

**Universidad Tecnológica de Tehuacán**

**Tecnologías de la Información**

**Ingeniaría en Gestión y Mantenimiento de Software**

**Actividad 4: De la Arquitectura al Prototipo Funcional**

**ADR Actualizados**

**Equipo:**

**Alan Baltazar Figueroa**

**Alejandro Contreras Martínez**

**Emmanuel Castro Salvador**

**Ciro Julián Cervantes Zamora**

**Manelic Alitzel Reyes Torres**

**Danna Paola Tobón Mosqueda**

**Kevin Armando Montalvo Marcial**

**Materia:**

**Arquitectura de Software**

**Docente:**

**José Miguel Carrera Pacheco**

**28 de septiembre de 2025, San Pablo Tepetzingo, Tehuacán, Puebla**

**ELECCIÓN DE ARQUITECTURA MONOLÍTICA**

Se optó por una arquitectura monolítica sobre microservicios porque el desarrollo resulta tres a cuatro veces más rápido, permitiendo tener un sistema funcional en semanas en lugar de meses. Con esto, un solo equipo puede manejar todo el proyecto, reduciendo los costos de infraestructura hasta en un 70% y ofreciendo una curva de aprendizaje mínima para los desarrolladores. El alto acoplamiento entre componentes se considera manejable para el tamaño actual del proyecto y su equipo de desarrollo.

**BASE DE DATOS ÚNICA VS DISTRIBUIDA**

Se decidió implementar una base de datos única utilizando SQLite para desarrollo y PostgreSQL para producción, en lugar de optar por bases de datos distribuidas, porque esta aproximación simplifica significativamente las consultas que requieren cruzar información entre usuarios, pagos e inscripciones, haciendo que sean más rápidas y menos propensas a errores. Las operaciones de backup y recuperación se vuelven más directas y se garantiza la consistencia de los datos en todas las transacciones, lo que es crucial para un sistema de pagos. Aunque existe el riesgo potencial de un cuello de botella bajo cargas extremadamente altas y hay menos aislamiento entre los diferentes tipos de datos, estos riesgos se consideran bajos para el volumen de transacciones esperado en la universidad, y la simplicidad operativa resultante justifica plenamente la decisión.

**FRONTEND ÚNICO VS MICRO-FRONTENDS**

La elección de desarrollar una aplicación React única, en lugar de seguir una arquitectura de micro-frontends, se basó en la necesidad de maximizar la reutilización de código entre las interfaces de estudiante y administrador, creando un proceso de desarrollo más eficiente. Esto permite un solo proceso de construcción y despliegue, lo que acelera los ciclos de release y reduce la duplicación de componentes comunes como navegación y autenticación.

**DESPLIEGUE SIMPLE VS KUBERNETES**

Se seleccionó una estrategia de despliegue simple utilizando Node.js junto con la base de datos, renunciando al uso de Kubernetes, porque esta aproximación requiere una configuración de infraestructura mínima y reduce drásticamente los costos de hosting y la necesidad de contar con especialistas en DevOps. Los despliegues son más rápidos, directos y comprensibles para el equipo completo, lo que acelera la resolución de incidencias.

**COMUNICACIÓN DIRECTA VS API GATEWAY**

Se decidió que el frontend se comunique directamente con el backend sin la intermediación de un API Gateway, ya que esta arquitectura elimina un punto potencial de falla y reduce la latencia en las comunicaciones, haciendo que la aplicación sea más rápida y responsiva para el usuario final. La configuración resulta mucho más simple y el debugging de problemas se vuelve más directo, al poder trazar las peticiones de principio a fin sin componentes intermedios.