cual me eh decidido a estudiar en la universidad algo que siempre quise estudiar, que de momento es innecesario comentar, el desarrollo de software me parece muy útil , ya que actualmente la tecnología es lo que casi domina a la humanidad , y saber manejar esa tecnología, es casi manejar al ser humano en sí, que suele ser ignorante, o es lo que me han comentado bastante , ya que mientras las personas se la pasan horas y horas en redes sociales y en aplicaciones o páginas web, los creadores de esas páginas y aplicativos , les muestran la información que quieren que vean para tenerlos entretenidos y ocupados en cosas sin importancia y que no dan productividad, pero algo que yo si digo es, que el desarrollo de software no es para todos, ya que siempre va a haber personas que se desarrollaran en otros ámbitos diferentes , aunque no igual de productivos.

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

Resumen

El artículo resalta cómo la Arquitectura de Software es esencial en el desarrollo, definiendo la estructura general de un sistema y mostrando cómo interactúan sus componentes. Según el Instituto de Ingeniería de Software, es como un mapa que muestra las partes y sus conexiones sin meterse en detalles. La investigación encontró que los elementos más importantes en estas arquitecturas son los componentes y los Conectores, que se repiten en muchos modelos y lenguajes. También se descubrió que las "vistas arquitectónicas" son la herramienta clave para ayudar a los equipos a entenderse mejor, aunque tienen la desventaja de necesitar un lenguaje formal para automatizar procesos, lo que complica reutilizar diseños. Estos hallazgos no se quedarán solo en la teoría; se usarán para proponer una manera de representar el conocimiento arquitectónico y documentar las decisiones tomadas, superando las limitaciones actuales de las vistas arquitectónicas.  
Reflexión: La arquitectura de software es crucial para estructurar sistemas complejos. Incorporar vistas arquitectónicas fomenta la comunicación entre equipos, aunque su implementación requiere lenguajes formales que pueden complicar la automatización. Superar estas limitaciones permitirá documentar decisiones de diseño con mayor precisión.

Imagen en blanco y negro

Descripción generada automáticamente con confianza media

Bibliografía Miguel Angel Sánchez Palmero, Nemury Silega Martínez , Olga Yarisbel Rojas Grass, 26/10/2018

# Lenguajes de Patrones de Arquitectura de Software: Una Aproximación Al Estado del Arte

Resumen   
En el artículo "Lenguajes de Patrones de Arquitectura de Software: Una Aproximación al Estado del Arte" leemos cómo los lenguajes de patrones han cambiado la forma en que se piensa la arquitectura de software. Desde los inicios de la ingeniería de software en los años 1950 con la programación estructurada y la orientación a objetos hasta ahora... contribuyó a crear la base de los patrones de diseño que usamos hoy. Inspirados por el arquitecto Christopher Alexander (que habló de “patrones” en la arquitectura de edificios) estos conceptos se adaptaron al mundo del software para encontrar soluciones que se pudieran repetir. El artículo explica cómo estos lenguajes de patrones ayudan a que los diseñadores y desarrolladores resuelvan problemas complejos de una manera más clara y sencilla. El artículo nos muestra ejemplos prácticos, como su uso en la seguridad de la información y en el diseño de sistemas e-Business lo cual nos demuestra lo útiles que son. Finalizando, se concluye que los lenguajes de patrones son fundamentales, o esenciales para que el desarrollo de software sea más rápido y con mejor calidad, facilitando la solución de problemas arquitectónicos de forma más simple y efectiva.  
Reflexión: Los lenguajes de patrones son herramientas esenciales para resolver problemas recurrentes en diseño de software. Su influencia histórica demuestra cómo estas soluciones sistemáticas mejoran la calidad y rapidez del desarrollo, particularmente en áreas como seguridad y e-Business.

Imagen que contiene texto, pizarrón

Descripción generada automáticamente  
Bibliografía, Victor Hugo Jimenez-Torres , Wilman Tello-Borja , Jorge Iván Rios-Patiño, 4/12/2014

# Análisis comparativo de Patrones de Diseño de Software

Resumen  
 El documento compara cinco patrones de diseño de software (Template Method, MVC, MVP, Front Controller y MVVM) y explica cómo ayudan a organizar y mantener el código, evitando duplicaciones. No hay un “mejor” patrón; cada uno tiene sus pros y contras. Por ejemplo, Template Method es ideal para reutilizar algoritmos con pequeñas variaciones. MVC es popular por separar bien la lógica, la vista y el modelo (aunque puede ser un poco difícil al principio). MVP separa la interfaz y la lógica (genial para apps grandes) pero es más complejo. Front Controller gestiona todas las solicitudes desde un solo punto, mejorando así la seguridad, aunque escalar la app puede llegar a ser más complicado. MVVM es excelente para mantener la modularidad y hacer pruebas, pero puede llegar a ser difícil para principiantes.

Reflexión  
Mi conclusión es que existen varios patrones de diseño que nos pueden ayudar a organizar el código de nuestro proyecto, pero al final, debemos analizar bien, cuáles son las necesidades de nuestro programa o proyecto, pasa así mismo poder elegir un patrón para usar.

Diagrama

Descripción generada automáticamente

Bibliografía, Gavilánez Álvarez Oscar Danilo, Layedra Natalia, Ramos Vinicio, 28/07/2022

# Marco de Trabajo para Seleccionar un Patrón Arquitectónico en el Desarrollo de Software

La tesis aborda los desafíos de seguridad en arquitecturas de microservicios, proponiendo el patrón Microservice Security Pattern API Gateway (MSPAG), que utiliza JSON Web Tokens (JWT) y el algoritmo H256 para centralizar la autenticación y proteger los endpoints a través de un API Gateway. Incluye un marco teórico, análisis del estado del arte e implementación práctica en lenguajes como C#, Python y Java, evaluando su eficacia mediante pruebas. Concluye que la adaptación de patrones de seguridad como MSPAG fortalece la resistencia de los sistemas basados en microservicios, garantizando integridad y confidencialidad.

Reflexión: Seleccionar patrones arquitectónicos adecuados es clave para optimizar tiempos, costos y calidad. Proveer herramientas prácticas y bien estructuradas facilita establecer bases sólidas para proyectos desde su inicio.

Diagrama

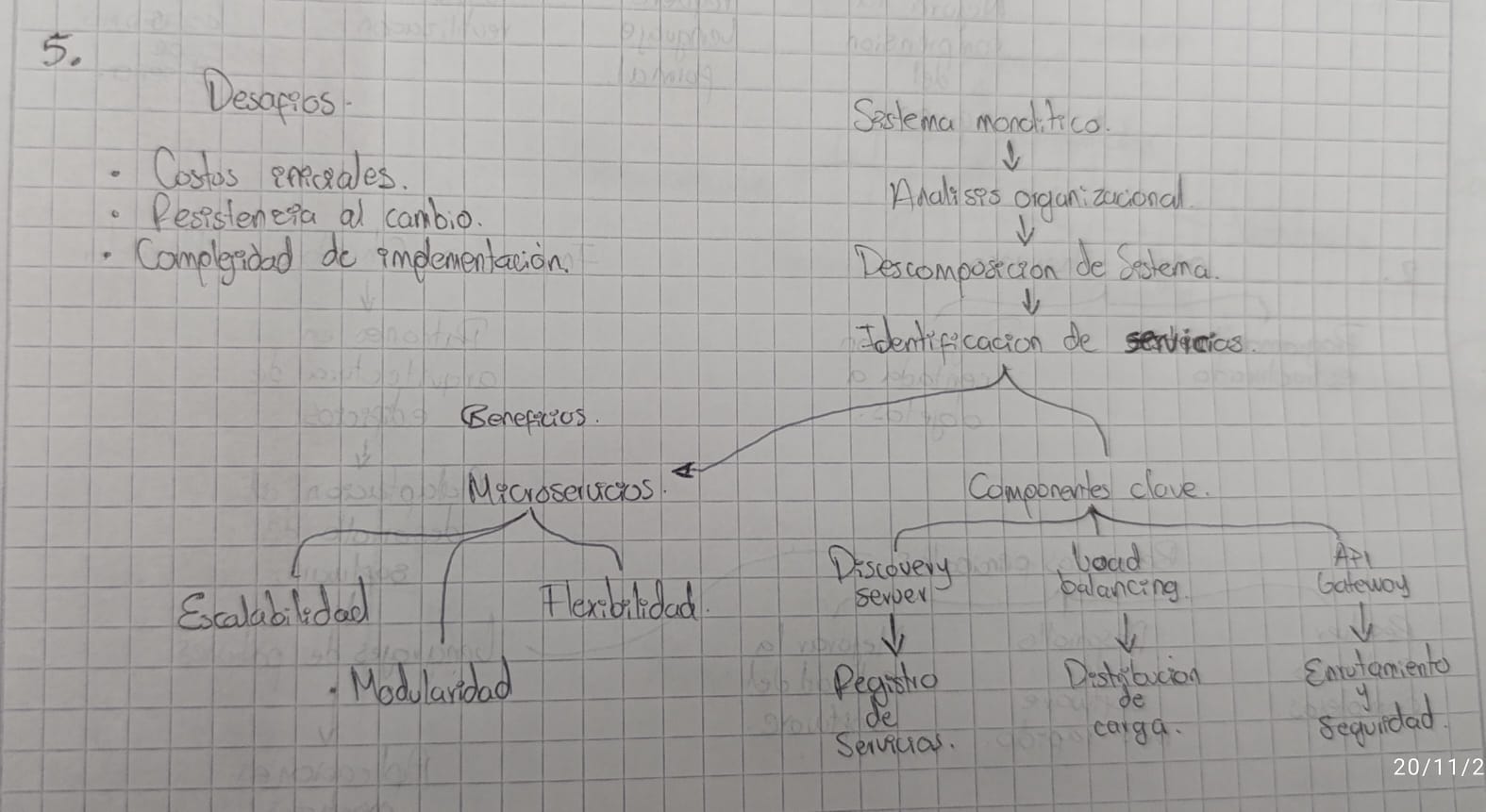
Descripción generada automáticamente

Bibliografía: Hernandez Guzman, Karen Monica; 2023-10-27

# Propuesta metodológica para migración de sistemas web con arquitectura monolítica hacia una arquitectura basada en microservicios.

El documento propone una metodología para migrar sistemas web de arquitectura monolítica a microservicios, destacando beneficios como escalabilidad y modularidad, pero subrayando la necesidad de una descomposición cuidadosa. Introduce el esquema SMMicro, que aborda análisis organizacional, descomposición de sistemas, y la construcción de un ecosistema de microservicios con componentes como Discovery Server y Load Balancing. La metodología es validada en un caso de estudio en la Escuela Politécnica Nacional, evidenciando mejoras en flexibilidad y escalabilidad. Concluye que los microservicios son ideales para sistemas grandes y dinámicos, pero requieren un enfoque estratégico para superar desafíos como costos iniciales y resistencia al cambio.

Reflexión: Migrar de arquitecturas monolíticas a microservicios aporta escalabilidad y modularidad, pero requiere planificación estratégica. Este enfoque es ideal para sistemas grandes, aunque implica costos iniciales y resistencia al cambio.



Bibliografía: Arboleda Cola, Carlos Augusto; 11-dic-2017

# ¿Son los microservicios la mejor opción? Una evaluación de su eficacia y eficiencia frente a los monolitos

El documento compara la eficacia y eficiencia de arquitecturas monolíticas y de microservicios, destacando que los monolitos ofrecen mejor rendimiento en contextos simples debido a su integración directa, mientras que los microservicios destacan en modularidad, escalabilidad y despliegue independiente, aunque implican mayor complejidad y consumo de recursos. Utilizando herramientas como Docker y JMeter, las pruebas mostraron que los monolitos tienen tiempos de respuesta más rápidos y menor tasa de error, mientras que los microservicios son más adecuados para sistemas grandes y dinámicos, especialmente con un API Gateway para optimizar su desempeño. Se concluye que la elección de la arquitectura depende de las necesidades y el contexto del sistema.

Reflexión: La elección depende del contexto. Los monolitos ofrecen simplicidad y eficiencia para sistemas pequeños, mientras que los microservicios son mejores para proyectos grandes y dinámicos, priorizando la escalabilidad y flexibilidad.

Diagrama

Descripción generada automáticamente

Bibliografía: Rodrigo Alejandro Gonzalez; Sergio Agustín Giménez; Numa Roy Molina Pualuk; Rodrigo Zalazar; 2024-09-02

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

El documento trata sobre la optimización de consultas SQL en bases de datos relacionales. Se discute cómo consultas mal codificadas pueden consumir excesivos recursos de procesamiento, afectando el desempeño de las aplicaciones. Se destaca la importancia de implementar herramientas de optimización, como PL/SQL Developer, que analizan y mejoran el rendimiento de las consultas mediante planes de ejecución.

Se describen las fases del procesamiento SQL: análisis, optimización, generación de registros y ejecución. La optimización puede ser automática o manual y requiere conocimientos sólidos de la arquitectura de la base de datos y de SQL. Un ejemplo práctico con tablas de datos demuestra cómo mejorar la eficiencia al implementar índices y ajustar las consultas.

El documento concluye que la optimización debe ser constante, y se mencionan técnicas como evitar cláusulas complejas y usar consultas simples. Se enfatiza que las herramientas gráficas ayudan a identificar visualmente mejoras posibles en las consultas.

Reflexión: Optimizar consultas SQL es vital para el rendimiento de bases de datos en la nube. Implementar herramientas adecuadas y técnicas de mejora constante garantiza aplicaciones eficientes y escalables.

Imagen en blanco y negro

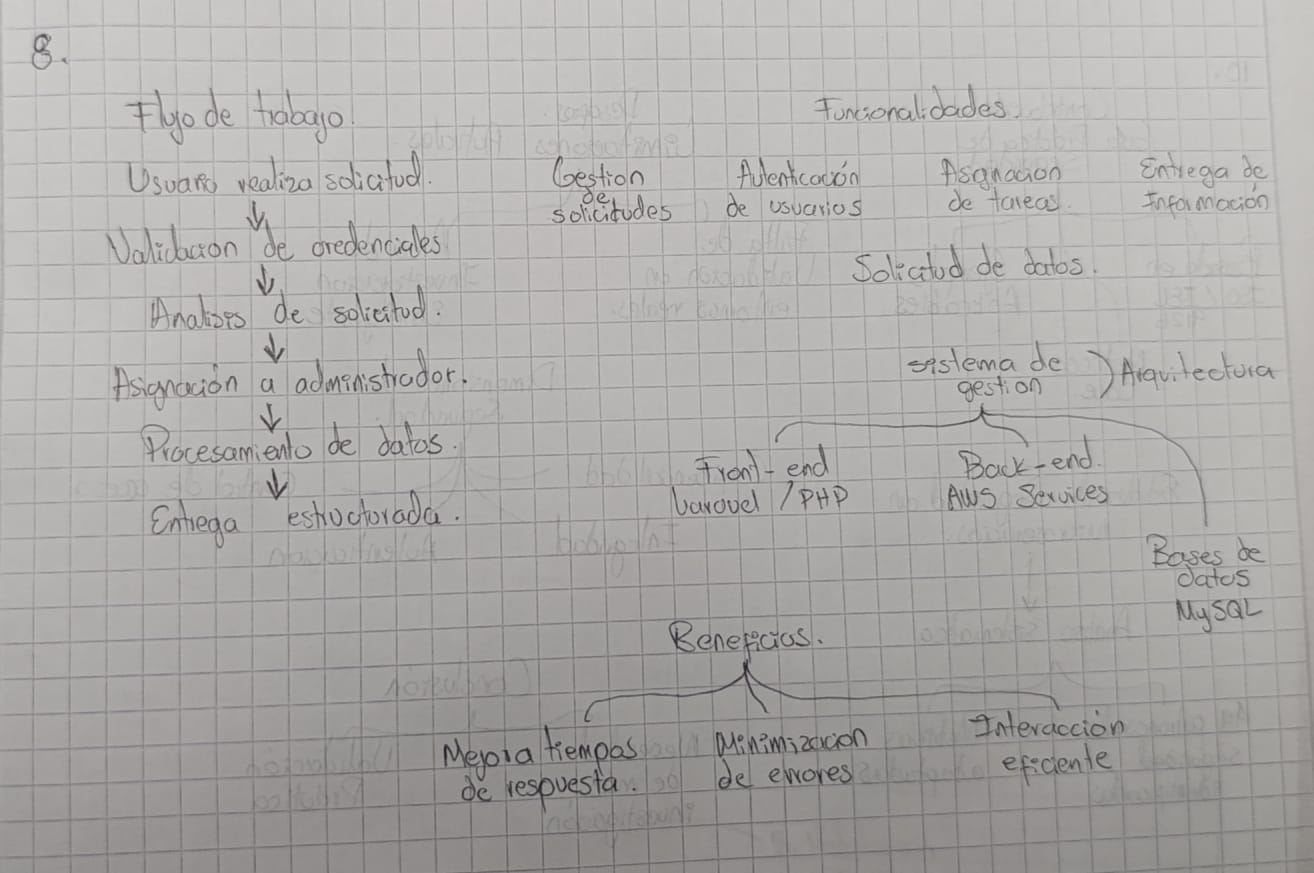
Descripción generada automáticamente con confianza media

Bibliografía: Manzanilla Vidal, Wilbert; Feliciano Morales, Severino; Alvarez Hilario, Valentin; Solis Carmona, Edgardo; Molina, Felix; 2018-11

# Solución para gestión y control de información para las bases de datos

El documento aborda una solución para la gestión y control de información en bases de datos, destacando su importancia en organizaciones modernas debido al crecimiento exponencial de datos impulsado por tecnologías como la inteligencia artificial, el internet de las cosas y la digitalización. La propuesta se enfoca en la implementación de un sistema de gestión de solicitudes de datos, que permite capturar, analizar y entregar datos de manera estructurada y segura, optimizando la administración y evitando la saturación de los sistemas de bases de datos.

Se describen métodos de desarrollo basados en ingeniería de software y la arquitectura de AWS para implementar servicios escalables y seguros. La herramienta diseñada utiliza tecnologías como PHP con Laravel y bases de datos MySQL. Entre las funcionalidades, destacan la gestión de solicitudes, asignación de tareas, autenticación de usuarios y la entrega de información eficiente.

El trabajo concluye que la solución facilita la interacción entre usuarios y administradores de bases de datos, mejora los tiempos de respuesta y minimiza errores. También se sugiere continuar con pruebas de funcionamiento y evaluar futuros desarrollos para robustecer el sistema y ampliar sus capacidades.  
Reflexión : Sistemas robustos para gestionar datos en entornos modernos son esenciales. La implementación de metodologías adecuadas mejora la interacción entre usuarios y administradores, optimizando recursos y reduciendo errores.

Bibliografía: Fuentes Pardo, Yudy Camila; 2024

# Problemas que afectan a la Calidad de Software en Entrega Continua y Pruebas Continuas

El documento examina los problemas que afectan la calidad del software en \*\*entrega continua (CD)\*\* y \*\*pruebas continuas (CT)\*\* en entornos ágiles, como pruebas largas, resultados inconsistentes, y desafíos con interfaces dinámicas, Big Data y dispositivos móviles. Destaca que las CT son esenciales para detectar defectos críticos rápidamente, aunque persisten retos como la cobertura limitada de pruebas no funcionales y en la nube. Se concluye que, pese a avances, se necesita investigación adicional para integrar eficientemente las pruebas continuas en los procesos de CD, asegurando calidad y confiabilidad.

Reflexión: La integración de pruebas continuas en entrega continua asegura calidad, pero enfrenta retos significativos. Superarlos exige más investigación y técnicas avanzadas para mejorar la confiabilidad.

Imagen en blanco y negro

Descripción generada automáticamente con confianza media

Bibliografía: Mascheroni, Maximiliano Agustín; Irrazábal, Emanuel; 2018

# Estudio de Métricas y Patrones de seguridad en Microservicios.

La tesis titulada "Estudio de Métricas y Patrones de Seguridad en Microservicios" escrita por Juana Victoria Penagos Sánchez aborda los desafíos de seguridad en esta arquitectura emergente, destacando la falta de estudios específicos y sistemáticos. Mediante un mapeo sistemático, analiza métricas y patrones de seguridad, identificando 49 patrones y la adaptación de métricas de otras arquitecturas, aunque muchas carecen de validación en entornos reales. Como contribución, propone una guía práctica basada en estándares de calidad (ISO/IEC 9126), con métricas clave y ejemplos aplicables. Concluye que la seguridad en microservicios es un área incipiente que requiere más investigación y validación práctica.  
Reflexión: La seguridad en microservicios es un campo emergente que requiere mayor validación práctica. Las métricas y patrones deben adaptarse para garantizar sistemas robustos y confiables.

Imagen en blanco y negro

Descripción generada automáticamente con confianza media

Bibliografía: Juana Victoria Penagos Sánchez; Enero de 2023

# Metodología para la identificación de sistemas a través de patrones

La tesis "Metodología para la identificación de sistemas a través de patrones", de Francisco Sánchez Canto, propone una metodología innovadora para sistematizar y transmitir el conocimiento experto en la identificación de sistemas, clave en el desarrollo de controladores industriales. Basada en la teoría de patrones, la metodología organiza el proceso en cuatro etapas: experimentación, preprocesado, obtención de parámetros y validación, complementadas por un catálogo y un lenguaje de patrones que guían de manera estructurada el desarrollo de modelos. Contribuye a reducir la curva de aprendizaje, mejorar la comunicación entre expertos y principiantes, y facilitar la aplicación de patrones en áreas técnicas complejas como la ingeniería de control.

Reflexión: Sistematizar la identificación de sistemas mediante patrones organiza procesos complejos, reduce la curva de aprendizaje y facilita la comunicación entre expertos y principiantes.

Diagrama, Esquemático

Descripción generada automáticamente

Bibliografía: Francisco Sánchez Canto; Febrero 2009

# Marco de Trabajo para Seleccionar un Patrón Arquitectónico en el Desarrollo de Software

El artículo presenta un marco de trabajo para seleccionar patrones arquitectónicos en el desarrollo de software, abordando problemas de desacoplamiento y falta de conocimiento arquitectónico que afectan la calidad de los productos. Basado en cinco fases, identifica los patrones más relevantes (como MVC, Microservicios y Arquitectura en la nube) y define reglas para elegir patrones según el tipo de desarrollo y características requeridas. El marco, validado mediante un prototipo, mejora tiempos, costos, calidad, escalabilidad y mantenibilidad. Concluye que la solución es una herramienta práctica que guía a arquitectos y desarrolladores en la selección de patrones adecuados, estableciendo estructuras correctas desde el inicio del proyecto.

Reflexión: Un marco práctico para elegir patrones arquitectónicos mejora la estructura del software desde el diseño inicial, promoviendo mantenibilidad, escalabilidad y menor costo a largo plazo.

Diagrama, Esquemático

Descripción generada automáticamente

Bibliografía: Mejía, Juan Camilo Giraldo; Agudelo, Fabio Alberto Vargas; Gil, Kelly Garzón; 25/07/2021

# Concepto de Arquitectura Software y Principios de Diseño

El documento es un material de estudio sobre \*\*Arquitectura de Software (AS)\*\*, dirigido a estudiantes de Ingeniería en Software, que aborda conceptos, principios y aplicaciones clave. Define la AS como el diseño de sistemas complejos basado en componentes, forma y lógica, destacando principios como modularidad y bajo acoplamiento. Explica el uso de patrones (creacionales, estructurales y de comportamiento) y estilos arquitectónicos como MVC y Cliente-Servidor para estructurar sistemas. Introduce el estándar \*\*ISO 42010\*\* para describir sistemas intensivos y presenta herramientas de gestión del conocimiento arquitectónico (AKM). También analiza aplicaciones en la \*\*Industria 4.0\*\*, como IoT y Big Data, y el uso de diagramas UML para documentar arquitecturas. Concluye que la AS combina decisiones iterativas y herramientas modernas para adaptarse a los desafíos actuales de la industria.

Reflexión: La arquitectura de software combina principios, herramientas y decisiones iterativas. Su aplicación correcta permite desarrollar sistemas modernos y adaptables, esenciales en la Industria 4.0.

Diagrama, Esquemático

Descripción generada automáticamente

Bibliografía: Rafael Capilla Sevilla; 01/012022

# Patrones. De Alexander a la Tecnología de Objetos

El texto explica los "patrones de diseño" en software, que son soluciones reutilizables a problemas comunes en el desarrollo de aplicaciones. Estos patrones surgieron a partir de las ideas del arquitecto Christopher Alexander, adaptadas al campo del software, especialmente en la programación orientada a objetos. El concepto se popularizó en 1995 con el libro Design Patterns de los "Cuatro de la Banda" (GoF), que presentó 23 patrones para resolver problemas recurrentes de diseño. Los patrones ayudan a crear sistemas más flexibles y fáciles de mantener, describiendo el contexto del problema, las fuerzas involucradas y la solución.

Reflexión: Los patrones de diseño inspiran soluciones flexibles y reutilizables, destacándose como un pilar para construir sistemas eficaces y fáciles de mantener en entornos orientados a objetos.

Un dibujo de un pizarrón blanco

Descripción generada automáticamente con confianza baja

Bibliografía: García-Peñalvo, Francisco J.; 1998-11

# Patrones y anti-patrones en equipos ágiles: una revisión sistemática

En la ingeniería del software, los patrones son soluciones reutilizables a problemas recurrentes, aceptadas por los profesionales para mejorar la calidad y reducir costes y errores. Su uso facilita la comunicación en los equipos y optimiza los procesos. La agilidad en el desarrollo de software busca reducir riesgos y garantizar la calidad y el cumplimiento de plazos y costes, promoviendo equipos autoorganizados. Los patrones y la agilidad comparten el objetivo de minimizar esfuerzos y asegurar calidad, y se integran en marcos ágiles como XP y Scrum. Este trabajo propone una investigación para desarrollar un marco de trabajo de patrones ágiles que apoyen la auto-organización de equipos a lo largo de su evolución.

Reflexión: Los patrones ágiles optimizan la autoorganización de equipos, fomentando la calidad y eficiencia en el desarrollo de software, especialmente en marcos como Scrum y XP.

Pizarrón blanco con texto en letras negras sobre fondo blanco

Descripción generada automáticamente con confianza media

Bibliografía: Ramos Vega, Cristina; 2019

# Sistema para el Control y Seguimiento de la Implementación de la Metodología de Gestión de Procesos de Negocio sustentada en el Uso de Patrones

En las empresas de desarrollo de software, bajo el paradigma de Gestión de Proceso de Negocio, se requiere un sistema para gestionar el seguimiento y control de los métodos, herramientas y técnicas utilizadas en el desarrollo, con el fin de lograr la reutilización de componentes en otros proyectos. Esto se logra mediante la adaptación y asociación a un modelo de calidad, lo que facilita la trazabilidad del proceso. Este Trabajo Especial de Grado propone implementar un sistema que valide estas hipótesis, permitiendo garantizar la calidad del producto final en cada fase mediante la reutilización de experiencias exitosas. La solución se basa en una metodología de Gestión de Proceso de Negocio con patrones, propuesta por el Dr. Pedro Bonillo, utilizando las plataformas Intalio BPMS 6.0 y NetBeans IDE con el framework JSF. Estas herramientas facilitan modelar, diseñar y ejecutar el proceso de negocio, adaptándolo a las mejores prácticas de cualquier organización.

Reflexión: Implementar sistemas basados en patrones para la gestión de procesos empresariales asegura la trazabilidad y reutilización de componentes, mejorando la calidad final del producto.

Imagen que contiene texto, pizarrón

Descripción generada automáticamente

Bibliografía: González, Deiby G.; Mendoza, Mayerling D.; 2-Dec-2014

# GENERACIÓN DE LA INTERFAZ DE USUARIO DE NEGOCIO A PARTIR DE PATRONES DE NEGOCIOS BASADA EN LOS FUNDAMENTOS METODOLÓGICOS DE TD-MBUID

El documento propone un proceso basado en modelos para desarrollar interfaces de usuario de negocio, utilizando la metodología TD-MBUID. A través de la asociación de patrones de datos, plantillas de presentación y modelos de interacción, busca interpretar mejor los modelos mentales de los usuarios sobre los datos. Este enfoque mejora la calidad y el tiempo de diseño de prototipos de interfaces, utilizando el lenguaje DataForm y herramientas de Eclipse. Su objetivo es optimizar la creación de interfaces alineadas con las necesidades del usuario final, aplicando patrones de diseño y metodologías basadas en modelos para el desarrollo eficiente de software empresarial.  
Reflexión: Un enfoque basado en modelos para desarrollar interfaces de usuario mejora la alineación con las necesidades del usuario y optimiza el diseño eficiente en proyectos empresariales.

Pizarrón blanco con texto en letras negras sobre fondo blanco

Descripción generada automáticamente con confianza media

Bibliografía: Triviño Arbeláez, Jorge Iván; 2016

# APLICACIÓN DE PATRONES DE DISEÑO ESTRUCTURALES PARA EL MODELAMIENTO DE CLASES DE LOS SISTEMAS EMPRESARIALES

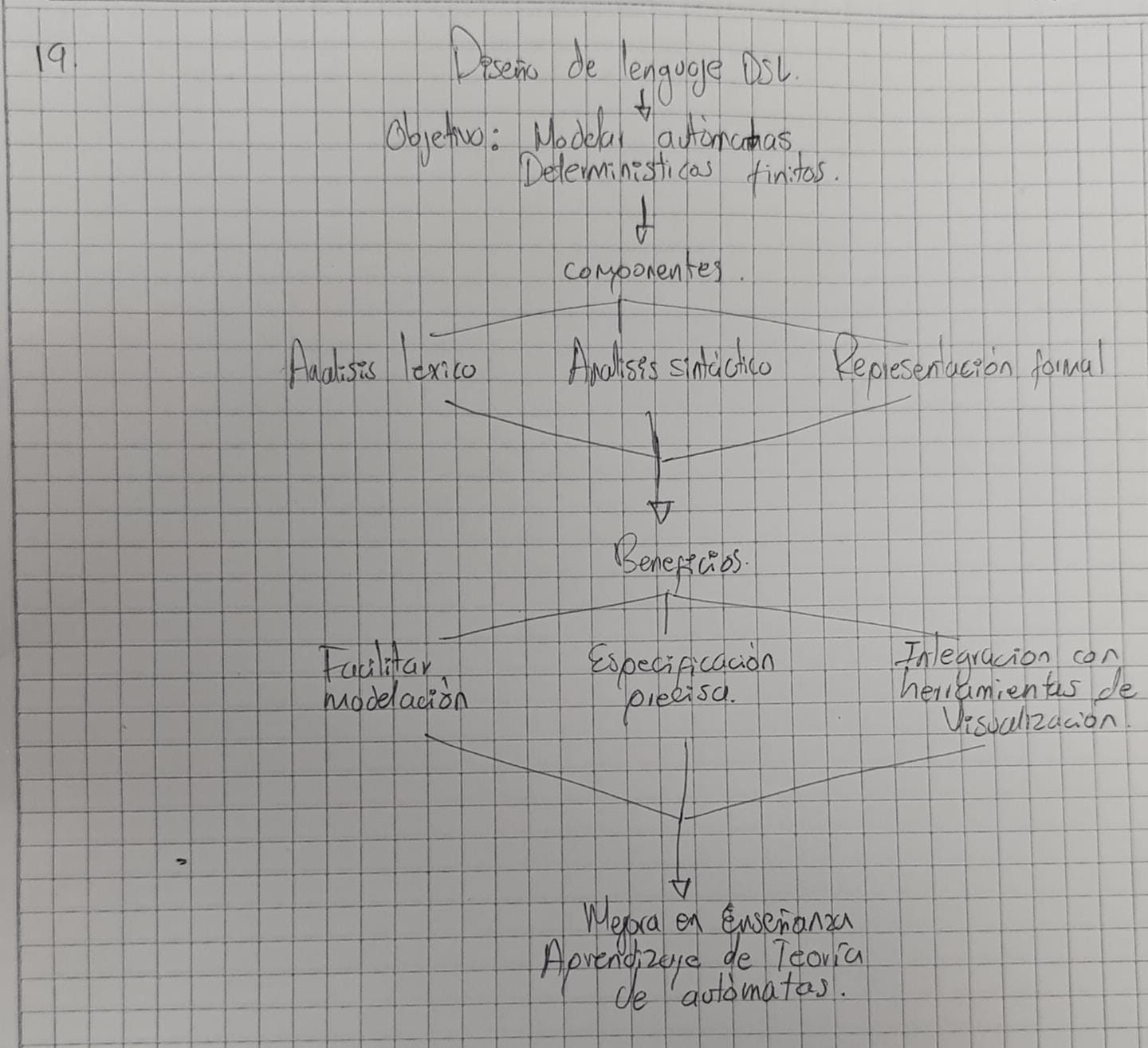
La tesis titulada "Aplicación de patrones de diseño estructurales para el modelamiento de clases de los sistemas empresariales" fue presentada por Manuel Mariano Zúñiga Carnero en 2020 para optar al grado de Doctor en Ingeniería de Sistemas en la Universidad Nacional Federico Villarreal. El estudio aborda la creciente complejidad de los sistemas empresariales y la necesidad de patrones de diseño específicos que faciliten el modelamiento de datos en sistemas de información administrativos. Mediante una investigación de enfoque cuantitativo, tipo aplicada, nivel correlacional y diseño pre-experimental, se evaluó la aplicación de patrones de diseño estructurales en el modelamiento de clases. La muestra incluyó a 76 estudiantes de Ingeniería de Sistemas de la Universidad Católica de Santa María. Los resultados indicaron una relación directa y positiva entre la aplicación de estos patrones y la mejora en el modelamiento de datos en sistemas empresariales, destacando la importancia de utilizar soluciones probadas para abordar problemas de diseño específicos en este ámbito.   
Reflexión: El uso de patrones estructurales facilita el modelamiento de sistemas complejos, asegurando calidad y permitiendo abordar desafíos específicos en sistemas empresariales.

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media  
Bibliografía: Zúñiga Carnero, Manuel Mariano; 2020

# Diseño e implementacion de un lenguaje para modelar autónamas deterministicos finitos

El trabajo de grado titulado "Diseño e implementación de un lenguaje para modelar autómatas determinísticos finitos" propone un lenguaje de dominio específico (DSL) que facilita a los estudiantes la modelación y especificación de autómatas finitos determinísticos en términos de sus propiedades. Este DSL permite una representación formal y precisa de los autómatas, integrándose con herramientas de visualización existentes para mejorar la comprensión y el aprendizaje de estos conceptos. La implementación del lenguaje se realizó utilizando técnicas de análisis léxico y sintáctico, asegurando su funcionalidad y usabilidad en entornos educativos. El estudio concluye que el uso de este DSL contribuye significativamente al proceso de enseñanza y aprendizaje de la teoría de autómatas, proporcionando una herramienta efectiva para la representación y análisis de autómatas finitos determinísticos.

Reflexión: Diseñar lenguajes específicos para modelar autómatas fortalece la enseñanza de teoría computacional, brindando herramientas prácticas y efectivas para su representación.

Bibliografía: Zúñiga Carnero, Manuel Mariano; Julio 2007

# Biologística: Estrategias computacionales bioinspiradas para la gestión de las redes de suministro complejas

El artículo "Biologística: Estrategias computacionales bioinspiradas para la gestión de las redes de suministro complejas" propone un modelo conceptual y tecnológico que, inspirado en sistemas biológicos y bajo la lógica de sistemas colectivos, busca una gestión más integral y orgánica de las redes de suministro. Los autores, Nelson Alfonso Gómez-Cruz y John Leonardo Vargas Mesa, argumentan que las cadenas de suministro tradicionales, basadas en control, optimización y predicción, son insuficientes para enfrentar desafíos como la personalización en masa y la adaptabilidad en entornos cambiantes. Al considerar las redes de suministro como sistemas colectivos, se pueden aprovechar características emergentes como flexibilidad, robustez y resiliencia. El estudio sugiere que la aplicación de nociones y métodos bioinspirados permitirá una gestión más eficiente y adaptable de estas redes complejas.

Reflexión: Aplicar estrategias bioinspiradas a las redes de suministro potencia su flexibilidad, resiliencia y adaptabilidad, abordando desafíos complejos con enfoques innovadores.

Diagrama, Esquemático, Escala de tiempo

Descripción generada automáticamente

Bibliografía: Giraldo G., Gloria L.; Acevedo O., Juan F.; Moreno N., David A.; 10 de noviembre de 2011

# Desarrollo de framework web para el desarrollo dinámico de aplicaciones.

El documento propone una ontología para estructurar y organizar el conocimiento del diseño de software, abordando diagramas y patrones esenciales para esta etapa del ciclo de vida del desarrollo. Se define ontología como una representación formal y compartida de conceptos que mejora la comunicación y reutilización de conocimiento. La propuesta incluye diagramas estructurales (clases, componentes) y de comportamiento (secuencia, actividades), además de patrones de diseño como GRASP y GoF, que abarcan creacionales, estructurales y de comportamiento. Esto permite resolver problemas recurrentes en múltiples niveles de abstracción. La ontología pretende sistematizar conceptos que suelen estar dispersos, facilitando su enseñanza y práctica profesional. Se destaca la utilidad de estas herramientas en la mejora de la calidad del software y la promoción de buenas prácticas. Asimismo, aborda el diseño de sistemas colaborativos para enseñar estos conceptos a estudiantes de ingeniería, ayudando a estandarizar el conocimiento y fomentar una mejor aplicación en proyectos de software.

Reflexion:La implementación de este Framework Web resalta la importancia de optimizar procesos clave como autenticación, gestión de usuarios y generación de formularios, mejorando la eficiencia y personalización del desarrollo. La selección de un modelo pequeño permitió validar los requerimientos y ajustar funcionalidades antes de escalar. Definir estándares de codificación y aprovechar una arquitectura Cliente/Servidor asegura calidad y sostenibilidad en el software. Sin embargo, se identifican retos técnicos, como la compatibilidad de librerías y acceso dinámico a datos, que requieren atención continua. Esto refuerza la necesidad de pruebas exhaustivas y mejoras constantes para garantizar un producto robusto y adaptable.

Un dibujo de un pizarrón blanco

Descripción generada automáticamente con confianza media

Bibliografía: Martínez Villalobos, Gustavo; Camacho Sánchez, Germán Darío; Biancha Gutiérrez, Daniel Alberto; abril 2010

# Tipificación de Dominios de Requerimientos para la Aplicación de Patrones Arquitectónicos

El artículo aborda la complejidad de la arquitectura de software y propone una metodología para facilitar la selección de patrones arquitectónicos en el desarrollo de sistemas. Los investigadores analizaron 68 proyectos de software para identificar requisitos funcionales y no funcionales comunes, creando un "dominio de requerimientos" que permite asociar patrones arquitectónicos específicos con diferentes tipos de requisitos.

El propósito principal es reducir la brecha entre ingenieros en formación y profesionales experimentados, proporcionando una guía sistemática para seleccionar patrones arquitectónicos. La investigación busca transformar un proceso tradicionalmente basado en la experiencia individual en un método más estructurado y reproducible.

Como validación, aplicaron su metodología en un proyecto real de evaluación docente para una institución educativa, desarrollando una aplicación móvil que demostró la utilidad del enfoque propuesto. El resultado es una herramienta que ayuda a los desarrolladores a tomar decisiones arquitectónicas más informadas y consistentes.

Reflexión: El uso de patrones arquitectónicos en el desarrollo de software es clave para garantizar calidad y eficiencia. Aunque su implementación no siempre está documentada, representan una herramienta valiosa para organizar el diseño y cumplir con los requisitos funcionales y no funcionales. Sin embargo, su aplicación efectiva requiere sensibilización en los equipos de desarrollo y estándares claros que promuevan buenas prácticas. Además, la integración de lenguajes de modelado como UML facilita la identificación y selección de patrones adecuados. Este enfoque permite mejorar la navegabilidad del código, optimizar procesos y asegurar que las soluciones arquitectónicas sean sostenibles y adaptables al contexto.

Imagen en blanco y negro de un pizarrón blanco

Descripción generada automáticamente con confianza baja

Bibliografía: Johanna M. Suárez; Luz E. Gutiérrez; 2016

# Buenas prácticas en la construcción del software.

El artículo reflexiona sobre la importancia de las buenas prácticas en la construcción de software en el contexto de la cuarta revolución industrial, donde las organizaciones enfrentan desafíos para desarrollar sistemas de información complejos de forma eficiente y sin exceder costos. Se destacan estándares como Clean Code, que promueve código simple y fácil de leer, y principios como KISS, DRY y el diseño de funciones pequeñas y específicas. También se resalta la relevancia de una correcta arquitectura para satisfacer requisitos funcionales y no funcionales, abarcando estilos como arquitecturas en capas, monolíticas, microservicios y basadas en eventos. Los patrones de diseño se presentan como herramientas clave para solucionar problemas comunes y estructurar el código de forma eficiente. Además, se enfatiza el uso de herramientas tecnológicas y metodologías como XP para optimizar procesos, mantener flexibilidad y asegurar entregables funcionales en ciclos iterativos. Por último, el artículo concluye que integrar estas prácticas y estándares no solo mejora la calidad y mantenibilidad del software, sino que también reduce tiempos y costos, beneficiando a las organizaciones y sus equipos de desarrollo.

Reflexión:  
Promover una cultura organizacional basada en buenas prácticas de desarrollo es esencial para la creación de software eficiente y sostenible. La adopción de estándares como Clean Code facilita la comprensión del código, mejora su mantenibilidad y reduce esfuerzos y costos operativos. Además, el uso de herramientas tecnológicas y marcos de trabajo permite automatizar tareas, optimizando el rendimiento de los equipos de desarrollo. Diseñar una arquitectura adecuada es crucial para alinear las soluciones con las necesidades funcionales y no funcionales del negocio, garantizando un desarrollo estructurado y efectivo. Estas prácticas fortalecen la calidad y el impacto del software en las organizaciones.

Diagrama

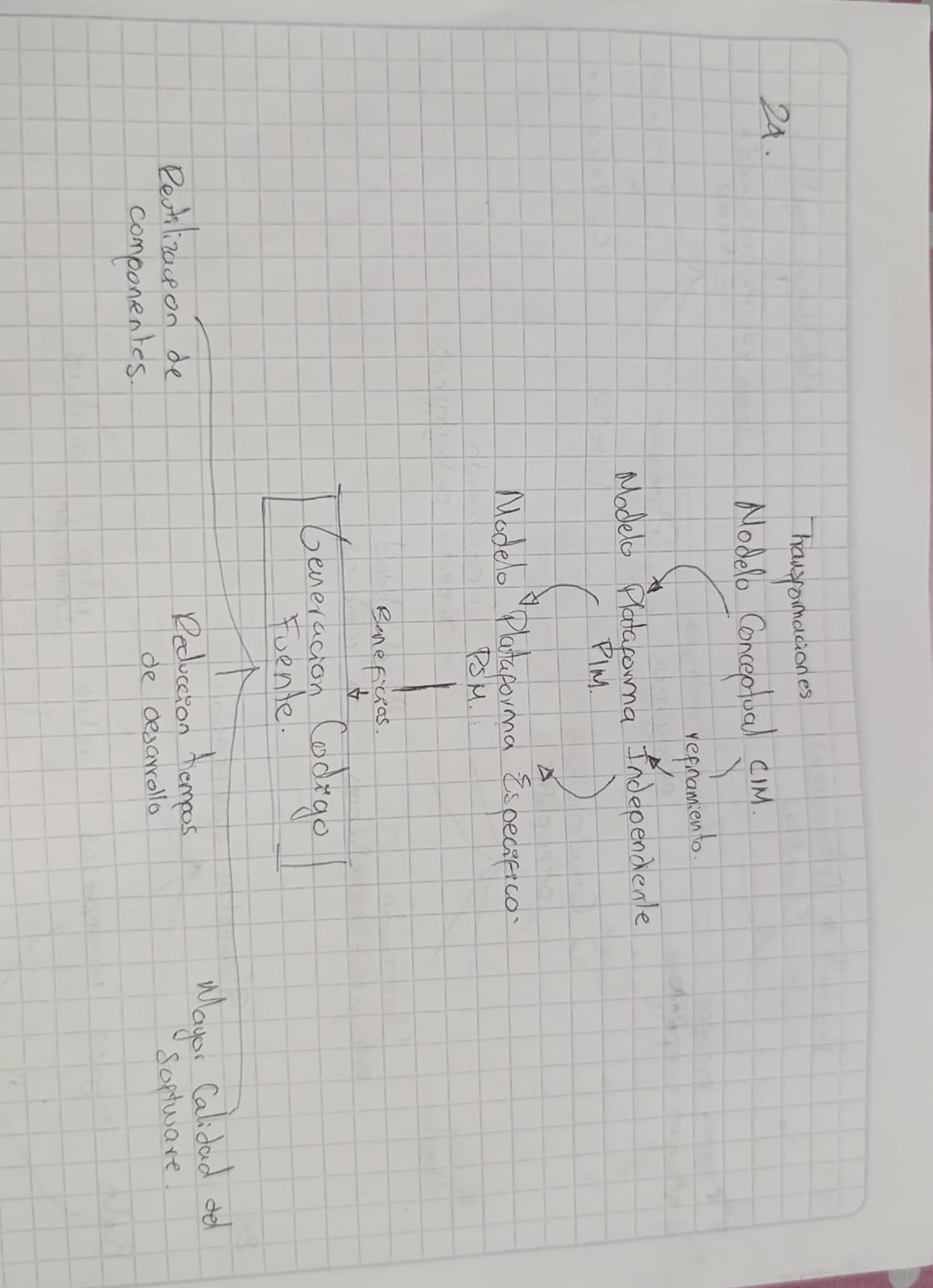
Descripción generada automáticamente

Bibliografía: Camilo Alberto Prieto; Diego Alejandro Madrid; 2024-01-23

# Líneas de Productos Software: Generando Código a Partir de Modelos y Patrones

El artículo explora las Líneas de Productos Software (LSP), un enfoque que fomenta la reutilización de componentes para mejorar la eficiencia en el desarrollo. Basado en Model Driven Development (MDD) y Model Driven Architecture (MDA), LSP permite crear sistemas con características compartidas desde un núcleo común, optimizando tiempos y productividad. MDA emplea modelos como CIM, PIM y PSM para representar diferentes niveles de abstracción y usa procesos de transformación que convierten modelos independientes en específicos, generando código adaptable a diversas tecnologías. Un prototipo desarrollado ilustra este enfoque, utilizando PHP, SQL y el patrón MVC para generar interfaces CRUD. Concluye que LSP incrementa la calidad y reutilización del software, mientras que la automatización reduce significativamente los tiempos de desarrollo.

Reflexión: La Línea de Productos Software (LSP) destaca como una estrategia clave para optimizar el desarrollo, al reutilizar artefactos flexibles y garantizar calidad. Su generación automatizada simplifica la creación de componentes esenciales como vistas CRUD, mientras que la Arquitectura Dirigida por Modelos (MDA) amplía la adaptabilidad al permitir implementaciones en diversas plataformas. Sin embargo, lenguajes como OCL aún requieren mayor adopción y madurez para consolidar este enfoque como un estándar industrial. LSP y MDA ofrecen un camino prometedor hacia la agilidad y eficiencia, aunque enfrentan retos en herramientas y estandarización.



Bibliografía: Gaitán Peña, C. A. (2017)

# BUENAS PRÁCTICAS PARA EL DESARROLLO DE APLICACIONES WEB A TRAVÉS DE PATRONES DE DISEÑO USANDO EL MARCO DE TRABAJO ZEND FRAMEWORK

El documento aborda la implementación de buenas prácticas en el desarrollo de aplicaciones web utilizando patrones de diseño y el Zend Framework. Se identifican problemas comunes en aplicaciones web que afectan su mantenibilidad y escalabilidad, proponiendo patrones como Adapter, Composite, Decorator y Factory Method para mejorar la calidad del código. Además, se detallan estándares de codificación específicos de Zend Framework, como convenciones de nombres, manejo de variables y documentación, que contribuyen a generar código entendible y mantenible. Finalmente, se evalúa la aplicabilidad de estas prácticas mediante ejemplos concretos y pruebas, mostrando cómo estas estrategias facilitan el desarrollo, reducen errores y mejoran la escalabilidad de las aplicaciones.

Reflexión:Los patrones de diseño y estándares como los de Zend Framework garantizan código mantenible, escalable y entendible. Facilitan cambios localizados, minimizan errores y promueven comprensión entre programadores. Usar nombres descriptivos y documentar adecuadamente simplifica ajustes y reduce costos, evitando retrabajos. Código bien escrito desde el inicio es clave para un desarrollo eficiente y sostenible.

Diagrama

Descripción generada automáticamente

Bibliografía: Correa Isaza, Daniel Guillermo; Mesa Fernández, Nicolás; 2012

# Modelo de Investigación en Ingeniería del Software:Una propuesta de investigación tecnológica

El documento sobre el "Modelo de Investigación en Ingeniería del Software" destaca que la investigación en esta disciplina busca generar conocimiento aplicable en entornos industriales, definiendo métodos y modelos útiles para el desarrollo de software. La estrategia propuesta consta de tres fases: investigación inicial, investigación aplicada a través de proyectos reales y transferencia del conocimiento mediante cursos y conferencias. Además, se identifican distintos tipos de investigaciones, enfocándose en el tipo de pregunta, resultado y validación. Los métodos de investigación se clasifican en observación, históricos y controlados. Se concluye que es esencial una estrategia que vincule la investigación académica con las necesidades industriales, manteniendo un enfoque metodológico riguroso.

Reflexión: El texto destaca que la Ingeniería de Software es una profesión tecnológica enfocada en desarrollar y aplicar tecnologías en proyectos reales. La investigación debe estar vinculada con la industria, validando tecnologías en entornos reales y transfiriéndolas a las empresas. Involucrar a las empresas no disminuye el rigor investigativo, sino que es clave para asegurar la aplicabilidad. Reflexionar sobre las estrategias y métodos investigativos, así como articular teorías existentes, es esencial para desarrollar tecnologías que realmente respondan a las necesidades de la industria.

Diagrama

Descripción generada automáticamente

Bibliografía: Jaime A. Chavarriaga L.; Hugo F. Arboleda J.; 1/05/1998

# Metodología ágil para el desarrollo de aplicaciones móviles educativas

El documento propone una metodología ágil para el desarrollo de aplicaciones móviles educativas, integrando aspectos pedagógicos y tecnológicos. Basada en la Programación Extrema (XP) y un framework que articula componentes pedagógicos con metodologías de software, la propuesta se centra en un proceso iterativo y flexible que fomenta la colaboración entre pedagogos y desarrolladores, asegura la adaptabilidad a cambios de requisitos y prioriza los aspectos pedagógicos desde el inicio. Incluye etapas de planeación, diseño, codificación y pruebas, orientadas a crear herramientas móviles sólidas y efectivas para el aprendizaje. Como ejemplo, se desarrolló una aplicación para prevenir el grooming, un proyecto realizado por estudiantes del Profesorado en Informática de la Universidad Nacional de Santiago del Estero. En definitiva, esta metodología busca aprovechar la tecnología móvil como un apoyo pedagógico para mejorar los procesos de enseñanza-aprendizaje.

Reflexión: El documento destaca la importancia de integrar pedagogía y tecnología en el desarrollo de aplicaciones móviles educativas, proponiendo una metodología que no solo responde a necesidades técnicas, sino también a objetivos de aprendizaje. Al priorizar la colaboración interdisciplinaria y la flexibilidad, invita a repensar la tecnología móvil como una herramienta significativa para la enseñanza, enfocada en solucionar problemas reales, como la concientización sobre el grooming, y en enriquecer los procesos educativos de manera efectiva e innovadora.

Diagrama

Descripción generada automáticamente

Bibliografía: Lara, Cecilia; Figueroa, Liliana Maria; 2020

# La Ingeniería de Software Libre y sus Herramientas Aplicadas a Proyectos Informáticos

El documento analiza la gamificación como estrategia para mejorar la motivación, el compromiso y la productividad en el desarrollo de software, identificando y clasificando 49 elementos clave a través de una revisión sistemática de literatura. Entre los más utilizados y efectivos destacan puntos, insignias y recompensas, que han demostrado su capacidad para motivar a los participantes y optimizar su desempeño. También se consideran otros elementos como retos, retroalimentación y avatares, que enriquecen la experiencia de gamificación. Finalmente, se propone un catálogo de estos elementos aplicable tanto en contextos académicos como en la industria, con el objetivo de potenciar la eficacia y compromiso de los equipos de desarrollo.

Reflexión: El texto resalta la importancia de la gamificación en la ingeniería de software, tanto educativa como industrial, para mejorar motivación, colaboración y compromiso. Aunque predominan puntos, insignias y tablas de posiciones, se requiere explorar combinaciones más efectivas. La creación de un catálogo gamificado facilitará estrategias personalizadas, ampliando su impacto en distintos contextos.

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Bibliografía: Liliana Machuca-Villegas; Gloria Piedad Gasca-Hurtado; Luz Marcela Restrepo Tamayo; Solbey Morillo Puente; 23/11/2019

# Modelos de casos de uso

El documento analiza la aplicación del modelo de casos de uso en entornos de desarrollo global de software, abordando los desafíos de comunicación, coordinación y control en equipos distribuidos. Mediante el uso de Desarrollo Dirigido por Modelos (MDD), se refina la plantilla CUPIDo para adaptarla a contextos globales, permitiendo la documentación automatizada y generación de diagramas. La investigación, enmarcada en proyectos de la Universidad Nacional de San Juan, busca mejorar la especificación de requisitos funcionales, integrar herramientas colaborativas y generar modelos útiles para diversas etapas del desarrollo. Los resultados preliminares incluyen la versión 1.4 de CUPIDo y su extensión CUPIDo+Pi, destacando su relevancia para la ingeniería de software en un entorno globalizado.

Reflexión: El documento destaca la necesidad de adaptar los casos de uso al desarrollo de software global, enfrentando desafíos de comunicación y coordinación en equipos distribuidos. La investigación, centrada en la plantilla CUPIDo y su automatización, muestra cómo innovar en la especificación de requisitos funcionales puede optimizar procesos y responder a las demandas del trabajo colaborativo en un mundo globalizado.

Diagrama

Descripción generada automáticamente

Bibliografía: Lund, María Inés; Chávez, Susana Beatriz; Ormeño, Emilio Gustavo; Martín, Adriana Elizabeth; Matturro, Gerardo; 2014

# Patrones de interacción para el diseño de videojuegos en smartphones

El documento "Patrones de Interacción para el Diseño de Videojuegos en Smartphones" de Leandro Filigrana, Andrés Solano y César A. Collazos investiga la usabilidad de los videojuegos móviles mediante patrones de interacción. Identifica problemas comunes que afectan la experiencia del usuario, como la baja calidad en la física del juego y las limitaciones en la personalización de controles. La investigación aplica patrones de interacción basados en soluciones probadas para problemas recurrentes, como la "Configuración del Área de juego" y el "Despliegue de información de alta densidad". Se espera que los patrones propuestos mejoren las herramientas y paradigmas en la industria de los videojuegos móviles, facilitando una experiencia de usuario más amigable. El documento concluye subrayando la importancia de reconfigurar el área de juego y presentar opciones de manera accesible para el usuario.

Reflexión: El documento propone una investigación para mejorar la interacción de los jugadores con los videojuegos en smartphones mediante patrones de interacción adaptados. Estos patrones buscan optimizar el diseño al solucionar problemas comunes, como la presentación de información y la facilidad de uso. Considerando las características específicas de los smartphones, la investigación tiene el potencial de hacer los videojuegos más intuitivos, reduciendo errores y la necesidad de asistencia externa. A futuro, se evaluarán estos patrones mediante un caso de estudio, validando su efectividad en el diseño de videojuegos móviles.

Diagrama

Descripción generada automáticamente

Bibliografía: Leandro Filigrana; Andrés Alegría; César A. Collazos; 2015-06-01

# Metodología para la construcción de Interfaces Gráficas Centradas en el Usuario

El documento propone una metodología para diseñar Interfaces Gráficas de Usuario (GUI) centradas en el usuario, combinando diseño centrado en el usuario, pensamiento de diseño, desarrollo ágil, usabilidad y experiencia de usuario. Destaca la importancia de las GUI en la interacción humano-computadora y cómo su calidad impacta la percepción del sistema. La metodología incluye nueve fases iterativas, desde la estructuración hasta el despliegue, involucrando al usuario en cada etapa. Se presenta un caso de estudio basado en una aplicación de comunicación para personas con necesidades complejas, enfrentando retos como limitaciones motrices y cognitivas. Se enfatiza la validación futura de esta metodología y el desarrollo de herramientas para evaluar su eficacia.

Reflexión: El documento destaca la importancia de diseñar GUI centradas en el usuario para mejorar interacción y percepción del sistema. Su enfoque iterativo e inclusivo asegura soluciones funcionales y accesibles, enfrentando el reto de equilibrar necesidades del usuario y viabilidad técnica. Este enfoque es clave para un desarrollo tecnológico inclusivo y adaptable.

Diagrama, Esquemático

Descripción generada automáticamente

Sebastián Sastoque; Cristian Narváez; German Ganica; 2016

# Arquitectura de software con programación orientada a objeto

El artículo examina la arquitectura de software desde la programación orientada a objetos (POO), destacando su capacidad para organizar sistemas mediante componentes, relaciones y principios de diseño que satisfacen requisitos funcionales y no funcionales. La POO utiliza conceptos como clases, objetos, herencia y métodos para modelar problemas de manera natural, acercando los conceptos al diseño práctico. Lenguajes como Python, Java, PHP y C++ han adoptado características de POO, siendo C++ y Java los más usados por su robustez. Entre las ventajas destacan la reutilización de código, mejor mantenibilidad y la integración de soluciones preexistentes. Esta evolución permite desarrollar software eficiente, rápido y de calidad, mejorando la experiencia del usuario.

Reflexión: La POO transforma el desarrollo de software al modelar problemas de forma natural, optimizando recursos y fomentando soluciones escalables. Su enfoque en la reutilización de código y mantenibilidad mejora la calidad del software, aunque requiere dominar conceptos clave. Lenguajes como Java y C++ consolidan su relevancia, haciendo de la POO una herramienta esencial para satisfacer necesidades cambiantes en tecnología.

Diagrama, Esquemático

Descripción generada automáticamente

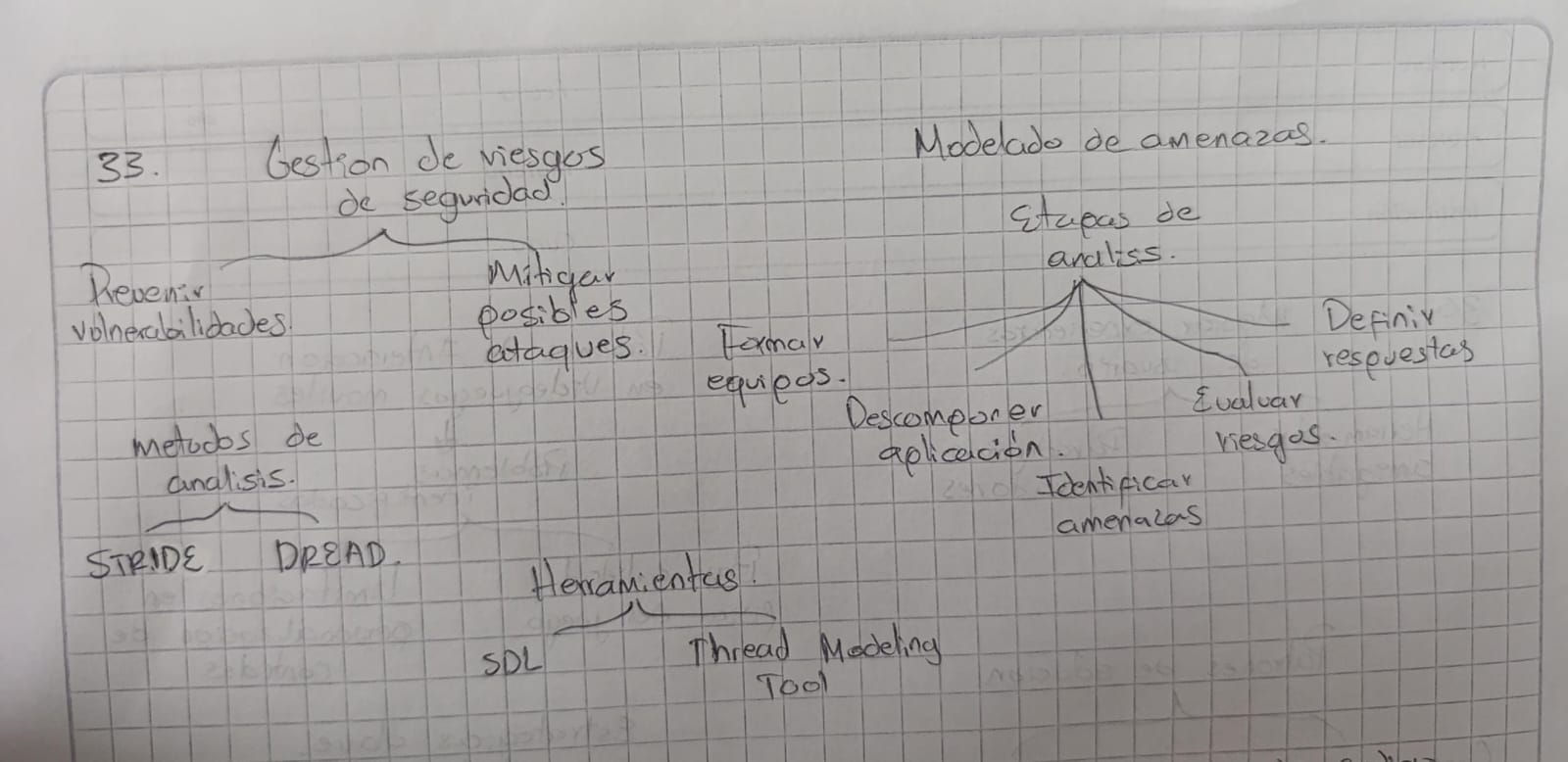
Bibliografía: Vera Vera, José Belisario ; 2023

# Modelo de amenazas, una ténica de análisis y gestión de riesgo asociados a software y aplicaciones

El documento analiza el \*\*modelado de amenazas\*\*, una técnica clave en la gestión de riesgos para garantizar la seguridad de software y sistemas. Se destaca su relevancia frente al aumento de vulnerabilidades debido a desarrollos acelerados. El proceso incluye identificar amenazas, evaluar riesgos y aplicar controles para mitigar posibles ataques.

Se detallan cinco etapas esenciales: formar un equipo de análisis, descomponer la aplicación para identificar componentes críticos, determinar amenazas usando métodos como STRIDE, asignar valores de riesgo mediante DREAD y definir respuestas priorizadas. Además, el texto menciona metodologías y herramientas como SDL y Threat Modeling Tool para implementar esta técnica.El modelado de amenazas se presenta como una práctica fundamental para integrar la seguridad en el ciclo de desarrollo del software (SDLC), contribuyendo a prevenir pérdidas económicas y a mantener la confianza de los usuarios. Aunque su implementación puede parecer costosa a corto plazo, sus beneficios a largo plazo justifican el esfuerzo, especialmente en sistemas críticos y aplicaciones web.

Reflexión: El modelado de amenazas es una herramienta esencial en la construcción de software seguro, ya que permite identificar y mitigar riesgos desde las primeras etapas del desarrollo. Aunque puede implicar costos iniciales, su implementación contribuye a prevenir vulnerabilidades que, a largo plazo, podrían generar pérdidas económicas y dañar la reputación de las organizaciones. Al integrar esta práctica de manera continua a lo largo del ciclo de vida del software, las empresas no solo protegen sus sistemas, sino que también mejoran la confianza de los usuarios y optimizan la seguridad de sus aplicaciones.



Bibliografía: Barba Olivares, Gabriel Eduardo; 2017-08-24

# Revisión sistemática sobre generadores de código fuente y patrones de arquitectura

Los proyectos de desarrollo de software pueden enfrentar retrasos y baja calidad debido a deficiencias en la organización del código y la integración de componentes. Para mejorar la productividad y reducir estos riesgos, se pueden utilizar Generadores de Código Fuente (GCF), que automatizan la creación de código en tareas repetitivas, como interfaces de usuario, conexiones a bases de datos y reportes. Además, es fundamental planificar y diseñar previamente la arquitectura del software para asegurar una estructura organizada. A través de una revisión sistemática de la literatura, se identificaron patrones de arquitectura y herramientas utilizadas en la generación de código para aplicaciones web. Se concluye que los patrones de arquitectura son clave para estandarizar y organizar el código en capas, facilitando el desarrollo y mejorando la calidad del software.

Reflexión: El uso de Generadores de Código Fuente (GCF) y patrones de arquitectura es clave para mejorar la calidad y eficiencia en el desarrollo de software. Estos enfoques automatizan tareas repetitivas, reducen errores y permiten la reutilización, lo que acelera el proceso de creación y mantenimiento de aplicaciones. Además, al integrar patrones de arquitectura, como el MVC o el multicapa, se mejora la organización y la escalabilidad del código. La relación entre los GCF, los patrones y los frameworks demuestra cómo la estandarización y la automatización pueden optimizar el desarrollo, facilitando la creación de sistemas más robustos y fáciles de mantener.

Diagrama, Esquemático

Descripción generada automáticamente

Bibliografía: Huari Casas, María Rosario; 2020-06-18

# Modelo de notificaciones para patrones de integración

El texto destaca la competitividad actual en el desarrollo de soluciones de software, donde los ingenieros deben tener competencias para planificar proyectos con tiempos reducidos sin comprometer la calidad. Se sugiere que una estrategia efectiva es reutilizar, producir e integrar herramientas existentes para superar las desventajas de proyectos previos y cumplir con los nuevos requerimientos del cliente. El uso de modelos de mensajería, patrones de integración y arquitecturas como SOA (Arquitectura Orientada a Servicios) son claves para optimizar el desarrollo y entregar soluciones eficientes. Además, se menciona el proceso de elaboración de una solución informática para la empresa Cm-Concretos, que abarca desde la arquitectura tecnológica hasta el diseño de procesos de negocio con herramientas como BPM.

Reflexión: La empresa está aprovechando herramientas metodológicas para priorizar y mejorar sus procesos clave, lo que les permite integrar eficazmente diferentes enfoques arquitectónicos en sus proyectos. Al priorizar, han logrado implementar mejoras en sus modelos de mensajería, mejorando la integración entre sus herramientas y manteniendo altos niveles de calidad y satisfacción del cliente. Los flujos de trabajo y el diseño de patrones de mensajería han sido fundamentales para una implementación transparente y ágil de enfoques SOA, minimizando riesgos y facilitando el desarrollo.

Diagrama

Descripción generada automáticamente

Bibliografía: Romero González, Luis Daniel; 2012-12-13

# Diseño de un framework para la creación automática de piezas de software para la implementación de sistemas informáticos empresariales

El documento tiene como propósito mejorar el proceso de desarrollo de software, específicamente en sistemas con acceso a bases de datos relacionales, donde las rutinas complejas requieren programadores especializados para su mantenimiento. Dado que las piezas de software comparten características comunes, los programadores tienden a reutilizar y adaptar código existente. Se propone estandarizar el proceso de codificación mediante reglas que faciliten la automatización del desarrollo. Para ello, se creó la herramienta CASE TheCoder, que genera automáticamente el código necesario. La validación de la herramienta mostró una reducción significativa en el tiempo de codificación, manteniendo la calidad del código.

Reflexión: La estandarización y automatización en el proceso de codificación mejoran la eficiencia y calidad del software, facilitando su integración y mantenimiento. El uso de normas de codificación hace el código más legible, mientras que la interrelación entre el modelo relacional de bases de datos y el diagrama de clases asegura la coherencia de la aplicación. Estos enfoques permiten un desarrollo más ágil y reducido en errores.

Diagrama, Esquemático

Descripción generada automáticamente

Bibliografía: Terán Santa Cruz, Franklin Edinson; 2021-05-06

# Identificación de la interacción competitiva de patrones en líneas de producto software

Este artículo aborda la Ingeniería de Líneas de Productos de Software (SPLE) y su relación con los atributos de calidad (QAs) y los patrones de diseño. SPLE busca desarrollar sistemas que compartan características comunes y se adapten a los requisitos de un dominio específico. Los QAs se refieren a atributos no funcionales que son esenciales en la ingeniería de software. Las interacciones entre los QAs y los patrones de diseño pueden ser sinérgicas o competitivas, y cuando son competitivas, se deben clasificar como adaptables o excluyentes. El artículo destaca la importancia de identificar estas interacciones en las primeras etapas de desarrollo para evitar problemas en productos posteriores. Además, se presenta una metodología para clasificar e identificar estas interacciones utilizando repositorios de QAs y patrones de diseño, con el fin de mejorar la toma de decisiones en el desarrollo de aplicaciones empresariales.

Reflexión: El artículo destaca la importancia de gestionar las interacciones entre patrones de diseño y atributos de calidad en el desarrollo de software. Resolver conflictos entre patrones que promueven diferentes niveles de calidad es clave para asegurar la eficiencia y fiabilidad del software. Las estrategias propuestas, como crear nuevos patrones o utilizar razonamiento basado en restricciones, ofrecen soluciones para integrar atributos de calidad de manera efectiva. Este enfoque amplía el modelo de calidad existente y abre la puerta a futuras investigaciones, mejorando la adaptabilidad y la calidad de las aplicaciones empresariales, garantizando productos robustos y fáciles de mantener.

Diagrama, Esquemático

Descripción generada automáticamente

Bibliografía: Julian Cifuentes; Hugo Arboleda; 2015

# Proceso de requerimiento y análisis para la definición de la arquitectura desde la perspectiva de usabilidad para el desarrollo de aplicaciones en la Web

El artículo presenta un proceso innovador para el desarrollo de aplicaciones web, enfocado en la usabilidad y diseñado para pequeñas y medianas empresas. El proceso, que busca superar las limitaciones de metodologías tradicionales, incluye etapas como obtención de información, especificación de requerimientos, validación y definición de arquitectura, con un enfoque especial en la usabilidad. La propuesta aborda problemas como la complejidad de las aplicaciones web y la falta de métodos adaptados a empresas pequeñas. Integra la usabilidad desde el inicio, asegurando que el producto sea intuitivo y satisfactorio para los usuarios. Además, destaca por su flexibilidad y enfoque práctico, permitiendo una evolución continua de la arquitectura durante el desarrollo.

Reflexión: Este artículo resalta la importancia de estructurar claramente el proceso de desarrollo ágil en pequeñas empresas, donde los roles se mezclan. Dividir el proceso en etapas definidas como análisis y diseño mejora la eficiencia y facilita su comprensión. Además, enfatiza que la usabilidad debe ser una prioridad desde el principio, ya que es clave para garantizar que los usuarios finales estén satisfechos y se sientan cómodos con la aplicación.

Diagrama, Esquemático

Descripción generada automáticamente

Bibliografía: Arciniegas, José L.; Fernández, Verónica; Hormiga, Amparo; Tulande, Aleyda; Urbano, Fernando A.; Collazos, César A.; 2009