

Universidad Católica Boliviana “San Pablo”  
Facultad de Ingeniería  
La Paz - Bolivia

SEMESTRE 2-2024

PREPARACIÓN Y EVALUACIÓN DE PROYECTOS DE INGENIERÍA

IND-330

Sistema de Alerta Temprana para incendios en exteriores en zonas rurales alejadas

Realizado por:

-Johan Marco Quispe Rodríguez  
-Miguel Barrionuevo  
-Juan José Valda  
-Gerson Cano

1. INTRODUCCIÓN

La arquitectura de software es más que un plano técnico; es la base que define cómo un sistema debe estructurarse para cumplir con sus objetivos y responder a los desafíos del mundo real. Desde gestionar la complejidad hasta asegurar escalabilidad y mantenimiento, la arquitectura conecta las necesidades del negocio con las soluciones tecnológicas. Según Bass, Clements y Kazman en Software Architecture in Practice, es clave para construir sistemas exitosos, no solo técnicamente, sino también estratégicamente. Este documento explora conceptos fundamentales, desde modelos arquitectónicos hasta patrones modernos como microservicios, para diseñar sistemas más adaptables y robustos.

1. OBJETIVO

El objetivo principal es comprender cómo los principios de arquitectura de software ayudan a diseñar sistemas flexibles, escalables y fáciles de mantener. Inspirándonos en autores como Eric Evans y Sam Newman, este análisis busca explicar técnicas y patrones arquitectónicos que faciliten la descomposición de sistemas complejos en soluciones más manejables. Esto permitirá al lector visualizar y aplicar conceptos como el diseño basado en el dominio o la arquitectura hexagonal en problemas reales.

1. ARQUITECTURA DE SOFTWARE

La arquitectura de software define la estructura organizacional de un sistema, incluyendo sus componentes, relaciones y principios de diseño. Según Bass et al., es el arte de tomar decisiones estratégicas que afectan el éxito del software a largo plazo. Este apartado introduce el tema como la combinación de decisiones técnicas y de negocio, esenciales para crear sistemas que respondan al cambio continuo.

* 1. PROCESO DE ARQUITECTURA DE SOFTWARE

Diseñar una arquitectura no es solo elegir tecnologías; es un proceso iterativo que conecta requisitos del negocio con soluciones técnicas. En Software Architecture in Practice, se describe cómo este proceso implica identificar prioridades, evaluar alternativas y tomar decisiones informadas para asegurar calidad y viabilidad. Este punto profundiza en cómo estructurar ese proceso de manera efectiva.

* 1. MODELOS Y VISTAS ARQUITECTURALES

Para entender un sistema complejo, necesitamos perspectivas claras. Bass et al. presentan los modelos y vistas arquitecturales como herramientas esenciales para representar diferentes ángulos de un sistema: desde lo técnico hasta lo funcional. Aquí exploramos cómo estas representaciones ayudan a comunicar ideas y detectar problemas antes de que aparezcan.

* 1. DISEÑO BASADO EN EL DOMINIO (DDD)

Eric Evans, en Domain-Driven Design, argumenta que para resolver problemas técnicos de forma eficiente, primero debemos entender profundamente el negocio. DDD se trata de centrar el diseño del software en el "lenguaje" y las necesidades del dominio, conectando equipos técnicos y no técnicos en una visión compartida. Este enfoque es esencial para sistemas complejos y en evolución.

* 1. ARQUITECTURA HEXAGONAL

La arquitectura hexagonal, introducida por Alistair Cockburn y popularizada por autores como Robert C. Martin, propone una estructura que conecta el núcleo del sistema (lógica de negocio) con el exterior a través de interfaces claras. En Clean Architecture, se explica cómo este enfoque facilita pruebas y reduce dependencias, creando sistemas más modulares y flexibles.

* 1. MICROSERVICIOS

Los microservicios no son solo una moda; representan una evolución en cómo pensamos y diseñamos software. Sam Newman, en Building Microservices, los describe como pequeñas aplicaciones independientes que trabajan juntas como un sistema más grande. Este patrón permite a los equipos moverse rápido, escalar componentes de forma independiente y adaptarse al cambio con mayor agilidad.

* 1. ESTRATEGIAS DE DESCOMPOSICIÓN

Migrar a microservicios no es un proceso automático; requiere estrategias inteligentes para dividir una aplicación monolítica en servicios manejables. Chris Richardson, en Microservices Patterns, ofrece guías prácticas para identificar límites contextuales y dividir funcionalidad de forma que se mantenga coherencia y rendimiento.

* 1. TOPOLOGÍAS

La manera en que organizamos equipos y sistemas afecta directamente la efectividad de nuestra arquitectura. En Building Microservices, Newman explica cómo las topologías definen quién es responsable de qué, asegurando que equipos y servicios puedan escalar juntos. Aquí exploraremos las estructuras más comunes y sus beneficios.

1. EJEMPLO ESQUEMÁTICO DE UN SISTEMA BASADO EN MICROSERVICIOS
2. CONCLUSIONES

El viaje por la arquitectura de software revela que no se trata solo de tecnología, sino de decisiones estratégicas que impactan directamente en el éxito de un sistema. Reflexionaremos sobre cómo los conceptos aprendidos pueden ser aplicados para construir software más robusto, alineado con las necesidades del negocio y listo para los desafíos del futuro.

1. BIBLIOGRAFÍA