Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Crear tablas

Tarea 3

Jorge Daray Padilla Perez 216584703

MARIA MAGDALENA MURILLO LEANO

Guadalajara 15-2-2022

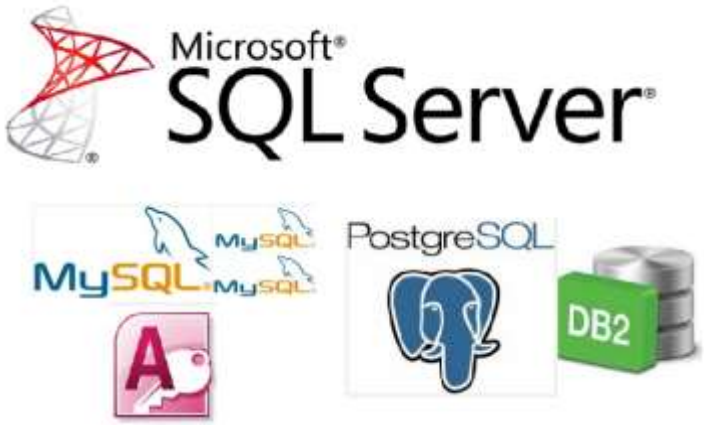
UNIVERSIDAD DE GUADALAJARA
Centro Universitario de Ciencias Exactas e Ingenierías
Padilla Perez Jorge Daray
216584703
ingeniería en computación

Actividad 3.- requerimientos para diseñar base de datos

1. Paso

PARA LA CONSTRUCION DE UNA BASE DE DATOS ES ESCENCIAL CONOCER CUALES SON ALGUNOS DE LOS SISTEMAS GESTORES DE BASES DE DATOS (SGBD)

ALGUNOS DE LOS SGBD UTILIZADOS CON FRECUENCIA SON:



2. Paso diseño e implementación

El diseño e Implementación son mucho más que el análisis (refinamiento y estructuración de los requisitos) por lo que se requiere una separación de intereses.

– El Análisis: prepara y simplifica la siguiente actividad de diseño e implementación, delimitando los temas que deben resolverse y las decisiones que deben tomarse en esas actividades.

– En el Diseño: debemos modelar el sistema y encontrar su forma incluyendo su arquitectura: una forma que de vida a todos los requisitos incorporados en el sistema.

3. Paso Elaboración

Es necesario recabar toda la información posible sobre la realidad, para luego analizarla con detenimiento, desde distintos puntos de vista con el fin de lograr diseñar un modelo que la represente de manera abstracta lo más fielmente posible.

Debemos efectuarnos algunas preguntas con el fin de analizar nuestro sistema:

¿Qué? ¿Quién? ¿Cuándo? ¿Cómo? ¿Dónde?

4. Paso Características

Entidades

- Abstracciones de un objeto del mundo real.
- Representación una colección de objetos que tienen propiedades comunes.
- Ejemplo: CLIENTE

Atributos

- Propiedades de una entidad
- Ejemplo: Nombre y apellido, edad, dirección, etc.

Relaciones o Flujo de datos

- Intercambio de información entre entidades
- Representan datos en movimiento lógicamente relacionados.
- Describen el movimiento de paquetes de datos de una parte del sistema a otra.

Procesos

- Una actividad, tarea, proceso, función, etc.
- Transforma entradas en salidas

Almacenes

- Colección de datos en reposo.
- archivo en disco
- datos en un fichero de papel
- Ejemplo: una FACTURA

Terminadores o Entidades Externas.

- Representan objetos con los cuales el sistema se comunica.
- Personas, agrupamientos, organizaciones
- otros sistemas de software o hardware
- Se encuentran por fuera del sistema.

5. Paso Arquitectura de tres niveles

Una base de datos perfectamente ordenada permite acceder a información exacta y actualizada.

Access organiza su información en tablas.

- Nivel interno: modelo físico y describe todos los detalles para el almacenamiento de la base de datos, así como los métodos de acceso.

- ❖ Nivel conceptual: oculta los detalles de las estructuras de almacenamiento y se concentra en describir entidades, atributos, relaciones, operaciones de los usuarios y restricciones.
- ❖ Nivel externo: describe la parte de la base de datos que interesa a un grupo de usuarios determinado

6. Paso Seguridad

Los SGBD deben garantizar esta información se encuentre asegurada frente a personas que puedan dañar el sistema

7. Paso respaldo y recuperación

8. Tener un buen diseño de BD

- ✓ Se debe evitar la información duplicada.
- ✓ Se divide la información en tablas para reducir datos redundantes.
- ✓ Definir la relación entre las tablas.
- ✓ Establecer la finalidad de la base de datos.

Requerimientos Funcionales

Los requisitos funcionales son declaraciones de los servicios que prestará el sistema, en la forma en que reaccionará a determinados insumos. Cuando hablamos de las entradas, no necesariamente hablamos sólo de las entradas de los usuarios. Pueden ser interacciones con otros sistemas, respuestas automáticas, procesos predefinidos. En algunos casos, los requisitos funcionales de los sistemas también establecen explícitamente lo que el sistema no debe hacer. Es importante recordar esto: un RF puede ser también una declaración negativa. Siempre y cuando el resultado de su comportamiento sea una respuesta funcional al usuario o a otro sistema, es correcto. Y más aún, no sólo es correcto, sino que es necesario definirlo. Y eso nos lleva al siguiente punto.

Muchos de los problemas en la ingeniería de software (hablando sobre el proceso de desarrollo en sí mismo) comienzan con especificaciones de requisitos inexactas. Es natural que un Analista de Negocio (o quien sea que esté definiendo y documentando los requerimientos del sistema) tome algunas suposiciones como conocimiento universal, o dé por sentado algún comportamiento. Pero recuerde, también es natural que un desarrollador de sistemas interprete un requisito ambiguo de la manera más simple posible, para simplificar su implementación. Un claro ejemplo de estos problemas se puede encontrar en las funciones comunes relacionadas con la Experiencia de usuario:

- Historia de Usuario: Como usuario, quiero que la aplicación sea fácil de usar, de modo que no tenga que pasar mucho tiempo aprendiendo a usarla.
- ¿Qué significa ser “fácil de usar”?
- ¿Fácil para quién?
- ¿Cómo se mide?
- ¿Cómo lo rastreas?
- ¿Cómo se prueba? ¿Contra qué criterios?

Esto no es algo contra los desarrolladores, sino más bien contra el comportamiento humano. Si usted asume algo, otros pueden asumir algo diferente, y eso está bien. Pero el analista es el responsable de la documentación, por lo que debe ser el mismo analista el que trate de asegurarse de que no hay lagunas de comprensión o puntos grises. Esta es también la razón por la que las sesiones de Pre-Planificación y Presentación de Historias son muy importantes para asegurarse de que todo el equipo está en la misma página con respecto a los requisitos, su objetivo de negocio y cómo implementarlos. Volviendo al ejemplo anterior, podríamos dividir el requisito en varios nuevos, más fáciles de entender, desarrollar, probar, rastrear y entregar. Uno de ellos podría serlo:

- Historia de usuario: Como usuario, quiero que se muestre un tutorial al iniciar sesión por primera vez, para que pueda ver para qué se utiliza cada opción de la pantalla de inicio.
- Ya sabes qué mostrar, un tutorial.

- Ya sabes lo que hay que comprobar, que explica cada opción en la pantalla de inicio.
- Usted sabe cuándo necesita ser mostrado, en el primer inicio de sesión del usuario (puede explicar más adelante si el tutorial puede ser omitido, mostrado en otros momentos, accedido desde alguna sección dentro del menú, etc.)

Volviendo a FR, en principio, la especificación de los requisitos funcionales de un sistema debe ser completa y coherente. Completar significa que todos los servicios solicitados por el usuario y/u otro sistema están definidos. La coherencia significa que los requisitos no tienen una definición contradictoria.

Requisitos no funcionales

Se trata de requisitos que no se refieren directamente a las funciones específicas suministradas por el sistema (características de usuario), sino a las propiedades del sistema: rendimiento, seguridad, disponibilidad. En palabras más sencillas, no hablan de “lo que” hace el sistema, sino de “cómo” lo hace. Alternativamente, definen restricciones del sistema tales como la capacidad de los dispositivos de entrada/salida y la representación de los datos utilizados en la interfaz del sistema.

Los requisitos no funcionales se originan en la necesidad del usuario, debido a restricciones presupuestarias, políticas organizacionales, la necesidad de interoperabilidad con otros sistemas de software o hardware, o factores externos tales como regulaciones de seguridad, políticas de privacidad, entre otros.

Existen diferentes tipos de requisitos y se clasifican según sus implicaciones.

- Requisitos del producto. Especifican el comportamiento del producto, como los requisitos de rendimiento sobre la velocidad de ejecución del sistema y la cantidad de memoria necesaria, los requisitos de fiabilidad que establecen la tasa de fallos para que el sistema sea aceptable, los requisitos de portabilidad y los requisitos de usabilidad.
- Requisitos organizativos. Se derivan de las políticas y procedimientos existentes en la organización cliente y en la organización del desarrollador: estándares en los procesos a utilizar; requisitos de implementación tales como lenguajes de programación o el método de diseño a utilizar; y requisitos de entrega que especifican cuándo se entregará el producto y su documentación.
- Necesidades externas. Se derivan de factores externos al sistema y a su proceso de desarrollo. Incluyen los requisitos de interoperabilidad que definen la forma en que el sistema interactúa con los demás sistemas de la organización; los requisitos legales que deben seguirse para garantizar que el sistema funciona dentro de la ley; y los requisitos éticos. Estos últimos se imponen al sistema para asegurar que será aceptado por el usuario.

A veces, en la práctica, la especificación cuantitativa de este tipo de requisitos es difícil. No siempre es posible para los clientes traducir sus objetivos en requisitos cuantitativos. Para algunos de ellos, como el mantenimiento, puede que no se pueda utilizar ninguna métrica; el coste de la verificación objetiva de

los requisitos cuantitativos no funcionales puede ser muy elevado. Y es por eso que también es muy importante ser muy cuidadoso al documentarlos. Un analista puede utilizar el apoyo del equipo de desarrollo y del equipo de pruebas para definir criterios mensurables con el fin de saber cuándo un requisito puede ser marcado con éxito como “Hecho”.

Al igual que hablamos de requisitos funcionales, la generalización nunca es buena, pero en este caso es aún más importante. ¿Por qué? Porque el resultado de un desarrollo de requisitos no funcionales puede no ser explícito para el usuario final.

- Mal ejemplo de RNF: El sistema debe ser seguro.
- ¿Qué tan seguro es “seguro”?
- ¿En qué situaciones?
- ¿Existe una norma a cumplir?
- ¿En qué secciones?
- ¿Qué debe ocurrir si el sistema no puede funcionar tan rápido como se requiere?

En estos casos, la aplicación de un requisito no funcional mal definido puede resultar problemática, costosa y lenta. También porque la mayoría de las veces, una RNF no es algo que el equipo trabaja aislado en un período fijo de tiempo. El RNF puede ser abordado durante todo el proyecto (e incluso antes de comenzar o después de la entrega, en la fase de mantenimiento). Una vez más, el ejemplo anterior podría dividirse en múltiples requisitos que son más fáciles de rastrear e implementar.

- Buen ejemplo de RNF: Todas las comunicaciones externas entre los servidores de datos, la aplicación y el cliente del sistema deben estar cifradas utilizando el algoritmo RSA.
- Sabes qué tipo de comunicaciones necesitan ser encriptadas.
- Usted sabe qué algoritmo usar y validar.

Los requisitos funcionales y no funcionales deben diferenciarse en el documento de requisitos, ya sea un SRS, una cartera de productos o cualquier artefacto que utilice. En la práctica, esto puede resultar difícil. Si se declara un requisito no funcional por separado de los requisitos funcionales, a veces es difícil ver la relación entre ellos. Si los RNF se declaran con requisitos funcionales, es difícil separar las condiciones funcionales de las no funcionales e identificar los requisitos que se refieren al sistema en su conjunto. Se debe encontrar un equilibrio adecuado que dependa del tipo de sistema o aplicación que se especifique. Por ejemplo, si está trabajando con un Product Backlog, podría tener Historias de Usuario separadas para RNFs, pero añadir un enlace a ellas en las RFs que puedan ser impactadas por ellas. Esta es una opción común en la mayoría de los sistemas de seguimiento de billetes utilizados en la actualidad.

En cualquier caso, tanto los RFs como los RNFs deben estar siempre documentados, incluso si es difícil establecer la relación entre ellos en los artefactos. Esto ayudará al equipo a reducir las discusiones de ida y vuelta, ahorrar tiempo y sobre todo, problemas innecesarios con el cliente.