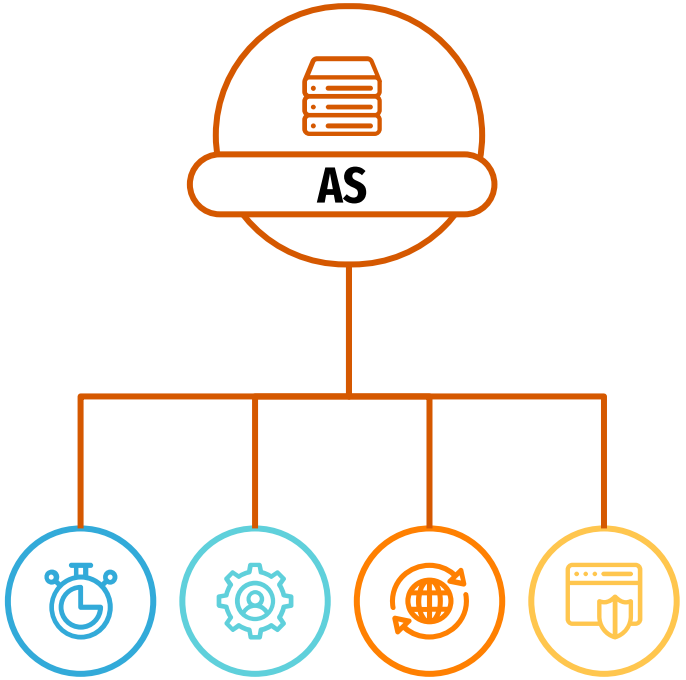
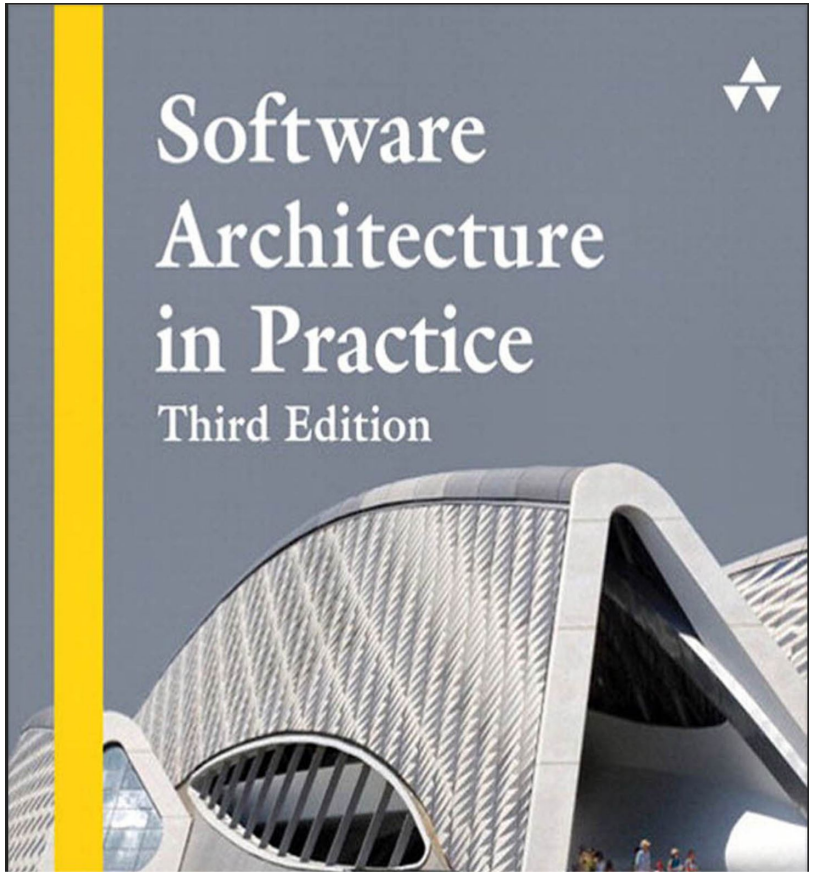
Arquitectura de Software

MSc. Marco Tulio Aldana Prillwitz



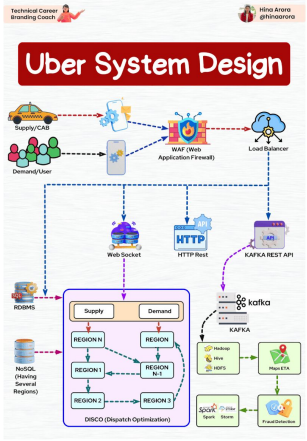
FIUSAC
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Arquitectura de Software



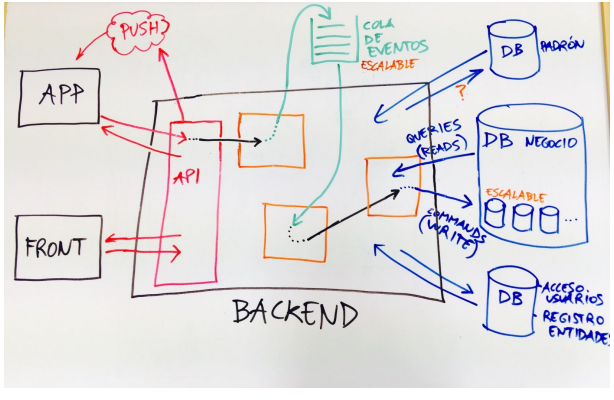
Arquitectura de Software

La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. En el libro "An introduction to Software Architecture", David Garlan y Mary Shaw definen que la arquitectura es un nivel de diseño que hace foco en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema".



Arquitectura de Software

- El concepto surge ya en los años 60 y se refiere a una planificación basada en modelos, patrones y abstracciones teóricas, a la hora de realizar una pieza de software de cierta complejidad y como paso previo a cualquier desarrollo o implementación.
- De esta forma disponemos de una guía teórica detallada que nos permite entender cómo van a encajar cada una de las piezas de nuestro producto o servicio.



Arquitectura de Software

Toda arquitectura de software debe describir diversos aspectos del software. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Es importante destacar que cada uno de ellos constituye una descripción parcial de una misma arquitectura y es deseable que exista cierto solapamiento entre ellos. Esto es así porque todas las vistas deben ser coherentes entre sí, evidente dado que describen la misma cosa.

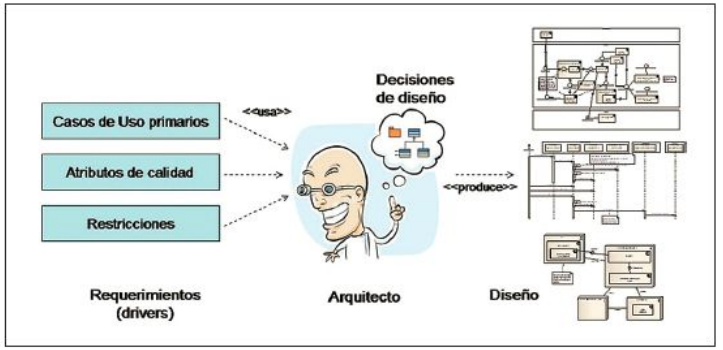


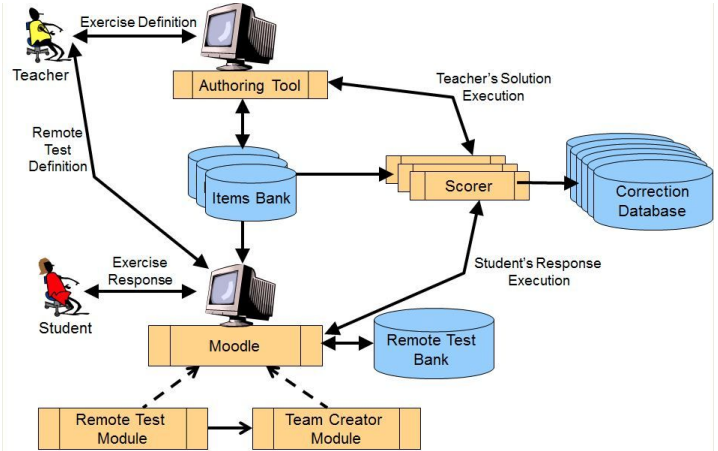
Figura 1: Durante el diseño de la arquitectura, el arquitecto toma como entrada los requerimientos que influyen la arquitectura (drivers) y produce un diseño arquitectónico.

¿Qué es Arquitectura de Software?

- Consiste en un conjunto de patrones y abstracciones coherentes que proporcionan un marco definido y claro para interactuar con el código del software.
- También se le denomina arquitectura lógica.
- Modela los elementos de software y sus propiedades visibles, es considerada un conjunto de decisiones principales de diseño para llegar a un resultado de calidad del sistema.
- La arquitectura se desarrolla a partir de un equipo autogestionado o de un arquitecto externo.

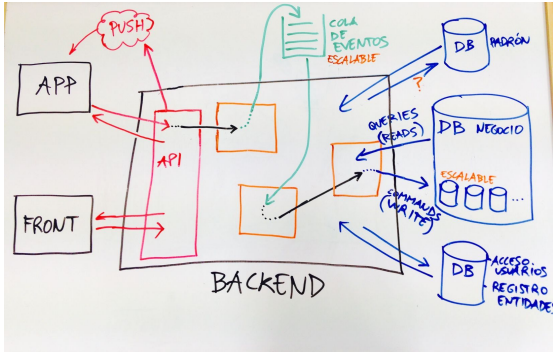
Arquitectura de Software

- Trata sobre estructuras, modelos y diagramas de comunicación de diferentes sistemas incluso entre diferentes módulos del sistema.
- Válida como la arquitectura de software está involucrada en cada uno de los pasos de un proceso de desarrollo de software.
- Describe el rol del arquitecto y como este rol puede ayudar en el éxito o fracaso de un desarrollo de sistema.



Importancia de Arquitectura de Software

- Nos permite planificar y elegir el mejor conjunto de herramientas para el desarrollo de nuestros artefactos de software.
  - Costo
  - Tiempo de desarrollo
  - Usuarios
  - Nivel de integración



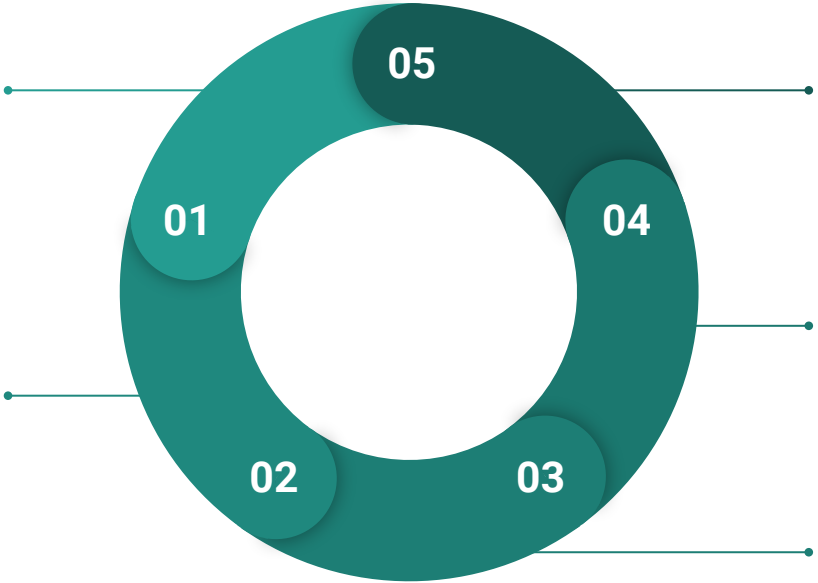
Etapas de la Arquitectura de Software

1. Análisis de requerimientos

Surge de una necesidad de crear un artefacto o un sistema.

2. Diseño de la solución

Análisis detallado de los problemas para trabajar en conjunto y plantear posibles soluciones.Tiene como resultado el detalle de la solución.



5. Mantenimiento y evolución

Se realiza un proceso de mejora continua hasta que el artefacto de software deje de ser necesario.

4. Despliegue

Implementación de la solución en la infraestructura. Se coloca el artefacto en disponibilidad a los usuarios.

3. Desarrollo y evaluación

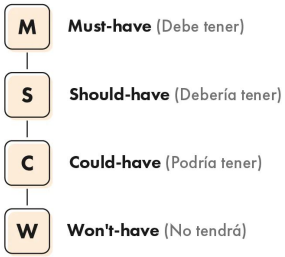
Implementación de la solución. Al finalizar esta etapa tendremos el artefacto de software.

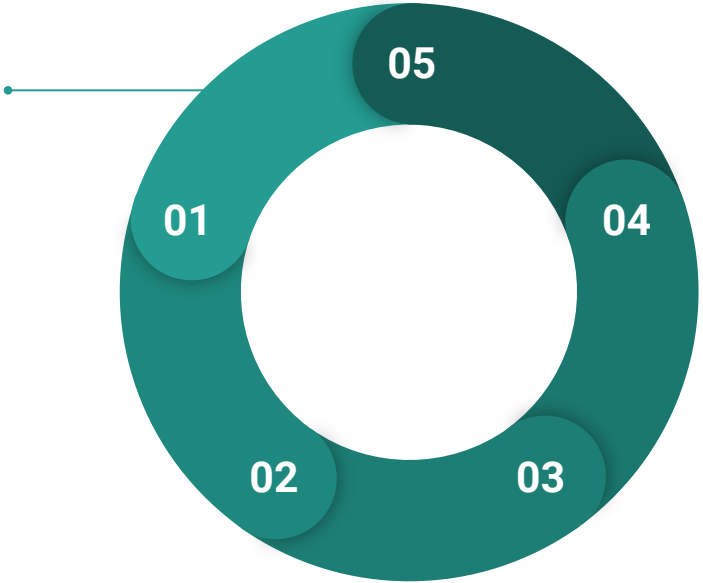
Todo nace de un disparador que puede ser un requerimiento o una necesidad de negocio

Análisis de requerimientos

1. Análisis de requerimientos

Surge de una necesidad de crear un artefacto o un sistema.



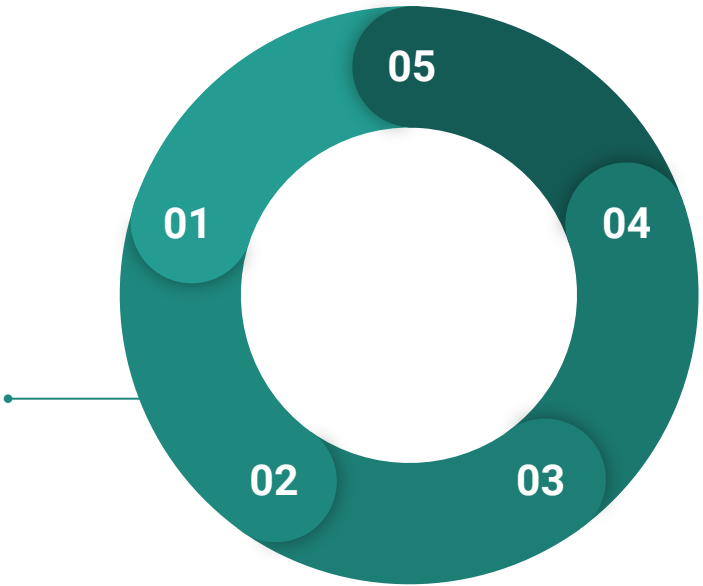


- 1. Requerimientos de negocio
- 2. Requisitos funcionales
- 3. Requisitos no funcionales: rendimiento, fiabilidad, seguridad, usabilidad, mantenibilidad
- 4. Restricciones técnicas: plataformas, compatibilidad
- 5. Escenarios y casos de uso: identificación de actores y acciones
- 6. Consideraciones de escalabilidad: previsión de crecimiento, optimización de recursos.
- 7. Rastreabilidad de requisitos: Vincular cada requisito con su origen, actor o necesidad
- 8 Validar Moscow y Smart

Diseño de la solución

2. Diseño de la solución

Análisis detallado de los problemas para trabajar en conjunto y plantear posibles soluciones.Tiene como resultado el detalle de la solución.

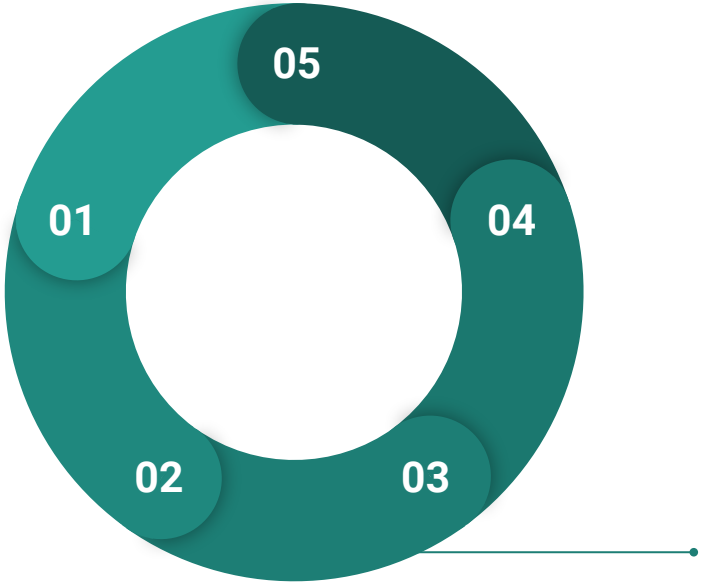


- 1. Diseño de componentes
- 2. Modelado de datos
- 3. Arquitectura de sistema
- 4. Gestión de dependencias
- 5. Diseño de interfaces de Usuario (UI)
- 6. Gestión de errores y excepciones
- 7. Seguridad
- 8. Rendimiento y escalabilidad
- 9. Pruebas y verificación
- 10. Documentación

Desarrollo y evaluación

Entrada importante : Criterios de aceptación

- 1. Implementación componentes
- 2. Prácticas de codificación
- 3. Integración continua
    - 3.1 Desarrollo incremental
    - 3.2 Automatización de construcción
- 4. Documentación de código
- 5. Gestión de configuraciones
- 6. Pruebas unitarias
- 7. Pruebas de integración
- 8. Pruebas de sistemas
- 9. Pruebas de rendimiento
- 10. Pruebas de seguridad
- 11. Evaluación continua
- 12. Ajustes interativos

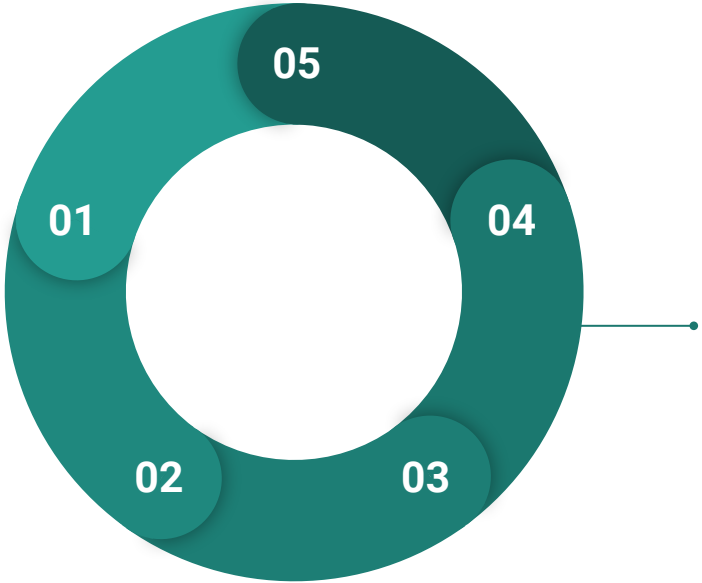


3. Desarrollo y evaluación

Implementación de la solución. Al finalizar esta etapa tendremos el artefacto de software.

Despliegue

- 1. Planificación del despliegue
- 2. Preparación del entorno de producción
- 3. Empaquetado del software
- 4. Proceso de implementación
- 5. Pruebas post-despliegue
- 6. Actualización de bases de datos
- 7. Comunicación de usuarios
- 8. Optimización y ajustes post-despliegue
- 9. Respaldo y recuperación
- 10. Seguimiento continuo

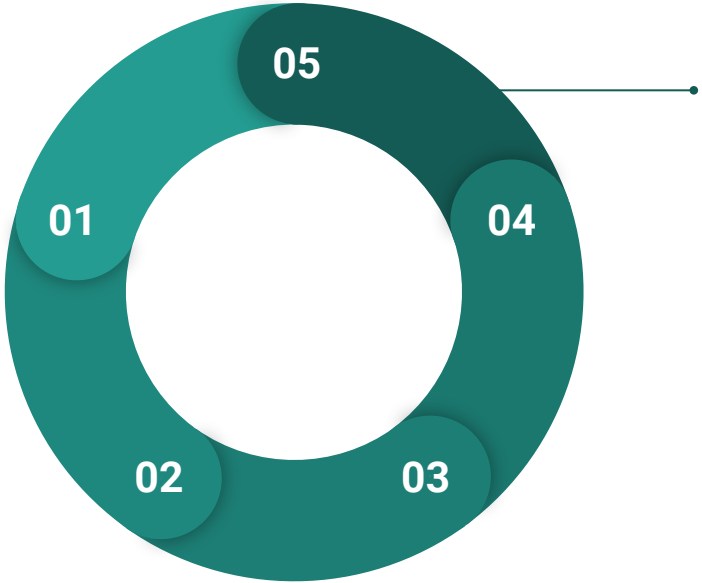


4. Despliegue

Implementación de la solución en la infraestructura. Se coloca el artefacto en disponibilidad a los usuarios.

Mantenimiento y evolución

- 1. Gestión de cambios
- 2. Corrección de errores
- 3. Mejora continua
- 4. Actualización de tecnologías
- 5. Gestión de versiones
- 6. Adaptabilidad
- 7. Seguridad
- 8. Documentación actualizada
    - Arquitectura
    - Procesos
- 9. Gestión de configuración
- 10. Evaluación del desempeño



5. Mantenimiento y evolución

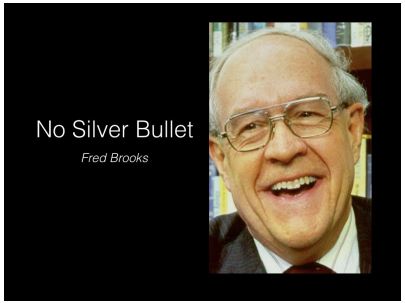
Se realiza un proceso de mejora continua hasta que el artefacto de software deje de ser necesario.

Salida importante : Sistema deprecado, ciclo de vida cumplido

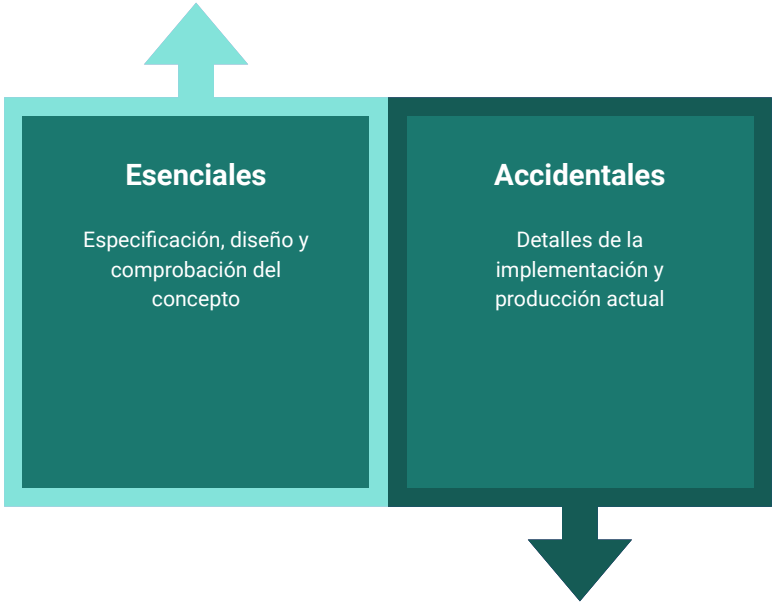
MVP : Minimum Viable Product

- Popularizado por Eric Ries en su libro "The Lean Startup".
- Es la versión más simple de un producto que aún cumple con los requisitos básicos para ser lanzado y al mismo tiempo, recopila cantidad mínima de características necesarias para obtener retroalimentación valiosa de los usuarios.
- Características de MVP :
  - Funcionalidad Esencial
  - Lanzamiento Rápido
  - Aprendizaje Iterativo
  - Validación de Hipótesis

Dificultades en el desarrollo de software



- No desarrollar
- Prototipado rápido (MVP)
- Desarrollo evolutivo
- Grandes diseñadores



Roles del proceso de software

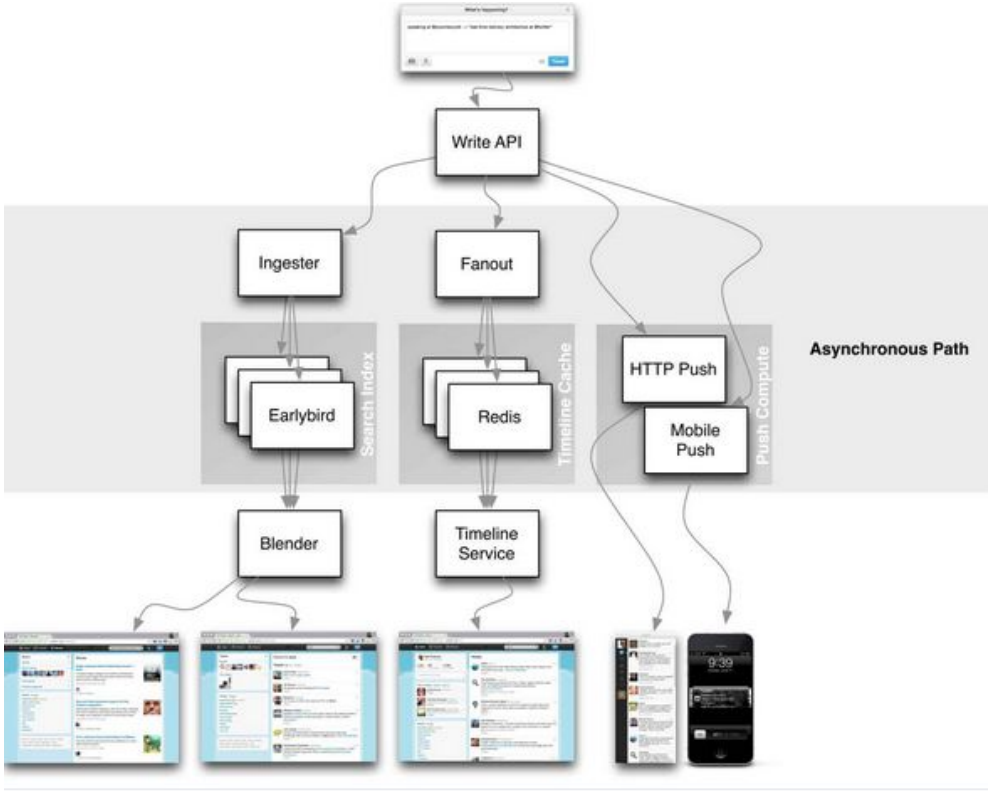
Arquitectura de Software

Arquitectura de software: "La estructura del sistema, compuesta por elementos de software, sus propiedades visibles y sus relaciones" Según: Software Architecture in practice (Bass, Clements & Kazman, 2003)

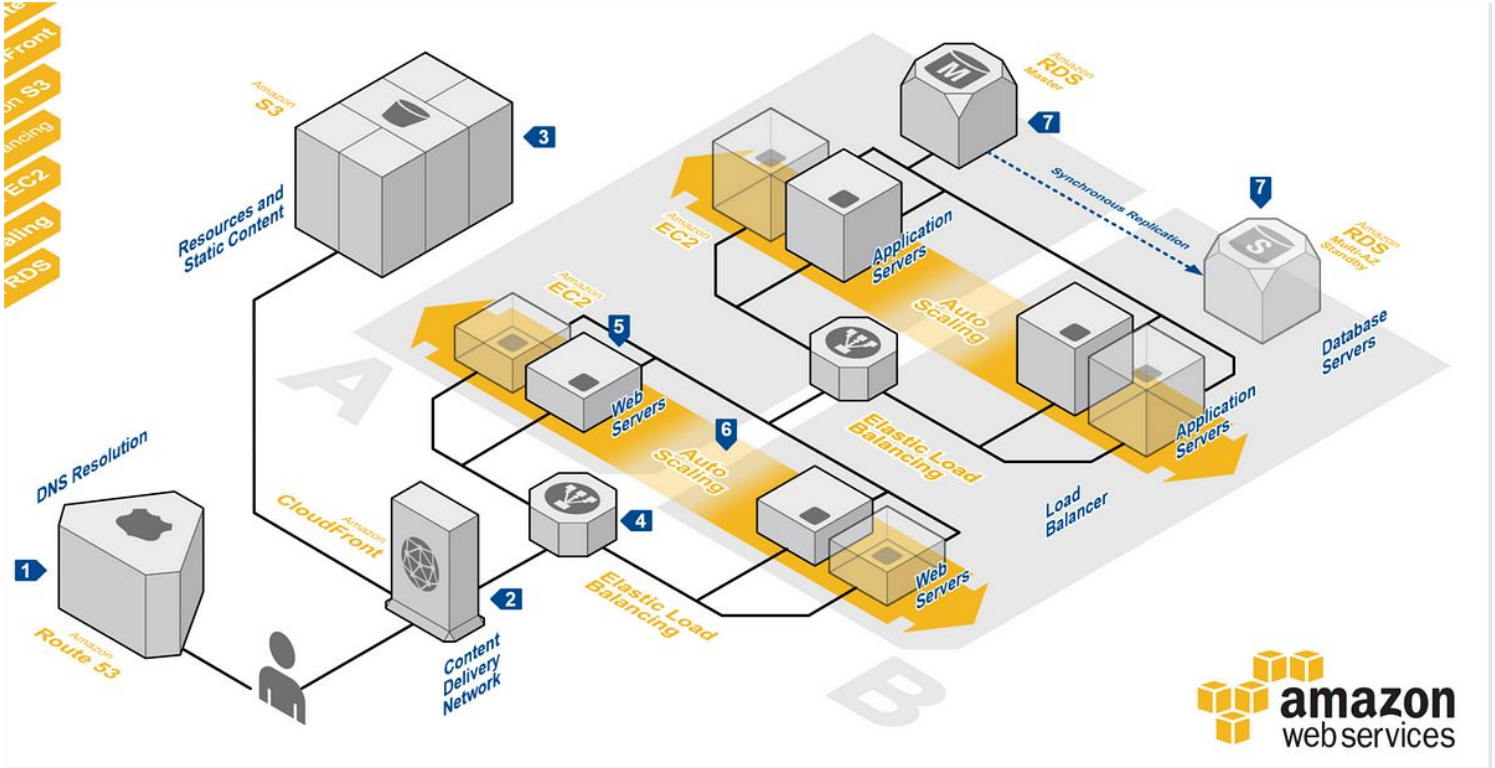
"Conjunto de decisiones principales de diseño tomadas para el sistema" Según: Software Architecture: Foundations, Theory and Practice (Taylor, 2010)

"(...) la arquitectura se reduce a las cosas importantes, cualesquiera que sean" Según: Patterns of Enterprise Application Architecture (Fowler, 2002)

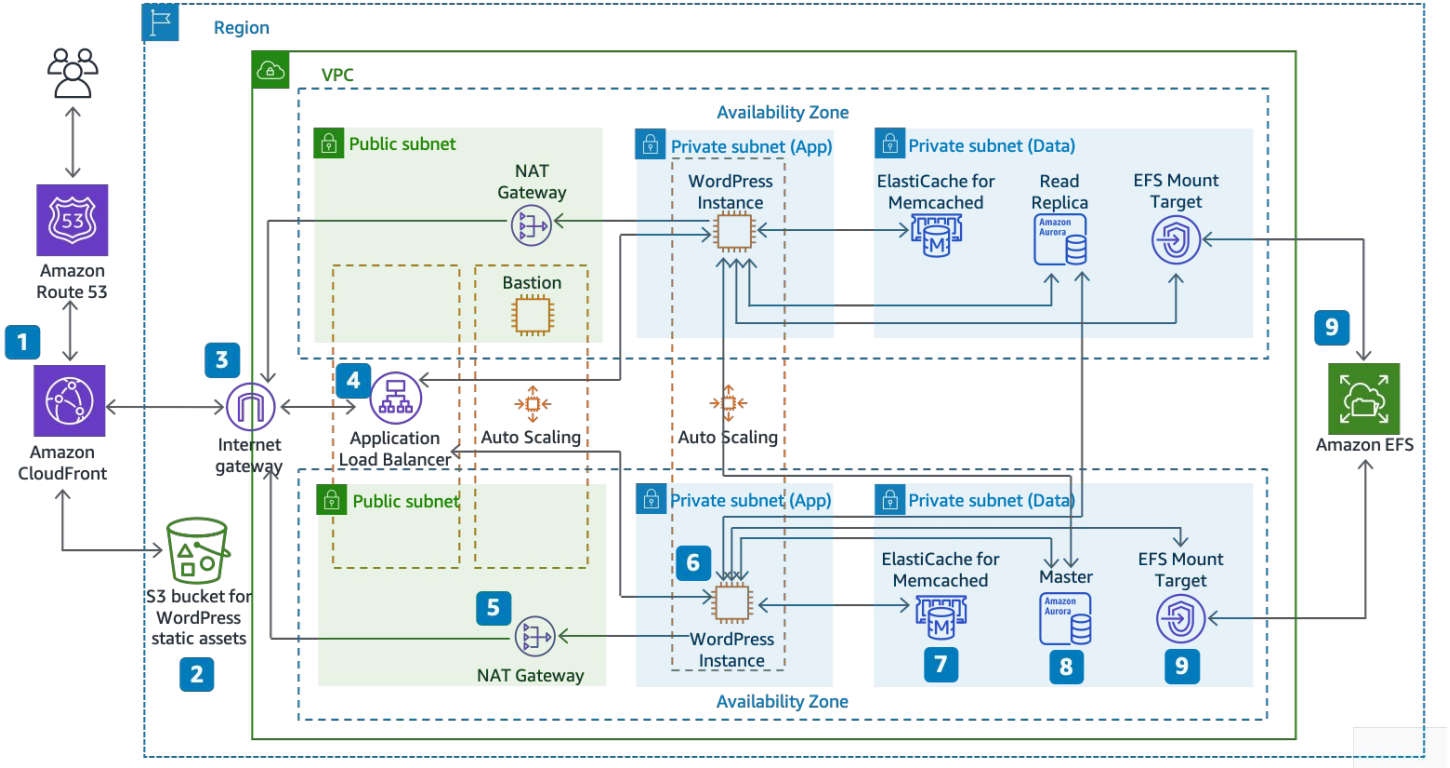
Ejemplos de Arquitectura de Software



Ejemplos de Arquitectura de Software



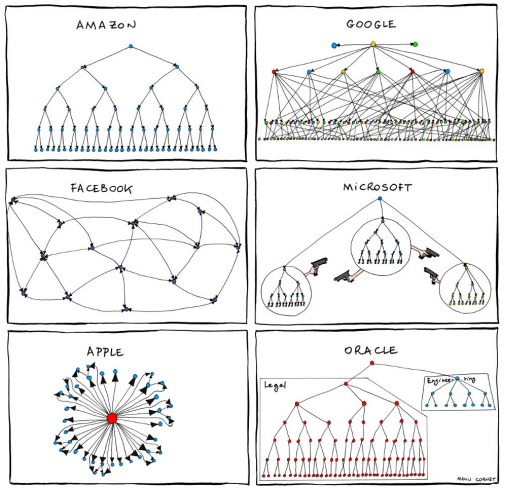
Ejemplos de Arquitectura de Software

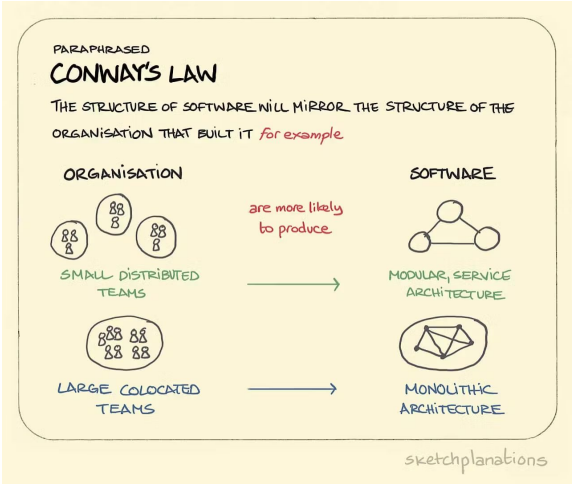


Comunicación y software - Ley de Conway -

La Ley de Conway, formulada por el programador y científico de la computación Melvin Conway en 1967, establece que "las organizaciones que diseñan sistemas están condenadas a producir diseños que son copias de la estructura de comunicación de esas organizaciones".

La ley destaca la influencia de la estructura organizativa en el diseño de sistemas de software.





Objetivos del arquitecto de software

El arquitecto de software es un profesional clave en el desarrollo de software con la responsabilidad de diseñar la estructura general de un sistema, tomando decisiones importantes sobre la organización de componentes, la distribución de responsabilidades y la elección de tecnologías.

Su objetivo es garantizar que el sistema cumpla con los requisitos funcionales, no funcionales, tanto presentes como futuros, de manera eficiente, sostenible y fácil de mantener.

1. Diseño efectivo del sistema
2. Eficiencia y rendimiento
3. Mantenibilidad y adaptabilidad
4. Cumplimiento de requisitos no funcionales
5. Adopción de tecnologías apropiadas
6. Gestión de riesgos
7. Colaboración y comunicación
8. Alineación con objetivos de negocio

Bibliografía

- "Enterprise Architecture as Strategy: Creating a Foundation for Business Execution" (Arquitectura empresarial como estrategia: creando una base para la ejecución empresarial) de Jeanne W. Ross, Peter Weill y David C. Robertson.
- "Enterprise Architecture at Work: Modelling, Communication and Analysis" (Arquitectura empresarial en acción: modelado, comunicación y análisis) de Marc Lankhorst.
- "The Open Group Architecture Framework (TOGAF®) Version 9.2" (El marco de trabajo de arquitectura de The Open Group (TOGAF®) Versión 9.2).
- "Enterprise Architecture Planning: Developing a Blueprint for Data, Applications, and Technology" (Planificación de arquitectura empresarial: desarrollo de un plan para datos, aplicaciones y tecnología) de Steven H. Spewak y Michael J. Tiemann.
- "An Introduction to Enterprise Architecture: Third Edition" (Una introducción a la arquitectura empresarial: tercera edición) de Scott A. Bernard.
- "The Zachman Framework for Enterprise Architecture: A Primer on Enterprise Engineering and Manufacturing" (El marco de trabajo de Zachman para arquitectura empresarial: una introducción a la ingeniería y fabricación empresarial) de John A. Zachman.
- "Building Enterprise Architecture: How to Define, Design, and Deliver a Successful EA" (Construyendo arquitectura empresarial: cómo definir, diseñar y entregar una EA exitosa) de Melissa Cook, S. Jane Fritz y Peter Matthijssen.
- "Enterprise Architecture Body of Knowledge (EABOK)" (Cuerpo de conocimiento de arquitectura empresarial) del Architecture and Governance Magazine.
- "No Silver Bullet: Essence and Accidents of Software Engineering" fue escrito por Frederick P. Brooks Jr. y fue publicado por primera vez en 1986.