Resúmenes de las Exposiciones

1. Angie Trujillo Gonzales  
   resumen: En la década de 1980, la arquitectura de software era un concepto amplio, pero con el tiempo se ha convertido en un elemento clave para garantizar la escalabilidad y disponibilidad de los proyectos de software.
2. Anyi Casanova  
   TEMA: La edad de oro de la arquitectura de Software

Resumen: La arquitectura de software es un pilar fundamental en el diseño de sistemas eficientes y escalables. Desde los años 80, ha evolucionado de ser una simple descripción del sistema a convertirse en una disciplina esencial que guía el desarrollo de software robusto y funcional. Este enfoque permite adaptarse a los constantes cambios tecnológicos, asegurando diseños sostenibles a mediano y largo plazo. Los arquitectos y herramientas avanzadas desempeñan un rol crucial al garantizar que los sistemas sean mantenibles, escalables y duraderos, logrando que la arquitectura funcione como una maqueta dinámica que responde a las necesidades actuales.

1. Aura Fierro

TEMA: Implementación de una arquitectura guiada por el dominio (DDD)

Resumen: El diseño guiado por el dominio (DDD) busca resolver problemas complejos enfocándose en el conocimiento del negocio durante el diseño del software. Este enfoque promueve la implementación de arquitecturas limpias y hexagonales para gestionar la lógica del negocio de manera más eficiente. La arquitectura limpia organiza el código, mientras que la hexagonal desacopla la lógica del negocio y del sistema, ofreciendo mayor alineación con el negocio, facilidad de mantenimiento y sostenibilidad a largo plazo. Estas metodologías son ideales para la gestión de microservicios, permitiendo una adaptabilidad vertical y horizontal y creando una base sólida para sistemas robustos y escalables.

1. Camilo Bautista  
   TEMA: Arquitectura de software: fundamentos y teoría  
   Resumen: La arquitectura de software es la base para desarrollar sistemas robustos, organizados y sostenibles. Define cómo los componentes de un sistema se estructuran e interactúan, lo que facilita el mantenimiento y la escalabilidad. Una buena arquitectura actúa como un boceto que organiza y orienta el desarrollo, evitando confusiones y desorganización. También abarca conceptos fundamentales y estructurales que son valiosos para la implementación de sistemas complejos, consolidándose como un conjunto de decisiones clave que garantizan un desarrollo eficiente y efectivo.
2. Carolina Martinez  
   TEMA: Arquitectura para herramientas de costos y programación  
   Resumen: Esta herramienta educativa combina teoría y práctica en ingeniería civil, permitiendo simular cómo los costos afectan el desarrollo de proyectos. Basada en metodologías secuenciales y diagramas UML, apoya el aprendizaje autónomo y optimiza la planificación y ejecución de proyectos. Es especialmente útil para estudiantes y profesionales, ya que facilita el análisis de presupuestos y tiempos, ayudando en la toma de decisiones informadas. Al fusionar el conocimiento teórico con aplicaciones prácticas, transforma la educación en el campo de la ingeniería y mejora la comprensión de los factores que impactan los costos en proyectos reales.
3. Cristian Narváez  
   TEMA: Documentación y análisis de los principales frameworks de arquitectura de software  
   Resumen: Los frameworks de arquitectura de software son esenciales en el desarrollo de sistemas empresariales grandes y complejos. Este artículo analiza los principales enfoques, como la arquitectura en capas, cliente-servidor y en tres capas. La arquitectura en capas facilita el mantenimiento y los cambios aislados; la cliente-servidor optimiza la interacción entre usuarios y servidores, mientras que la arquitectura en tres capas divide la presentación, lógica y datos. Estos frameworks son claves en herramientas como ERP (gestión de recursos empresariales) y CRM (gestión de relaciones con clientes), asegurando integración, escalabilidad y productividad en aplicaciones empresariales a largo plazo.
4. Cristian Jeanpool

TEMA: Análisis comparativo entre MVC y MVP para el rendimiento de aplicaciones web

Resumen: Este artículo evalúa los patrones de diseño **MVC** (Modelo-Vista-Controlador) y **MVP** (Modelo-Vista-Presentador) en términos de rendimiento en aplicaciones web. Se realizaron pruebas en un entorno controlado con el mismo escenario, máquina y configuraciones, midiendo factores como tiempos de desarrollo, líneas de código, uso de CPU y tiempos de respuesta. Los resultados indican que **MVC** sobresale frente a **MVP**, destacándose por su mejor rendimiento y eficiencia en la gestión de recursos. Este patrón resulta especialmente útil en aplicaciones que requieren rapidez y estabilidad, mientras que MVP puede ser más adecuado en proyectos con necesidades específicas.

1. Sebastian Penna   
   TEMA: Arquitectura hexagonal

Resumen: La arquitectura hexagonal fomenta la **separación de responsabilidades** al dividir el sistema en componentes independientes que se comunican mediante **puertos** y **adaptadores**. Los puertos actúan como interfaces para conectar la lógica del negocio con el exterior, y los adaptadores integran servicios externos sin modificar la lógica interna del sistema. Esta estructura facilita el mantenimiento y permite adaptar el sistema a nuevas tecnologías sin comprometer su núcleo. Es una metodología que garantiza flexibilidad y sostenibilidad a largo plazo, además de simplificar el desarrollo al gestionar de manera eficiente la complejidad del software.

1. Jose Gasca

TEMA: Modelo y herramienta de software

Resumen: El modelo de software presentado aborda tres perspectivas principales: **conceptual**, **lógica** y **física**, con el propósito de gestionar riesgos en el desarrollo web.

* **Conceptual**: consta de cinco fases (creación del proyecto, parametrización, evaluación, identificación y gestión de riesgos).
* **Lógica**: utiliza diagramas UML para modelar la estructura de clases y relaciones.
* **Física**: se enfoca en la implementación en bases de datos.

Este modelo, acompañado de una herramienta interactiva e incremental, optimiza el proceso de desarrollo al mitigar riesgos, mejorar la calidad del software y garantizar un uso eficiente de los recursos.

1. Julian Fierro  
   TEMA: Arquitectura para una herramienta de patrones de diseño

Resumen: Este enfoque arquitectónico facilita la **reutilización de la lógica interna del software** mediante la implementación de patrones de diseño como **Composite** (estructuración jerárquica de elementos) y **Observer** (sincronización entre componentes). La herramienta gestiona patrones de modelado de forma flexible, promoviendo consistencia y eficiencia en las operaciones. Además, asegura que los cambios en diferentes componentes se sincronicen adecuadamente, optimizando la ejecución y permitiendo una adaptabilidad constante.

1. Laura Ariza  
   TEMA: Mapeo de arquitectura de software  
   Resúmen: El mapeo de arquitectura de software es un proceso que analiza la estructura interna de un proyecto, identificando cómo están organizados y se comunican sus diferentes componentes. Es crucial presentar la arquitectura de forma clara y organizada, ya que una gestión deficiente o una falta de documentación adecuada puede generar problemas en el sistema. En estos casos, se pueden emplear herramientas especializadas para recuperar y documentar información, proponiendo soluciones que optimicen la estructura y mejoren la eficiencia del proyecto.
2. Mar Artunduaga  
   TEMA: Arquitectura de microservicios para desarrollo web  
   Resúmen: Originalmente, un programa desarrollado por la Asamblea seguía una estructura monolítica, lo que complicaba su mantenimiento y escalabilidad. Para superar estas limitaciones, se decidió adoptar una **arquitectura de microservicios**, la cual divide la aplicación en servicios autónomos que pueden desarrollarse, implementarse y escalarse de manera independiente. Este enfoque mejora la adaptabilidad, facilita la gestión del sistema y permite que cada servicio opere de forma eficiente, optimizando el desarrollo web y garantizando un sistema más flexible y escalable.
3. Maria Murcia  
   TEMA: Patrones de usabilidad de la arquitectura de software  
   Resúmen: Los patrones de usabilidad en arquitectura de software están diseñados para **mejorar la experiencia del usuario** al interactuar con aplicaciones y sistemas. Por ejemplo, el proyecto **STATUS** incorpora funciones como "deshacer" y "cancelar" para mejorar la accesibilidad y eficiencia. Al aplicar estos patrones, se optimiza la interacción del usuario y se garantiza que el software sea intuitivo y fácil de usar, aumentando la satisfacción del usuario final.
4. Mariana Charry  
   TEMA: Arquitectura de software para entornos móviles

Resúmen: La arquitectura de software para dispositivos móviles tiene como objetivo **estandarizar metodologías** de desarrollo, lo que permite la creación de aplicaciones más robustas, escalables y eficientes. Con la evolución de la tecnología móvil, se han desarrollado aplicaciones más complejas y de mayor tamaño, basadas en arquitecturas sólidas que aseguran alta calidad y un rendimiento óptimo, mejorando la experiencia del usuario en dispositivos móviles.

1. Mariana Calderon  
   TEMA: Marco de trabajo para seleccionar un patrón arquitectonico  
   Resúmen: Un marco de trabajo para la selección de patrones arquitectónicos ayuda a los desarrolladores a **elegir el diseño más adecuado** según los requisitos específicos del proyecto. Este enfoque analiza patrones como **MVC**, **MVP**, **microservicios** y arquitecturas **en la nube**, considerando aspectos como funcionalidad, limitaciones técnicas y tipo de aplicación. La selección adecuada optimiza la escalabilidad, eficiencia y adaptabilidad del sistema.
2. Maydy Conde  
   TEMA: Desarrollo de aplicaciones web usando el modelo Vista-Controlador  
   Resumen: El patrón **MVC (Modelo-Vista-Controlador)** es una herramienta efectiva para desarrollar software interactivo. Este modelo separa la lógica del negocio de la interfaz, lo que facilita las pruebas y la implementación en entornos cliente-servidor. Aunque es más complejo para aplicaciones web, su estructura modular permite crear sistemas más organizados y sostenibles, adecuados para el desarrollo escalable.
3. Mayra Tamayo  
   TEMA: Análisis comparativo de patrones de diseño  
   Resumen: Este análisis compara las **ventajas y desventajas** de diferentes patrones de diseño, como **MVC**, **MVP**, **MVVM**, y el **Controlador Frontal**, en las distintas etapas del desarrollo de software. Cada patrón tiene un propósito específico y no existe uno superior en todos los casos. La elección depende de las necesidades del proyecto, permitiendo mejoras prácticas sostenibles y garantizando eficiencia, escalabilidad y mantenimiento.

Ejemplo:

**Método de plantilla**: Define una plantilla base en un algoritmo dejando detalles específicos para la implementación.

**MVC**: Separa la lógica de negocio de la interfaz de usuario.

**MVP**: Desacopla la lógica de presentación de la vista, optimizando pruebas y mantenimiento.

**MVVM**: Ideal para aplicaciones dinámicas con enlaces de datos bidireccionales.

**Controlador frontal**: Centraliza todas las soluciones en un único controlador.

1. Patricia Sarmiento, mi tema
2. Maryuri bonilla gonzales

Tema: Monolito vs Microservicios en arquitectura de software

Resume: analisa las diferencias entre las arquitecturas monoliticas y microservicios

1. Johan Calderón

La arquitectura de software organiza sistemas a un alto nivel definiendo elementos y sus relaciones. Simplifique el desarrollo reutilizando componentes utilizando arquitecturas de referencia. Se desarrolla utilizando métodos, modelos y escenarios estructurales para guiar el diseño y la comprensión de sistemas complejos.

1. **Andrés Pantoja:**

Este estudio analiza dos variantes del patrón MVC para desarrollo web, evaluando su impacto en escalabilidad y tiempos de desarrollo. También examina los patrones de tubos y filtros, concluyendo que los tubos son más eficaces, aunque requieren más investigación para su implementación.

1. **Nikoll:**

La arquitectura propuesta divide el control del robot en tres niveles:

**Nivel básico:** Maneja motores, sensores y acceso al hardware.

**Nivel medio:** Ofrece bibliotecas que facilitan el desarrollo de control y conectan hardware y software.

**Nivel avanzado:** Integra una interfaz con un panel de control y un simulador 3D para programar y seguir al robot en entornos virtuales.  
Desarrollada en C#, la arquitectura es modular, extensible y efectiva en la integración de capas, simplificando el desarrollo y reduciendo riesgos mediante simulaciones previas.

1. **Juan Cerquera:**   
   Se evaluaron las arquitecturas más usadas junto con desarrolladores:

**Arquitectura en la nube:** Ofrece seguridad y flexibilidad, ideal para aplicaciones web.

**MVC:** Destaca en mantenimiento y rendimiento, usado en aplicaciones móviles, de escritorio y web.

**Microservicios:** Promueven seguridad y escalabilidad, enfocados en aplicaciones web.

**MVP:** Proporciona flexibilidad y modularidad, empleado en aplicaciones móviles y web.

1. **Erick:**

Los microservicios son una arquitectura que supera las limitaciones de los sistemas monolíticos, permitiendo que los componentes operen de forma independiente. Esto optimiza el desarrollo y el mantenimiento, proporcionando soluciones escalables y eficientes.

1. **Marlon:**   
   El patrón Flyweight minimiza el uso de memoria compartiendo datos entre objetos similares. Es útil en sistemas como videojuegos y aplicaciones gráficas, donde la eficiencia de memoria es fundamental.