Ciclo de vida del desarrollo de sistemas

Análisis y diseño de sistemas - Universidad Pública de El Alto

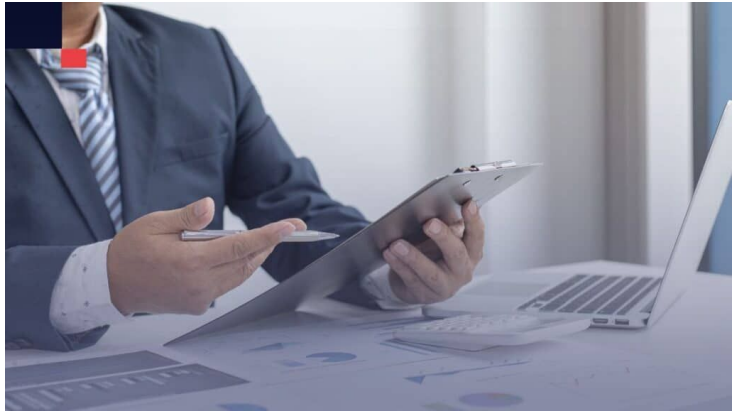
Etapas del análisis y desarrollo:

- Estudio de factibilidad: Técnico, Económico, operativo
- Gestión de requerimientos
- Diseño: Arquitectura, componentes y datos.
- Implementación y pruebas
- Mantenimiento y evolución

Estudio de factibilidad: técnico

Dimensiones:

- Tecnología requerida
- Conocimientos y competencia del equipo
- Compatibilidad de la infraestructura
- Escalabilidad y Futura expansión
- Cuestiones regulatorias y de cumplimiento



Estudio de factibilidad: económica

Dimensiones:

- Costo de oportunidad
- Costos directos e indirectos
- Costo de transición
- Costo de licencia y cumplimiento
- Modelos de precios y tarifas
- Retorno de Inversión(ROI)

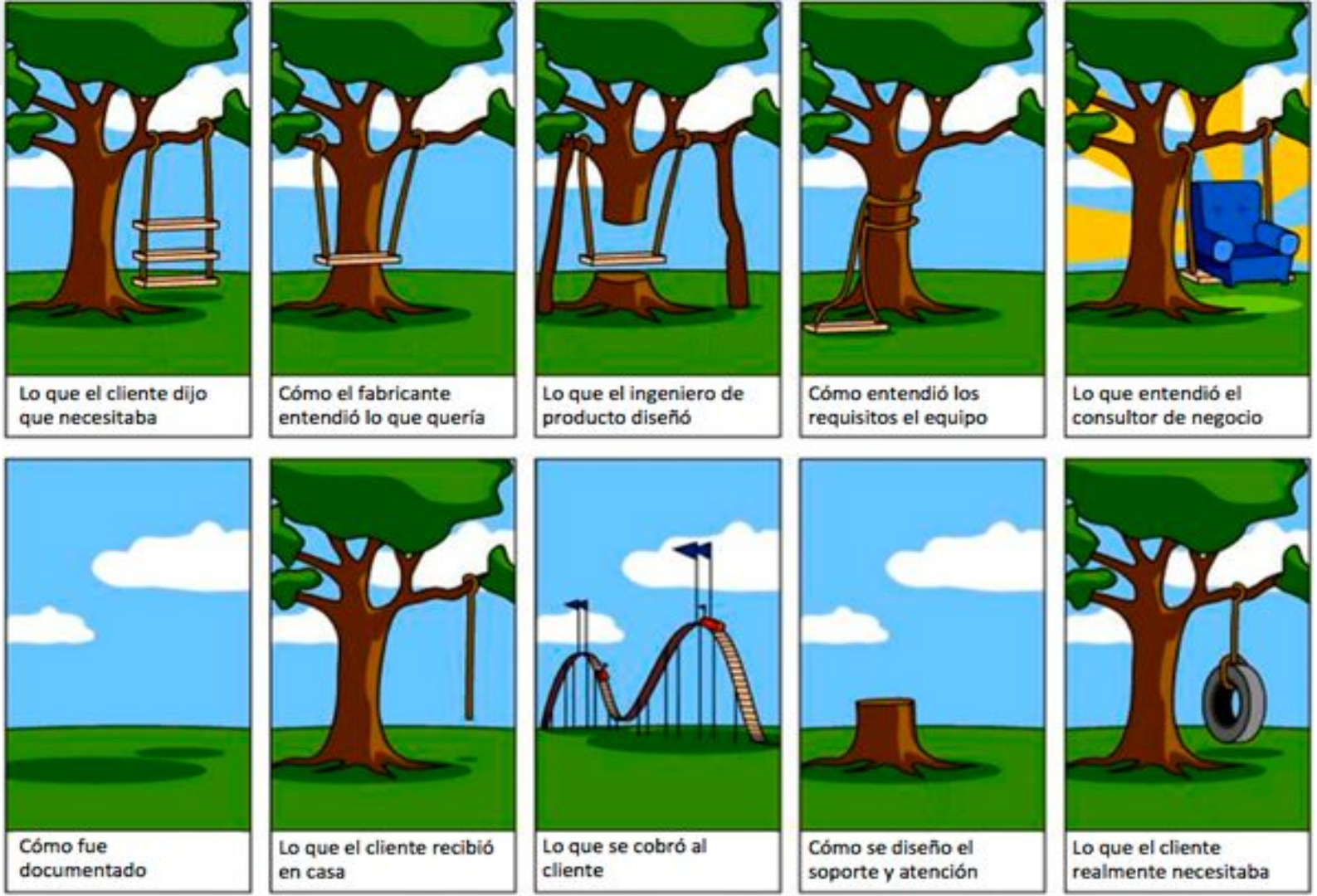
Estudio de factibilidad: Operativa

Dimensiones:

- Adopción del usuario
- Capacitación y desarrollo
- Flujo de trabajo y procesos de negocio
- Cambio organizacional
- Riesgos y mitigaciones
- Retorno de Inversión(ROI)

Gestión de requerimientos

Establece las bases para lo que será el proyecto final, los errores o malentendidos en esta fase pueden llevar a retrasos costosos, insatisfacción del cliente o hasta el fracaso


Lo que el cliente dijo que necesitaba
Cómo el fabricante entendió lo que quería
Lo que el ingeniero de producto diseñó
Cómo entendió los requisitos el equipo
Lo que entendió el consultor de negocio
Cómo fue documentado
Lo que el cliente recibió en casa
Lo que se cobró al cliente
Cómo se diseño el soporte y atención
Lo que el cliente realmente necesitaba

Gestión de requerimientos: Recopilación

- Entrevistas estructuradas y No estructuradas
- Talleres de requerimientos
- Observación directa y etnográfica
- Análisis de documentos y artefactos
- Prototipado
- Historias de usuario y casos de uso
- Técnicas de ingeniería de requerimientos (UML)
- Documentación

Gestión de requerimientos: Análisis

- Tipos de requerimientos (Funciona y no Funcional)
- Modelado y especificación (UML, C4)
- Priorización y negociación (MOSCOW Must have, Should have, Could have, Won't have)
- Validación y Verificación
- Documentación
- Revisiones formales

Gestión de requerimientos: Documentación SRS

- Formato y Estructura
- Claridad y precisión
- Detallando contexto y condiciones
- Casos de uso y escenarios
- Rastreabilidad
- Versionado

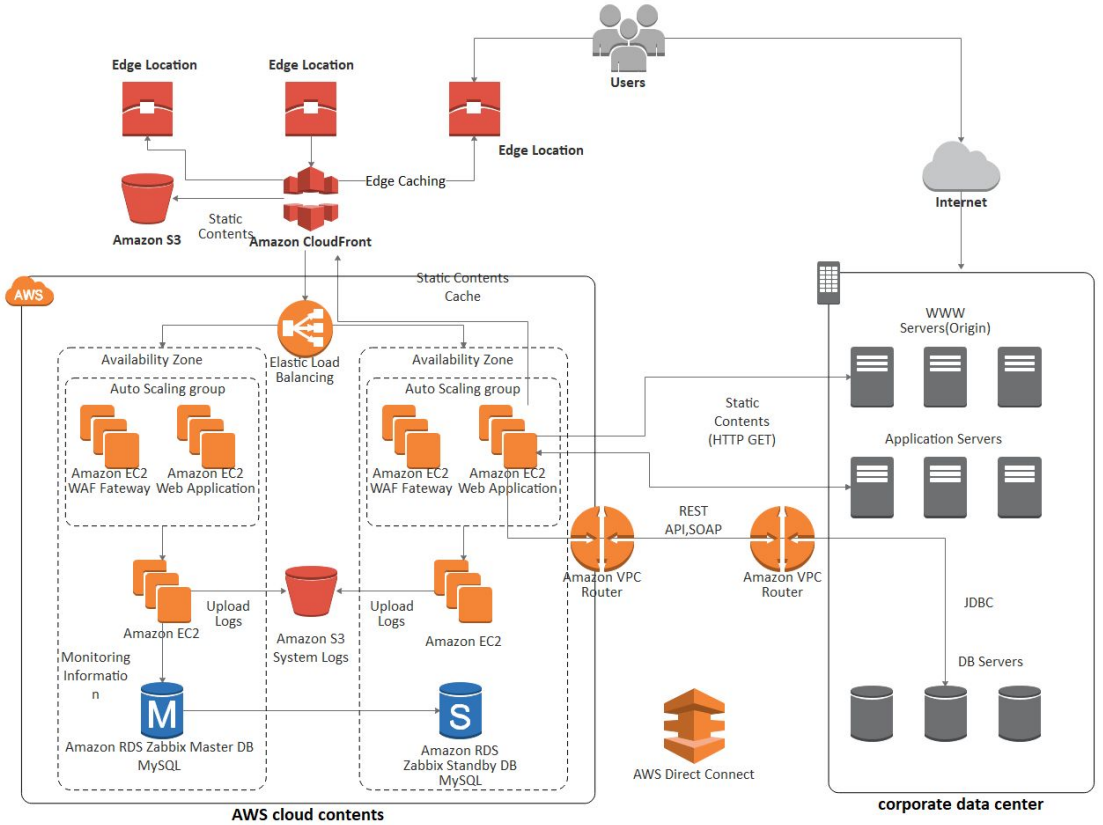
Gestión de requerimientos: Validación

- Revisiones por Pares
- Retroalimentación de stakeholders
- Modelado y simulación
- Validación Formal
- Verificación cruzada
- Pruebas de aceptación de usuario
- Documentación de resultados de validación

Gestión de requerimientos: Herramientas

- Colaboración en tiempo real
- Trazabilidad
- Integración con otras herramientas
- Plantillas y reutilización
- Gestión de cambios
- Reportes y análisis
- Seguridad y cumplimiento

Diseño:
Arquitectura, Componentes Y Datos



Diseño: Arquitectura

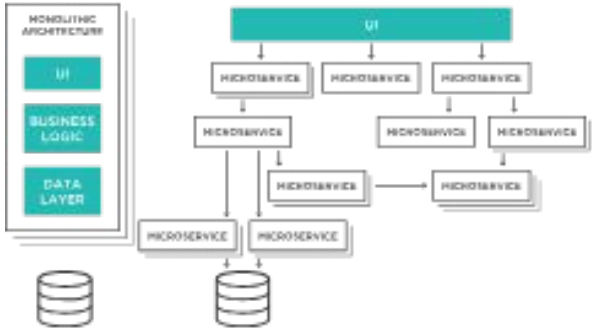
División del sistema: cómo se dividirá el sistema, en subsistemas o módulos?

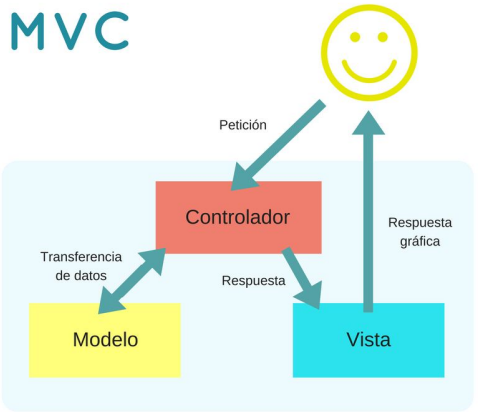
Interacción de los componentes: Como se comunicaran entre sí los distintos componentes o módulos?

Escalabilidad y Extensibilidad: El diseño permite un crecimiento y adaptación futura del sistema?

Diseño: Arquitectura,
Elección del patrón arquitectónico

- Modelo Vista Controlador
- Modelo Vista - Vista de Modelo
- Microservicios





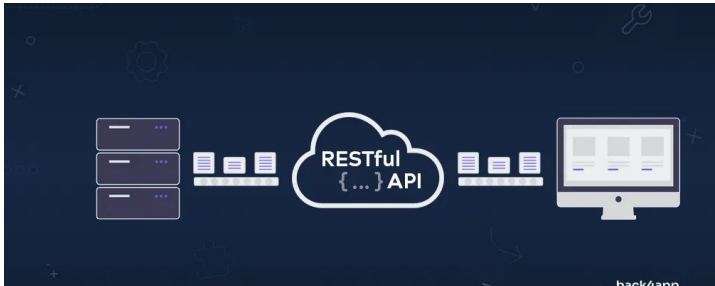
Diseño: Arquitectura,

División del sistema en subsistemas o módulos

- Por Funcionalidad
- Por Dominio

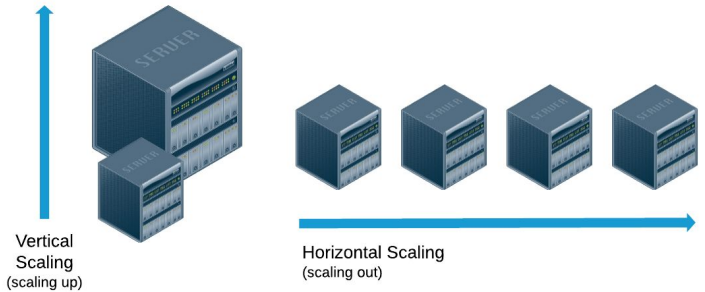
Diseño: Arquitectura,
Interacción entre los componentes

- API RESTful
- Cola de mensajes
- WebSockets
- UDP



Diseño: Arquitectura,
Escalabilidad y extensibilidad

- Horizontal vs vertical
- Desacoplamiento (SPOF)



Diseño: Componentes

- **Funciones y Operaciones**
- **Interfaces,** Comunicación entre componentes
- **Dependencias**

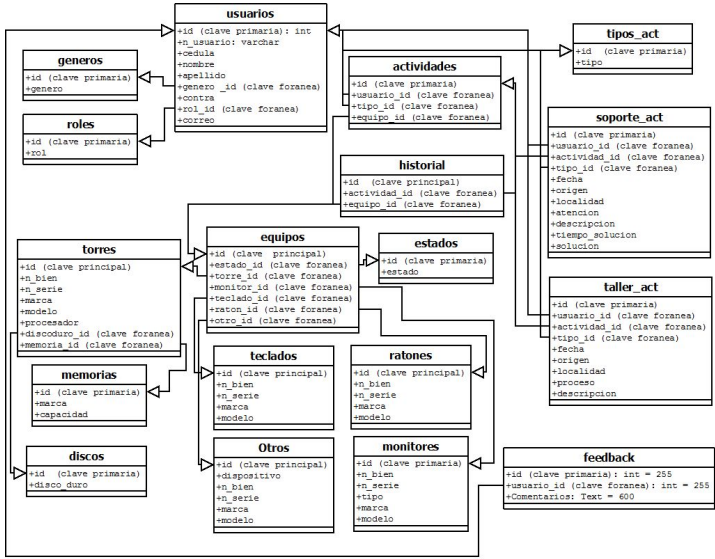
Diseño: Componentes

- Funciones y Operaciones
- Encapsulamiento
- Cohesión
- Interfaces
- APIs (Interfaces de programación de aplicaciones)
- Dependencias
- Inyección de Dependencias
- Gestión de paquetes

Diseño: Datos

En esta etapa se definen las estructuras de datos, cómo se almacenarán, y cómo se accederá a ellos. Aspectos importantes a considerar son:

- **Esquemas de la Base de Datos:** Diseño de las tablas, relaciones y restricciones.
- **Políticas de Acceso:** ¿Quién puede acceder a qué datos y cómo?
- **Optimización de Consultas:** ¿Cómo se realizarán las consultas para ser eficientes en tiempo y recursos?



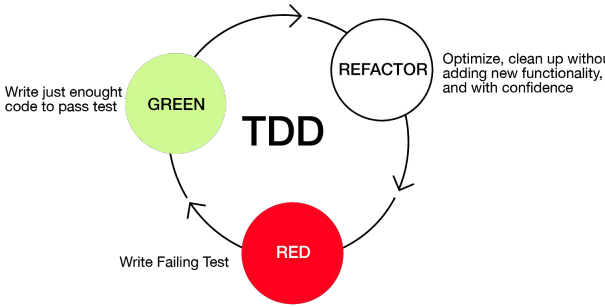
Implementación y pruebas

La etapa de implementación y pruebas es donde las especificaciones y diseños se traducen en un sistema funcional. Este paso es fundamental para la realización exitosa del proyecto, y es aquí donde se empieza a ver la concreción del trabajo previo

Implementación y pruebas: Calidad de código

Refactorización: Este es el proceso de reorganizar y limpiar el código sin cambiar su comportamiento. Es como reorganizar un armario para que todo sea más fácil de encontrar.

Comentarios: Los comentarios en el código son como las notas al margen en un libro, ayudan a entender qué es lo que está haciendo una sección particular del código.



Implementación y pruebas: Revisiones de código

Pull Request/Merge Request: En plataformas como GitHub o GitLab, este es un mecanismo para proponer cambios en el código y discutirlos con otros desarrolladores antes de integrarlos al proyecto

Implementación y pruebas: control de versiones

Git: Este es un sistema de control de versiones que permite realizar un seguimiento de cambios en el código fuente durante el desarrollo del software. Puede imaginarlo como un sistema de "guardado múltiple" para su proyecto.

Branching y Merging: Estos son métodos para trabajar en diferentes versiones de un proyecto simultáneamente. Imagine que está escribiendo un libro y quiere probar diferentes finales; cada final sería una "rama" diferente.

Implementación y pruebas: Tipos de pruebas

Pruebas Unitarias: Estas son como los exámenes médicos para cada órgano del cuerpo; se centran en unidades individuales (o "componentes") del código para asegurarse de que funcionan correctamente en aislamiento.

Pruebas de Integración: Imagina que cada músico en una orquesta toca perfectamente su instrumento individualmente. Las pruebas de integración son como ensayar juntos para asegurarse de que todo suena bien en conjunto.

Pruebas de Aceptación: Estas son las "previsualizaciones" para un público seleccionado antes de la apertura oficial de una obra de teatro. Buscan asegurarse de que el sistema en su conjunto satisface los criterios de aceptación y las necesidades del usuario.

Mantenimiento

Corrección de Errores: Es inevitable que se encuentren errores después del lanzamiento del software. Podrían ser pequeños errores ortográficos en la interfaz de usuario o problemas más críticos que afectan la funcionalidad.

Optimización del Código: Después del lanzamiento, el rendimiento del sistema puede necesitar ser mejorado para manejar una mayor carga o para hacerlo más eficiente.

Actualizaciones: Los sistemas operativos y las bibliotecas de terceros sobre las que depende su software recibirán actualizaciones. Su sistema debe mantenerse actualizado para asegurarse de que sigue siendo compatible.

Evolución

Agregar Nuevas Características: Con el tiempo, los requisitos del negocio o las necesidades de los usuarios pueden cambiar, lo que podría requerir nuevas funcionalidades.

Mejoras en el Sistema: No todas las actualizaciones son acerca de añadir nuevas características. A veces, es sobre la mejora de las existentes para que sean más eficientes o más fáciles de usar.

Escalabilidad: A medida que su software gana más usuarios o tiene que manejar más datos, puede necesitar ser escalado.