UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
CAMPUS 1

ING. EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE

6 “M”

TALLER DE DESARROLLO 4

SUBCOMPETENCIA 1

ACT. 1.3 DEFINICIÓN DE LOS REQUERIMIENTOS DE LA ARQUITECTURA

ALUMNO: MARCO ANTONIO ZÚÑIGA MORALES – A211121

DOCENTE: DR. LUIS GUTIÉRREZ ALFARO

TUXTLA GUTIÉRREZ, CHIAPAS
VIERNES, 25 DE AGOSTO DE 2023

ÍNDICE

Introducción

En el proceso de diseño y desarrollo de software, los requerimientos desempeñan un papel fundamental al definir lo que un sistema debe lograr y cómo debe comportarse. Los requerimientos de arquitectura de software son la base sobre la cual se construirá la estructura y funcionalidad del sistema. Estos requerimientos se desglosan en tres categorías principales: Requerimientos Funcionales, Requerimientos No Funcionales y Casos de Uso. Cada uno de estos conceptos juega un papel crucial en la determinación del éxito y la efectividad de un proyecto de desarrollo de software.

A continuación, en este documento se hablará de los siguientes temas antes mencionados:

Desarrollo

Definición De Los Requerimientos De La Arquitectura

Los requerimientos de la arquitectura de software son especificaciones detalladas que definen cómo debe estar estructurado y organizado un sistema de software. Estos requerimientos determinan las partes principales del sistema, cómo se comunican entre sí, las tecnologías y plataformas a utilizar, y cómo se asegura la calidad y el rendimiento del sistema. Son fundamentales para establecer una base sólida para el diseño y desarrollo del software, ya que guían las decisiones arquitectónicas y garantizan que el sistema cumpla con los objetivos del proyecto y las necesidades de los usuarios.

Los requerimientos de la arquitectura de software pueden incluir los siguientes aspectos:

1. **Componentes y Módulos:**
   Definición de los componentes principales del sistema y cómo se relacionan entre sí. Esto incluye identificar los módulos, servicios y subsistemas que formarán parte de la arquitectura.

2. **Interacciones:**
   Especificación de cómo los diferentes componentes se comunicarán y colaborarán entre sí. Esto incluye definir los protocolos de comunicación, los flujos de datos y las interfaces de programación de aplicaciones (API).

3. **Despliegue:**
   Indicación de cómo se implementarán y desplegarán los componentes en la infraestructura de hardware y software. Esto puede incluir decisiones sobre la arquitectura de servidores, contenedores, nubes y otros entornos de implementación.

4. **Escalabilidad:**
   Requerimientos relacionados con la capacidad del sistema para manejar aumentos en la carga de trabajo y la cantidad de usuarios. Esto puede implicar estrategias de escalado vertical u horizontal.

5. **Seguridad:**
   Especificación de los requisitos de seguridad del sistema, incluida la autenticación, autorización, cifrado y otras medidas de protección de datos y accesos.

6. **Rendimiento:**
   Definición de los estándares de rendimiento que el sistema debe cumplir. Esto puede incluir tiempos de respuesta, latencia, rendimiento de base de datos y otras métricas.

7. **Mantenibilidad:**
   Requerimientos que facilitan el mantenimiento y la evolución del sistema en el tiempo, como la modularidad, la documentación adecuada y las prácticas de desarrollo limpio.

8. **Usabilidad:**
   Especificaciones relacionadas con la experiencia del usuario y la interfaz de usuario. Esto puede incluir pautas para la navegación, el diseño de la interfaz y la accesibilidad.

9. **Integración:**
   Definición de cómo el sistema interactuará con sistemas externos, ya sea a través de API, integraciones de bases de datos u otros mecanismos.

10. **Restricciones Técnicas:**
    Inclusión de limitaciones técnicas y tecnológicas que el sistema debe respetar, como requisitos de hardware, compatibilidad de software y restricciones de plataforma.

Requerimientos Funcionales

Los requerimientos funcionales se refieren a las capacidades y funciones específicas que un sistema de software debe ofrecer. Describen las interacciones del sistema con los usuarios y otros sistemas, detallando las acciones que el software debe realizar en diferentes situaciones. Estos requerimientos responden a preguntas sobre "qué" debe hacer el software y son esenciales para definir el comportamiento del sistema desde una perspectiva funcional.

Algunos ejemplos de requerimientos funcionales del software son los siguientes:

- **Registro de Usuarios:**
  El sistema permitirá a los usuarios registrarse proporcionando un nombre de usuario, dirección de correo electrónico y contraseña. Una vez registrados, los usuarios podrán iniciar sesión en el sistema.

- **Búsqueda de Productos:**
  El sistema proporcionará una función de búsqueda que permita a los usuarios buscar productos por nombre, categoría o precio. Los resultados de la búsqueda se mostrarán en una lista.

- **Añadir al Carrito:**
  Los usuarios podrán agregar productos al carrito de compras desde la lista de resultados de búsqueda. El sistema actualizará automáticamente la cantidad y el costo total del carrito.

- **Realizar Pedido:**
  Los usuarios podrán revisar su carrito de compras y proceder a realizar un pedido. El sistema solicitará la información de envío y presentará un resumen del pedido antes de confirmar.

- **Procesamiento de Pagos:**
  El sistema permitirá a los usuarios seleccionar un método de pago, ingresar los detalles de la tarjeta y procesar el pago. Se enviará una confirmación de pago al usuario y al sistema administrativo.

- **Envío de Notificaciones:**
  El sistema enviará notificaciones por correo electrónico a los usuarios cuando se realice un pedido, se procese un pago o se actualice el estado del pedido.

- **Generación de Reportes:**
  El sistema generará reportes mensuales de ventas que incluirán información sobre los productos más vendidos, ingresos totales y análisis de tendencias.

- **Gestión de Usuarios Administrativos:**
  Los administradores podrán agregar, editar y eliminar usuarios administrativos. También podrán asignar roles y permisos a los usuarios.

- **Control de Inventario:**
  El sistema mantendrá un registro actualizado del inventario de productos, actualizando automáticamente las cantidades disponibles después de cada compra.

- **Seguimiento de Pedidos:**
  Los usuarios podrán rastrear el estado de sus pedidos a través del sistema, viendo detalles como el estado del envío y la fecha de entrega estimada.

Requerimientos No Funcionales

Los requerimientos no funcionales se centran en atributos que afectan la calidad y el rendimiento del sistema más allá de sus funcionalidades básicas. Incluyen aspectos como el rendimiento, la seguridad, la usabilidad, la escalabilidad, la disponibilidad y otros factores que influyen en la experiencia del usuario y en el éxito general del software. Estos requerimientos son críticos para garantizar que el sistema no solo funcione correctamente, sino que también cumpla con los estándares de calidad y satisfaga las expectativas de los usuarios.

- **Rendimiento:**
  El sistema debe cargar la página de inicio en menos de 2 segundos y manejar una carga máxima de 1000 usuarios concurrentes sin degradar la velocidad de respuesta.

- **Usabilidad:**
  La interfaz de usuario debe ser intuitiva y fácil de usar para usuarios sin experiencia previa en el software. Los botones y elementos de navegación deben estar claramente etiquetados.

- **Seguridad:**
  La información del usuario, incluidos los datos personales y la información de pago, debe ser cifrada y protegida de accesos no autorizados. El sistema debe tener medidas de autenticación y autorización robustas.

- **Disponibilidad:**
  El sistema debe estar disponible para su uso las 24 horas del día, los 7 días de la semana, con un tiempo de inactividad planificado mínimo para el mantenimiento.

- **Escalabilidad:**
  El sistema debe ser escalable para manejar un aumento en la cantidad de usuarios y la carga de trabajo. Debe ser capaz de crecer a medida que aumente la demanda.

- **Compatibilidad:**
  El software debe ser compatible con una variedad de navegadores web, sistemas operativos y dispositivos, como computadoras de escritorio, tabletas y teléfonos móviles.

- **Mantenibilidad:**
  El código debe estar bien documentado y seguir las mejores prácticas de programación para facilitar la futura expansión, actualizaciones y correcciones.

- **Tiempo de Respuesta:**
  El sistema debe responder a las solicitudes de los usuarios en menos de 200 milisegundos en promedio, para proporcionar una experiencia fluida.

- **Integridad de Datos:**
  Los datos almacenados en el sistema deben ser precisos y consistentes, sin errores de duplicación ni pérdida.

- **Cumplimiento Normativo:**
  El sistema debe cumplir con las regulaciones y estándares relevantes de la industria, como GDPR para la privacidad de datos.

Casos de uso

Los casos de uso son escenarios detallados que describen cómo los usuarios interactúan con el sistema en situaciones del mundo real. Representan acciones específicas que los usuarios realizan con el software para lograr un objetivo determinado. Los casos de uso son herramientas valiosas en el análisis y diseño de sistemas, ya que ayudan a comprender las necesidades de los usuarios, a definir flujos de trabajo y a identificar los requisitos funcionales y no funcionales.

Los casos de uso son una herramienta crucial en la ingeniería de software para capturar y describir las interacciones entre los usuarios o actores y el sistema. Representan situaciones del mundo real en las que un usuario interactúa con el software para lograr un objetivo específico. Cada caso de uso describe un escenario particular que muestra cómo se utilizará el software en una situación concreta

Elementos de los casos de uso

Algunos de los elementos de los casos de uso son los siguientes:

- **Nombre del Caso de Uso:** El título o nombre del caso de uso refleja la acción específica que se está describiendo, como "Prestar un Libro", "Realizar una Compra", etc.

- **Actores:** Los actores son las entidades que interactúan con el sistema. Pueden ser usuarios finales, sistemas externos o incluso otros componentes del mismo sistema. Los actores desempeñan un papel en el caso de uso y pueden iniciar o recibir acciones del sistema.

- **Descripción:** Proporciona una breve explicación de lo que implica el caso de uso. Resume en qué consiste la interacción entre el actor y el sistema.

- **Flujo Principal:** Este es el flujo de eventos más común y deseado en el caso de uso. Describe los pasos específicos que ocurren cuando el actor interactúa con el sistema para lograr el objetivo del caso de uso.

- **Flujos Alternativos:** Los flujos alternativos representan acciones que difieren del flujo principal, pero aún conducen al mismo objetivo. Pueden surgir debido a decisiones del usuario, condiciones especiales o situaciones inesperadas.

- **Flujo de Excepción:** Estos son los eventos que ocurren cuando algo sale mal o se presenta una situación excepcional. Describen cómo el sistema maneja errores o problemas técnicos y cómo comunica estos problemas al actor.

- **Precondiciones:** Las precondiciones son las condiciones que deben cumplirse antes de que el caso de uso pueda comenzar. Pueden incluir cosas como el usuario debe haber iniciado sesión o el sistema debe estar en un estado específico.

- **Postcondiciones:** Las postcondiciones describen el estado del sistema después de que se ha completado el caso de uso. Pueden reflejar cambios en los datos, estados o cualquier otro efecto visible.

- **Diagramas de Casos de Uso:** A menudo, los casos de uso se visualizan mediante diagramas de casos de uso, que muestran las relaciones entre los actores y los casos de uso. Estos diagramas proporcionan una vista de alto nivel de cómo se conectan diferentes partes del sistema.

Diagrama de casos de uso

El diagrama de casos de uso es una herramienta de modelado en la ingeniería de software que se utiliza para visualizar las interacciones entre los actores (usuarios o sistemas externos) y un sistema de software. Estos diagramas son parte de la notación UML (Lenguaje de Modelado Unificado) y son ampliamente utilizados para capturar y comunicar los requisitos funcionales del sistema desde la perspectiva de los usuarios y sus objetivos.

Un diagrama de casos de uso consta de varios elementos:

- **Actores:** Representan las entidades externas que interactúan con el sistema. Los actores pueden ser usuarios reales, otros sistemas, dispositivos u otros elementos que se comunican con el sistema.

- **Casos de Uso:** Representan las distintas funciones, tareas o interacciones que los actores pueden llevar a cabo en el sistema. Cada caso de uso describe una acción específica que un actor puede realizar con el sistema.

- **Relaciones:** Las líneas que conectan los actores con los casos de uso muestran cómo los actores están involucrados en esos casos de uso. Estas relaciones indican qué funciones están disponibles para cada actor.

- **Inclusión y Extensión:** Estas relaciones detallan cómo un caso de uso puede incluir otro caso de uso en su flujo normal, o cómo un caso de uso puede extender otro caso de uso en situaciones específicas.

- **Sistema:** El sistema generalmente se representa como un rectángulo en el centro del diagrama, rodeado por los actores y los casos de uso que interactúan con él.

Conclusión

En conclusión, los pilares fundamentales en el proceso de diseño y desarrollo de software son los requerimientos de la arquitectura, los requerimientos funcionales, los requerimientos no funcionales y los casos de uso. Estos componentes trabajan en conjunto para dar forma a sistemas de software exitosos y alineados con las necesidades de los usuarios y las metas del proyecto.

Los requerimientos de la arquitectura de software establecen los cimientos sobre los cuales se construirá todo el sistema. Definen la estructura, los componentes y las interacciones clave, proporcionando una guía esencial para los arquitectos y desarrolladores.

Los requerimientos funcionales se centran en lo que el sistema debe hacer. Estas especificaciones detallan las acciones, operaciones y tareas que el software debe ser capaz de llevar a cabo para satisfacer las necesidades de los usuarios.

Los requerimientos no funcionales, por su parte, abarcan aspectos vitales de la calidad del sistema, como su rendimiento, seguridad, usabilidad y escalabilidad. Estos requisitos garantizan que el software no solo funcione correctamente, sino que también brinde una experiencia confiable y satisfactoria.

Los casos de uso complementan este panorama al proporcionar escenarios concretos de interacción entre los usuarios y el sistema. Estos casos capturan cómo se utilizará el software en situaciones del mundo real y ayudan a definir flujos de trabajo eficientes y procesos intuitivos.

En conjunto, estos conceptos forman un marco integral para abordar el proceso de desarrollo de software de manera efectiva. Al combinar una arquitectura bien definida, funcionalidades precisas, atributos de calidad sólidos y una comprensión clara de las interacciones de los usuarios, los equipos de desarrollo están mejor equipados para crear sistemas de software exitosos y satisfactorios.

Referencia Bibliográfica

Casallas, R. (s/f). Casos de Uso · Libro Desarrollo de Software. Gitbooks.io. Recuperado el 23 de agosto de 2023, de https://rcasalla.gitbooks.io/libro-desarrollo-de-software/content/libro/temas/t_requerimientos/req_casosuso.html

de los Angeles Ingrig Sánchez Zarazúa, L. I. M. (s/f). Descripción de funcionalidad: una práctica para especificar requerimientos. Unam.mx. Recuperado el 23 de agosto de 2023, de https://www.red-tic.unam.mx/recursos/2021/funcionalidad-especificar-requerimientos .pdf

Ávila, C. (s/f). Requerimientos NO funcionales. Edu.co. Recuperado el 18 de agosto de 2023, de https://repositorio.konradlorenz.edu.co/micrositios/001-1527/requerimientos_no_func ionales.html

Northware. (2022). Requerimientos en el desarrollo de software y aplicaciones. Northware. https://www.northware.mx/blog/requerimientos-en-el-desarrollo-de-software-y-aplicaciones/

Fernández, L. F. (2006). Arquitectura de software. Software Guru, 2(3), 40-45.

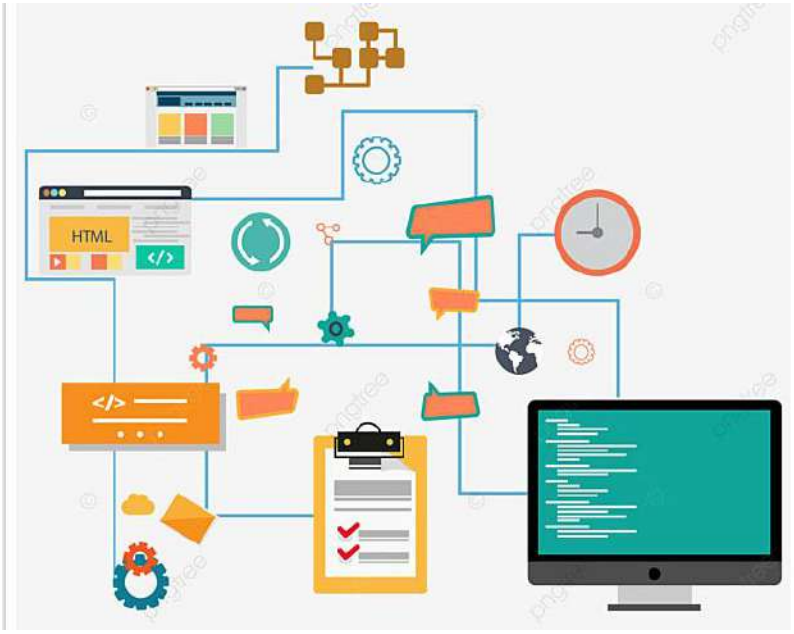
Cristiá, M. (2008). Introducción a la Arquitectura de Software. Research-Gate.[Online]. Recuperado de: https://www. researchgate. net/publication/251932352 Introduccion a la Arquitectura de Software.

Sommerville, I. (2005). Requerimientos del software. Ingeniería del software, 7a ed., PEARSON EDUCACIÓN, Madrid, SPA, 109-110.

Chaves, M. A. (2005). La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. InterSedes: Revista de las Sedes Regionales, 6(10), 1-13.

Soto, S. D., Rivero, L. C., & Olguín, E. L. (2019). El uso de software libre en el control de inventarios: caso de estudio. Ciencia Administrativa, 12(1), 2-7.

Anexos



¿Qué es un Arquitecto de software?

El **Arquitecto de software** es el encargado de desarrollar la propuesta técnica para **crear plataformas digitales**. Desde la definición de la estructura hasta los estándares de código que se deben implementar.

Hireline