UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
CAMPUS 1

Licenciatura en Ingeniería en Desarrollo y Tecnologías de Software

Act.1.3 Definición de los Requerimientos de la Arquitectura

6 “M”

Materia: Taller de Desarrollo 4

Docente: Luis Gutiérrez Alfaro

ALUMNO:
A 211387
Steven de Dios Montoya Hernández

TUXTLA GUTIÉRREZ, CHIAPAS
Viernes, 25 de Agosto de 2023, 23:59

Índice

Introducción:

En el apasionante mundo del desarrollo de software, existe un conjunto fundamental de conceptos que son los pilares sobre los que se construyen soluciones tecnológicas innovadoras y eficaces. Estos pilares son los requisitos arquitectónicos, los requisitos funcionales, los requisitos no funcionales y los casos de uso. Estos elementos esenciales no solo son las piedras angulares del diseño y desarrollo de software, sino que también definen la naturaleza de un sistema exitoso. En este informe, exploramos cómo estos elementos interconectados forman la base de soluciones técnicas que satisfacen las necesidades y expectativas de los usuarios.

Definición de los Requerimientos de la Arquitectura

Los requerimientos son una especificación de lo que debe ser implementado. Son descripciones acerca de cómo debe comportarse el sistema o bien acerca de una propiedad o atributo del sistema. Pueden ser una restricción en el proceso de desarrollo del sistema

Estos requerimientos son esenciales para garantizar que la arquitectura cumpla con los objetivos, necesidades y expectativas del proyecto. La definición de los Requerimientos de la Arquitectura puede involucrar varios aspectos clave:

Objetivos del sistema: Establecer los objetivos generales del sistema, como funcionalidades clave, rendimiento deseado, niveles de seguridad y calidad del servicio.

Necesidades de los usuarios: Identificar las necesidades y expectativas de los usuarios finales y otros stakeholders involucrados en el proyecto. Estos requisitos pueden incluir funcionalidades específicas, usabilidad, escalabilidad y otros aspectos importantes.

Restricciones técnicas: Definir las limitaciones y restricciones técnicas que deben tenerse en cuenta al diseñar la arquitectura. Esto puede incluir elecciones de tecnología, compatibilidad con plataformas existentes y requisitos de integración.

Requerimientos de rendimiento: Establecer los criterios de rendimiento, como tiempos de respuesta, capacidad de carga, eficiencia en el uso de recursos y escalabilidad.

Requerimientos de seguridad: Identificar los niveles de seguridad requeridos, incluyendo autenticación, autorización, cifrado y protección contra posibles amenazas y ataques.

Requerimientos de escalabilidad: Definir cómo el sistema debe crecer y adaptarse a medida que aumenta la demanda de usuarios y datos.

Requerimientos de mantenibilidad: Especificar cómo se debe estructurar la arquitectura para facilitar futuras actualizaciones, modificaciones y mejoras.

Requerimientos de integración: Definir cómo el sistema se conectará con otros sistemas, bases de datos o servicios externos.

Requerimientos de disponibilidad: Establecer los criterios de tiempo de actividad y disponibilidad del sistema, así como cómo se manejarán las interrupciones y fallas.

Requerimientos de usabilidad: Identificar los criterios de facilidad de uso, accesibilidad y experiencia del usuario.

Requerimientos de cumplimiento: Definir cómo el sistema debe cumplir con regulaciones y estándares específicos de la industria.

Requerimientos de rendimiento: Establecer los criterios de rendimiento, como tiempos de respuesta, capacidad de carga, eficiencia en el uso de recursos y escalabilidad.

Requerimientos Funcionales

¿Qué son los requisitos funcionales?

Un requisito funcional es una declaración de cómo debe comportarse un sistema. Define lo que el sistema debe hacer para satisfacer las necesidades o expectativas del usuario. Los requisitos funcionales se pueden considerar como características que el usuario detecta. Son diferentes de los requisitos no funcionales, que definen cómo debe funcionar internamente el sistema (p. ej., rendimiento, seguridad, etc.).

Los requisitos funcionales se componen de dos partes: función y comportamiento. La función es lo que hace el sistema (por ejemplo, "calcular el impuesto sobre las ventas"). El comportamiento es cómo lo hace el sistema (p. ej., "El sistema calculará el impuesto sobre las ventas multiplicando el precio de compra por la tasa impositiva").

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir. Los requisitos de comportamiento para cada requisito funcional se muestran en los casos de uso. Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación.

Como se define en la ingeniería de requisitos, los requisitos funcionales establecen los comportamientos del software.

Típicamente, un analista de requisitos genera requisitos funcionales después de realizar los casos de uso. Sin embargo, esto puede tener excepciones, ya que el desarrollo de software es un proceso iterativo y algunos requisitos son previos al diseño de los casos de uso. Ambos elementos (casos de uso y requisitos) se complementan en un proceso bidireccional.

Un requisito funcional típico contiene un nombre, un número de serie único y un resumen. Esta información se utiliza para ayudar al lector a entender por qué el requisito es necesario, y para seguir al mismo durante el desarrollo del producto.

El núcleo de los requisitos yace en la descripción del comportamiento requerido, que debe ser clara y concisa. Este comportamiento puede provenir de reglas organizacionales o del negocio, o ser descubiertas por interacción con usuarios, inversores y otros expertos en la organización.

Los Requerimientos Funcionales son especificaciones detalladas de las funciones y características que un sistema o software debe cumplir para satisfacer las necesidades de los usuarios y alcanzar sus objetivos. Estos requisitos describen cómo el sistema debe interactuar con los usuarios, procesar datos y llevar a cabo diversas tareas. Algunos ejemplos de Requerimientos Funcionales en una arquitectura de software podrían incluir:

Interfaz de Usuario: Descripción de la apariencia, diseño y funcionalidades de la interfaz gráfica del usuario, incluyendo botones, menús, formularios, etc.

Procesamiento de Datos: Definición de cómo se capturan, almacenan, procesan y presentan los datos en el sistema. Esto podría incluir la gestión de bases de datos, cálculos, generación de informes, etc.

Flujo de Trabajo: Especificación de los pasos que los usuarios deben seguir para realizar tareas específicas en el software. Esto podría incluir secuencias de acciones, decisiones y validaciones.

Gestión de Usuarios: Detalles sobre la autenticación, autorización y roles de usuarios en el sistema, así como la gestión de perfiles y contraseñas.

Comunicación y Notificaciones: Requisitos relacionados con la forma en que el sistema se comunica con los usuarios, ya sea a través de notificaciones por correo electrónico, mensajes en pantalla u otras formas de comunicación.

Integración con Otros Sistemas: Especificaciones sobre cómo el software se integrará con otros sistemas o servicios externos, como APIs de terceros o sistemas heredados.

Seguridad: Detalles sobre los controles de seguridad que se implementarán en el software para proteger los datos y garantizar la privacidad de los usuarios.

Rendimiento y Escalabilidad: Requerimientos relacionados con el rendimiento del sistema, incluyendo tiempos de respuesta, capacidad de carga y escalabilidad para manejar un aumento en la demanda de usuarios.

Soporte y Mantenimiento: Especificaciones sobre cómo se proporcionará soporte técnico y mantenimiento del software, incluyendo actualizaciones, correcciones de errores y mejoras.
Casos de Uso: Descripciones detalladas de situaciones específicas en las que el sistema debe comportarse de manera particular para satisfacer las necesidades del usuario.

Requerimientos No Funcionales

¿Qué son los requisitos no funcionales?

Los requisitos no funcionales, también conocidos como "requisitos de calidad" o "requisitos de atributos del sistema", son características y criterios que describen cómo debe ser el rendimiento, la seguridad, la usabilidad y otros aspectos de un sistema o software más allá de su funcionalidad básica.
Utilizar requisitos no funcionales es una parte importante al momento de desarrollar un sistema, hará que tu software o aplicación sea eficaz, eficiente y cumpla con las expectativas y necesidades del usuario.

No obstante, debes entender que estos trabajan de la mano con los requisitos funcionales. ¿A qué nos referimos con esto? Verás, los requisitos funcionales se enfocan en el qué, mientras que los requisitos no funcionales se enfocan en el cómo y en qué medida cumplen con ciertas cualidades. A continuación te lo explicamos detalladamente.

Los Requerimientos No Funcionales son criterios y restricciones que se aplican a un sistema de software para definir su calidad, rendimiento y comportamiento en áreas que van más allá de las funciones específicas que realiza. Estos requisitos no se centran en qué hace el sistema, sino en cómo lo hace y cómo satisface las necesidades y expectativas de los usuarios. Algunos ejemplos de Requerimientos No Funcionales en una arquitectura de software incluyen:

Rendimiento: Establecer criterios para la velocidad, eficiencia y capacidad de respuesta del sistema. Esto podría incluir tiempos de carga, velocidad de procesamiento, latencia y rendimiento bajo carga.

Escalabilidad: Definir cómo el sistema puede crecer y adaptarse a un aumento en la cantidad de usuarios o la carga de trabajo sin degradar su rendimiento.

Disponibilidad: Especificar el tiempo de funcionamiento esperado del sistema y cómo se gestionarán las interrupciones planificadas o no planificadas.

Seguridad: Definir los mecanismos de seguridad necesarios para proteger los datos y la integridad del sistema. Esto podría incluir cifrado, autenticación, autorización y medidas para prevenir ataques.

Fiabilidad: Establecer cómo el sistema debe comportarse en situaciones de fallo, y cómo debe recuperarse de ellos sin pérdida de datos o funcionalidad.

Mantenibilidad: Especificar cómo se debe estructurar el código y la arquitectura para facilitar la detección y corrección de errores, así como la incorporación de futuras mejoras.

Usabilidad: Definir la facilidad de uso y la experiencia del usuario, incluyendo aspectos como la intuitividad de la interfaz, la accesibilidad y la capacidad de aprendizaje.

Portabilidad: Establecer la capacidad del sistema para ejecutarse en diferentes plataformas, sistemas operativos o entornos sin necesidad de modificaciones significativas.

Interoperabilidad: Especificar cómo el sistema debe interactuar y compartir datos con otros sistemas o servicios externos.

Cumplimiento normativo: Definir cómo el sistema debe cumplir con regulaciones y estándares específicos de la industria, como requisitos de privacidad, seguridad o accesibilidad.

Tolerancia a fallos: Indicar cómo el sistema debe manejar y recuperarse de errores y fallos, evitando la pérdida de datos o la degradación del servicio.

Casos de Uso

Los casos de uso en una arquitectura de software son descripciones detalladas de cómo los usuarios interactúan con el sistema para lograr objetivos específicos. Estos casos de uso son escenarios que representan situaciones del mundo real en las que los usuarios utilizan el software para llevar a cabo ciertas tareas o funciones. Los casos de uso son una herramienta importante en el proceso de diseño y desarrollo de software, ya que ayudan a comprender las necesidades de los usuarios y a definir cómo debe comportarse el sistema en diversas situaciones. Cada caso de uso generalmente consta de los siguientes elementos:

Nombre del caso de uso: Un nombre descriptivo que identifica la tarea o función específica que se está describiendo.

Actores: Las entidades que interactúan con el sistema en el caso de uso. Pueden ser usuarios, sistemas externos u otros componentes del sistema.

Descripción: Una descripción detallada de la secuencia de pasos que el actor y el sistema siguen para lograr el objetivo del caso de uso.

Precondiciones: Las condiciones que deben cumplirse antes de que el caso de uso pueda comenzar.

Flujo principal: Los pasos en la secuencia de acciones que representan el escenario típico del caso de uso.

Flujos alternativos: Las variantes o situaciones excepcionales que podrían ocurrir durante la ejecución del caso de uso.

Postcondiciones: El estado en el que se encuentra el sistema y los actores después de que se completa el caso de uso.

Extensiones: Acciones adicionales o flujos alternativos que se pueden activar en respuesta a ciertas condiciones.

Los casos de uso ayudan a definir los requisitos del sistema desde la perspectiva del usuario y proporcionan un marco para diseñar y probar las funcionalidades del software. También son útiles para la comunicación entre los miembros del equipo de desarrollo, los usuarios y otros stakeholders involucrados en el proyecto.

Algunos ejemplos de casos de uso en una arquitectura de software podrían ser:

Registrarse como nuevo usuario: Un usuario se registra en el sistema proporcionando información personal y creando una cuenta.

Realizar una compra: Un usuario selecciona productos, agregar artículos al carrito y completa el proceso de compra.

Gestionar perfiles de usuario: Un administrador del sistema modifica los roles y permisos de los usuarios registrados.

Generar un informe: Un usuario solicita la generación de un informe específico y selecciona los parámetros de filtro.

Recuperar contraseña olvidada: Un usuario que ha olvidado su contraseña utiliza un proceso de recuperación para restablecerla.

DIAGRAMAS DE CASOS DE USO

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).
Un diagrama de casos de uso consta de los siguientes elementos:

- Actor.
- Casos de Uso.
- Relaciones de Uso, Herencia y Comunicación.

Elementos

- **Actor:**
  Una definición previa, es que un Actor es un rol que un usuario juega con respecto al sistema.

- **Caso de Uso:**
  Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación de otro caso de uso.

- **Relaciones:**
  - Asociación
    Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso).
  - Dependencia o Instanciación
    Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea).

Conclusión

En última instancia, los requisitos arquitectónicos, los requisitos funcionales, los requisitos no funcionales y los casos de uso funcionan en sinergia para ofrecer un software robusto, eficiente y orientado al usuario. Desde la base creada por la arquitectura back-end hasta la implementación precisa de los requisitos funcionales y no funcionales, cada paso del proceso de desarrollo es fundamental para el éxito del sistema. Los casos de uso agregan un toque humano al representar escenarios de la vida real donde el software se convierte en una herramienta valiosa y funcional. Al comprender y aplicar estos conceptos, los profesionales del desarrollo de software pueden crear soluciones tecnológicas que no solo cumplen los objetivos, sino que también superan las expectativas y brindan un valor duradero en un mundo cada vez más impulsado por la tecnología.

Bibliografía:

Bass, L., Clements, P., & Kazman, R. (2012). "Software Architecture in Practice." Addison-Wesley Professional.

Wiegers, K. E., & Beatty, J. (2013). "Software Requirements." Microsoft Press.

Cockburn, A. (2000). "Writing Effective Use Cases." Addison-Wesley Professional.

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_funcionales.html

Casallas, R. (s/f). Casos de Uso · Libro Desarrollo de Software. Gitbooks.io. Recuperado el 18 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 18 de agosto de 2023, de https://www.red-tic.unam.mx/recursos/2021/funcionalidad-especificar-requerimientos.pdf