Proyecto práctico – Desarrollo de un sistema.

Material Teórico

1. Metodologías para Análisis de sistemas
2. Ejemplo de un Análisis de un sistema
3. Normalización de Base de Datos

Material para prácticas de clase

4. Video Presentación
5. Index con encabezado fijo
6. CSS de un menú de opciones
7. FrontEnd del desarrollo

Proyecto de desarrollo

8. Video Presentación
9. Detalle de Secciones y Herramientas.
10. Detalle Sección Blog.
11. Detalle Sección Index.
12. Estructura de Index.html
13. Logo y Menú Navbar
14. Código Fuente Archivo Index
15. Detalle proyecto Home Web

GUÍA METODOLÓGICA DESARROLLO DE SISTEMA DE INFORMACIÓN

1. Introducción

Una metodología para el desarrollo de sistemas de información debe ser global, ofrecer las herramientas y técnicas suficientes para cubrir todos esos aspectos que se pueden encontrar en un sistema de este tipo.
Muchas definiciones de sistemas se están dando dentro del mundo de la ingeniería del software: sistemas multimedia, aplicaciones web, sistemas de información global, etc.

En la actualidad, las aplicaciones se desarrollan normalmente en entornos distribuidos, es muy común el que se distribuyan por internet y normalmente tienen asociados elementos multimedia e hipermedia en grandes bases de datos. Se caracterizan por tener grandes requisitos funcionales y de seguridad, múltiples usuarios y en muchos casos indefinidos y con diferentes grados de conocimiento.

Estas aplicaciones se conocen como sistemas de información global, y son, un concepto mucho más genérico que engloba a las aplicaciones que se encuentran en los otros grupos.

El sistema de información global puede verse como si fuera una aplicación multimedia, puesto que normalmente maneja información almacenado en múltiples medios. Pero cuando se distribuye a través de internet, se podría ver como un sistema de información web. Sin embargo, ninguna de las metodologías de estos ámbitos sería adecuada, puesto que no tratan los aspectos de almacenamiento y funcionalidad de manera adecuada, que en los sistemas de información web suelen ser bastante críticos.

Los sistemas de información global almacenan grandes cantidades de información y requieren sistemas de seguridad muy potentes, así como una funcionalidad muy elaborada que asegure que los usuarios van a poder trabajar con esta información de manera adecuada.

Aspectos de los Sistemas de Información Global

La guía metodológica cubre las siguientes características

- Cubre todo el ciclo de vida del proyecto.
- Está orientada al procesos, en donde, debe indicarse qué hacer en cada momento del ciclo de vida.
- Está orientada al producto, es decir, en cada fase se indicará qué hay que obtener y el formato que deberá tener para ello.
- Es sencilla, en sus primeras fases, para facilitar la participación del cliente y del usuario.

• Es completa, para cubrir todas las necesidades del desarrollador y ofrecer una semántica suficiente como para trabajar de forma adecuada todos los aspectos críticos que se han venido destacando de los sistemas.

• Orientada al área asesora interna que direccione los requerimientos enmarcados en unos límites claros respecto a la organización.

2. Metodología para el desarrollo de sistemas de información

2.1. Ciclo de Vida

El método de ciclo de vida de para el desarrollo de sistemas es el conjunto de actividades que se realizan para desarrollar e implantar un Sistema de Información. Se parte por dividir la vida del proyecto en flujos de trabajo. El ciclo de vida debe comprender un total de seis flujos de trabajo: Planificación, Especificación, Análisis, Diseño, Desarrollo, Pruebas e implementación.

En principio estos flujos de trabajo se deben realizar de forma consecutiva, pero si nos basamos en la realidad de los proyectos software, desde un determinado flujo es necesario volver a flujos anteriores para redefinir nuevos aspectos. Por esto, la metodología debe ser secuencial pero permitirá iterar y volver a flujos anteriores.

2.2. Primera fase Planificación

Las tareas iniciales que se deben realizar en esta fase inicial del proyecto deben incluir actividades tales como la delimitación del ámbito del proyecto, la realización de un estudio de viabilidad, el análisis de los riesgos asociados al proyecto, una estimación del coste del proyecto, su planificación temporal y la asignación de recursos a las distintas etapas del proyecto.

2.2.1. Actividad 1: Delimitación del ámbito del proyecto

Es lo primero que debe establecerse. Qué va a comprender el proyecto, y qué cuestiones se dejarán fuera.

Tan importante es determinar los aspectos que abarcan el proyecto como fijar aquéllos aspectos que no se incluirán en el proyecto, los cuales han de indicarse explícitamente.

Si es necesario, se puede especificar todo aquello que se posponga hasta una versión posterior del sistema.

Si, en algún momento, fuese necesario incluir en el proyecto algún aspecto que no había sido considerado o que ya había sido descartado, es obligatorio reajustar la estimación del coste del proyecto y su planificación temporal.

En él se recogerá la descripción de más alto nivel de la funcionalidad que tendrá nuestro sistema de información, sus características principales y sus objetivos clave. Obviamente, este documento debe formar parte del contrato que se firme con el cliente en el arranque oficial del proyecto.

Debe estar escrito en un lenguaje que cualquiera pueda entender, evitando un vocabulario excesivamente técnico.

Como resultado de la delimitación del ámbito del proyecto debe generarse un documento breve, de 1 ó 2 páginas, en el que se describe el problema que nuestro sistema de información pretende resolver.

2.2.2. Actividad 2: Estudio de viabilidad

Antes de comenzar el proyecto, se debe evaluar la viabilidad económica, técnica y legal del mismo.

Con recursos ilimitados (tiempo y dinero), casi cualquier proyecto se podría llevar a buen puerto.

Deben identificar los principales factores que pueden hacer fracasar el proyecto de desarrollo de software y elementos claves que pueden ayudar a reducir el índice de fracaso.
Lo primero que debe hacerse es plantear si la mejor opción es desarrollar un sistema informatizado o es preferible un sistema manual.

Como resultado del estudio de viabilidad debe generarse un documento que describa el análisis realizado.

2.2.3. Actividad 3: Análisis de riesgos

Debe identificarse los "riesgos" que pueden afectar negativamente al plan del proyecto, estimar la probabilidad de que el riesgo se materialice y analizar su posible impacto en nuestro proyecto.

¿Qué sucedería si algún miembro clave del equipo abandona la empresa, o se va de vacaciones, se pone enfermo, etc.?

¿Qué pasa si al final nos encontramos con algún problema de compatibilidad del sistema que hemos desarrollamos con la configuración de los equipos sobre los que ha de funcionar?

¿Qué pasa si, inadvertidamente, borramos o modificamos erróneamente archivo clave? ¿Si un servidor se daña?

Como resultado del análisis de riesgos debe generarse un documento para que quede claro que esto fue analizado y cuáles fueron las posibles estrategias a considerar.

2.2.4. Actividad 4: Estimación del costo del proyecto

Sin duda, una de las tareas más delicadas de cualquier proyecto de desarrollo de software es la estimación inicial del costo de algo que no se conoce. La realización de malas estimaciones ha sido identificada como una de las dos causas más comunes del fracaso de un proyecto de desarrollo de software.

Debe tenerse en cuenta lo siguiente:

- Haber participado en proyectos de similares características puede ser esencial para poder realizar una buena estimación.

- Nunca se ha de realizar una estimación sobre la marcha, por mucho que haya presión para tener un resultado. Una respuesta apresurada sólo sirve que después no se pueda cumplir con las expectativas que se han creado. Una estimación siempre ha de ser meditada, después de un estudio pormenorizado de los distintos factores que pueden afectar a la realización de un proyecto.

-Cuantos más datos históricos se recopilen y más precisa sea la información de la que se disponga acerca del proyecto, mejor será la estimación.
-Debe descomponerse el proyecto en tareas para una mejor estimación Esto se debe a que, durante el transcurso del proyecto, también han de realizarse otras muchas tareas que probablemente se hayan olvidado incluir en la estimación.
-Considerar recursos técnicos humanos y económicos.

Como resultado del análisis de riesgos debe generarse un documento para especifique la estimación del costo del proyecto lo más real posible.

2.2.5. Actividad 5: Planificación temporal y asignación de recursos

Una vez que se decide seguir adelante con el proyecto, debe planificar su temporización. Una planificación excesivamente detallada (con el proyecto descompuesto en tareas de un día, por ejemplo) puede resultar contraproducente.

Una planificación por semanas suele ser razonable para afrontar con comodidad las contingencias que se puedan presentar, sin tener que estar continuamente reajustando el plan del proyecto.

La planificación del proyecto ha de reajustarse cada vez que cambien las circunstancias del mismo. Si no se ha podido terminar una tarea en el tiempo inicialmente establecido, no vale suponer alegremente que posteriormente se recuperará el tiempo perdido.

Los proyectos se retrasan poco a poco. Deben aprovecharse las primeras señales de alarma, tenerlas en cuenta, y no esconderlas fingiendo que todo marcha según lo previsto.

La planificación es fundamental en la gestión de un proyecto de desarrollo de software. Procure siempre mantener su plan al día. Un plan que no se ajusta a la realidad no sirve de mucho.

Cuando algún retraso indique que posiblemente le será imposible cumplir los plazos establecidos, hable con su cliente. A él le interesa saberlo y, aunque probablemente no se lo agradezca, a la larga resultará beneficioso y usted habrá cumplido con su obligación profesional.

Como resultado de la Planificación temporal y asignación de recursos se debe generar un documento que muestre los tiempos que se van a manejar, las actividades y los recursos asignados

2.2.6. Errores que deben evitarse

-Abreviar las etapas iniciales del proceso de desarrollo de software (planificación y análisis, generalmente) para pasar directamente a la "construcción" del sistema.

-No gestionar adecuadamente los cambios que inevitablemente ocurren durante el proyecto. Tan malo es permitir cualquier cambio de forma indiscriminada como ser excesivamente rígidos a la hora de no admitir cambios aunque éstos sean razonables.

-Reducir la interacción con el cliente, ya que aparentemente sólo se dedica a entorpecer el trabajo con sus continuos cambios de opinión y sus expectativas poco realistas.

-Añadir personal a un proyecto retrasado, por lo general, sólo lo retrasa más. La curva de aprendizaje que se necesita para comenzar a ser productivo ha de tenerse siempre en cuenta.

-Someter a los miembros del equipo a continuas interrupciones durante su jornada de trabajo (llamadas telefónicas, reuniones, consultas...).

Las calidad del trabajo intelectual depende de la capacidad del trabajador de mantener su "estado de flujo" (un estado relajado de inmersión total en un problema que facilita su comprensión y la generación de soluciones).

Se tarda unos 15 minutos en conseguir este estado, por lo que una simple interrupción cada 10 minutos afecta drásticamente al rendimiento del trabajador.

Hacer trabajar horas extra a los miembros del equipo de desarrollo sólo sirve para disminuir su productividad (trabajo realizado por unidad de tiempo).

-No informar de pequeños retrasos pensando que más tarde se recuperará el tiempo perdido. La planificación temporal del proyecto debe ir ajustándose conforme vamos aprendiendo más cosas acerca del problema al que nos enfrentamos.

-Confiar excesivamente en la mejora de rendimiento que se producirá gracias al uso de una nueva herramienta, tecnología o metodología

2.3. Segunda fase Especificación de Requisitos

La fase de especificación de requisitos debe conseguir el catálogo de requisitos del sistema que cubra:

-La definición de los objetivos del sistema.
-Los requisitos de almacenamiento de información.
-La descripción de los actores del sistema.
-Los requisitos funcionales, descritos a través de los casos de uso.
-Los requisitos de interacción, en lo que se recogerá el sistema de navegación de la aplicación y la interacción con el usuario.
-Los requisitos no funcionales. Estos son otra serie de requisitos como los requisitos de comunicaciones del sistema, de seguridad, de portabilidad, etc. que en la mayoría de los sistemas es necesario recoger para garantizar su adecuación a las necesidades.

Esta fase está dividida en actividades que se dividen a su vez en tareas las cuales son las siguientes.

2.3.1. Actividad 1: Obtener información sobre el problema y determinar objetivos

Tarea 1.1- Obtener información sobre el dominio del sistema Tarea 1.2- Preparar y realizar las entrevistas y reuniones Tarea 1.3- Identificar los objetivos del sistema

La primera actividad que hay que realizar es la de definir y conocer el sistema y el entorno de trabajo en el que se va a desarrollar la aplicación. Es importante obtener información sobre el dominio del sistema, estudiando folletos, otros sistemas, etc., diagramas, establecer el plan de entrevistas que permita al equipo de trabajo detectar los objetivos y los requisitos de la aplicación.

De esta forma se define un patrón que se recogerá en el documento de requisitos en el que se van a almacenar de forma estructurada y bajo un identificador único cada uno de los objetivos del sistema.

Estos objetivos van a servir como base para la definición del resto del sistema, de manera que los requisitos que se identifiquen y definan en las fases siguientes tendrán que hacerse en base a un objetivo concreto.

Como resultado se generar un documento de requisitos del sistema. Este documento puede contener Matrices de flujo, Diagramas de Actividades, Diagramas de procesos, etc.

2.3.2. Actividad 2: Identificar y definir los requisitos de almacenamiento de información.

Tarea 2.1- Identificar y definir los requisitos de almacenamiento de información. Tarea 2.2.- Describir la naturaleza de los datos

Consiste en identificar sobre qué conceptos se desea guardar información y cuál va a ser la información concreta almacenada sobre esta.

La naturaleza se refiere a la tipología de la información. En la metodología se definen una serie de naturalezas básicas, como son las cadenas, los enumerados, las imágenes, los sonidos o los números, entre otros. Pero también puede que la naturaleza de un dato sea otro requisito de almacenamiento de información. Como en el caso de los autores. Esta posibilidad permite relacionar unos requisitos de almacenamiento con otros.

Cuando los datos concretos no tengan como naturaleza una de las naturalezas básicas u otro requisito de almacenamiento, será necesario definir una nueva naturaleza. Es el caso del dato código del bien de nuestro ejemplo.

Una vez definidos los requisitos de almacenamiento y la naturaleza de los datos, se tienen sentadas las bases para la definición de la estructura conceptual de la aplicación.

2.3.3. Actividad 3: Definición de actores

Tarea 3.1- Definir los actores básicos del sistema Tarea 3.2- Definir la generalización de los actores Tarea 3.3- Definir la incompatibilidad de actores Tarea 3.4- Definir los actores derivados

Una vez definidos los requisitos de almacenamiento de información del sistema hay que identificar los actores capaces de interactuar con el mismo, siendo éste el objetivo de esta tercera actividad.

Un actor es una abstracción de una persona externa, de un proceso o de una cosa que interactúa con el sistema. Cada actor define un rol que los usuarios asumen cuando interactúan con el sistema. En esta tarea se definirá qué roles pueden aparecer, es decir, los actores de la aplicación, pero además se van a analizar las incompatibilidades que presentan y las relaciones de generalización entre ellos.

Un actor básico es todo actor que se puede identificar de forma individual atendiendo a algún tipo de criterio o punto de vista a la hora de interaccionar con el sistema software.

2.3.4. Actividad 4: Identificar y definir los requisitos funcionales del sistema

Tarea 4.1- Definir los diagramas de casos de uso Tarea 4.2- Describir los casos de uso

Los requisitos funcionales van a responder a la pregunta de ¿qué podrá hacer el sistema con la información que almacena? Esta pregunta responde los Casos de Uso, técnica que se usará para capturar estas necesidades.

Estos diagramas se deben describir mediante una información gráfica (diagramas de casos de uso) y una información textual.

Ejemplo de Caso de Uso

En los casos de uso aparecen dos elementos importantes, el caso de uso en sí y los actores. Los actores se definieron en la actividad anterior, así que aquí se hará referencia a esas definiciones.

Los diferentes Casos de uso del sistema se especifican en el Formato Especificaciones Casos de Uso.

2.3.5. Actividad 5: Identificar y definir los requisitos de interacción

Tarea 5.1- Identificar y definir los prototipos de visualización

En este punto, ya se sabe que información debe recogerse y qué hay que almacenar en el sistema, quién va a usar el sistema y qué se puede hacer en el sistema. Sin embargo, para los sistemas de información global, la interfaz es un aspecto crítico que es fundamental en el desarrollo.

Un requisito de interacción va a ser una forma de representar como se va a mostrar al usuario la información. Basándose en criterios establecidos por el cliente, los datos concretos de los requisitos de almacenamiento de información se van a mostrar agrupados en diferentes prototipos de visualización.

Cada uno de estos requisitos de interacción llevará asociado una funcionalidad. La funcionalidad vendrá dada por cada uno de los requisitos funcionales que se puedan ejecutar en ese requisito de interacción.

Los requisitos de interacción van a estar compuestos por dos aspectos: los criterios de búsqueda y los prototipos de visualización. Los primeros van a usar un lenguaje seminatural en el que se va a recoger cómo el usuario quiere recuperar la información.

Los segundos van a definir cómo se mostrará la información al usuario y la funcionalidad que tiene asociada esa muestra de información.

Debe tenerse en cuenta el manejo de imagen corporativo que esté activo en el momento en la institución.

2.3.6. Actividad 6: Identificar y definir los requisitos no funcionales del sistema

Tarea 6.1.- Definir los requisitos no funcionales

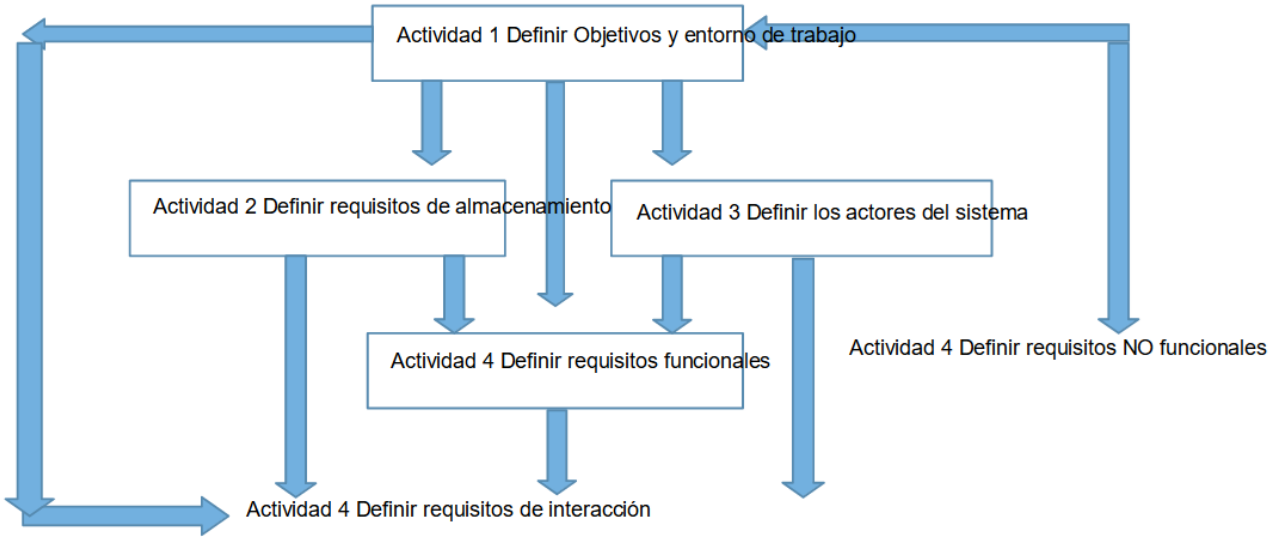
Para completar la especificación de requisitos, es necesario revisar de nuevo los objetivos para determinar otros requisitos que por su carácter no hayan sido catalogados anteriormente.

En el gráfico se muestra la relación de cada uno de los requisitos, y su interación entre sí para generar un resultado

Como resultado se generar un documento de requisitos del sistema

2.4. Tercer fase Análisis

En esta fase debe conseguirse un modelo de clases que represente al sistema. Este modelo irá acompañado por un modelo dinámico cuando resulte necesario, así como por una estructuración en paquetes cuando su complejidad sea alta.



Además, en esta fase de análisis debe hacerse un refinamiento de los casos de uso y asignar responsabilidades y participaciones de las clases de análisis. También debe consolidarse la navegación y proponerse los primeros prototipos de interfaz.

2.4.1. Actividad 1: Construir un modelo conceptual de análisis.

Tarea 1.1- Identificar y definir las clases del sistema.

Tarea 1.2- Agrupar las clases del modelo en paquetes y establecer relaciones entre ellos si es necesario.

Tarea 1.3- Realizar el refinamiento de los casos de uso para concretizarlos y asignar

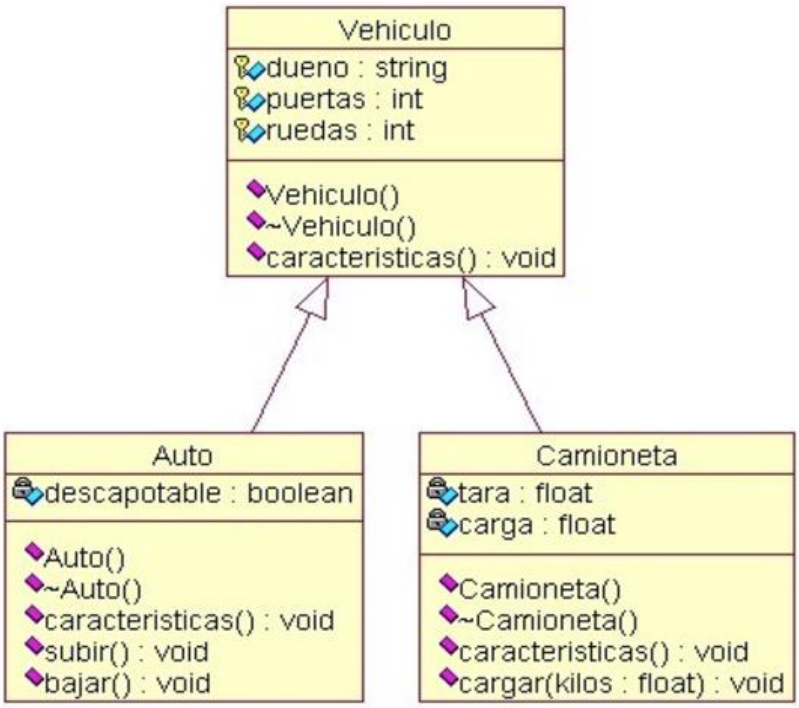
responsabilidades y participación en ellos de las clases de análisis.

Tarea 1.4- Definir el modelo dinámico de las clases de análisis

Partiendo de los requisitos de almacenamiento de información y de definición de los actores, se debe realizar el modelo conceptual del sistema. Esto consiste en realizar un modelo de clases, que mediante la nomenclatura de UML represente la estructura conceptual de la aplicación.

EJEMPLO DIAGRAMA DE CLASES DE ANÁLISIS

2.4.2. Actividad 2: Refinar el modelo de navegación del sistema.



Cuando ya se ha representado el modelo conceptual del sistema, es necesario estudiar y refinar el modelo de navegación del sistema que se está definiendo.

En los requisitos de interacción se capturó cómo se va a mostrar la información al usuario y cómo se puede navegar a través de estos módulos de información. Sin embargo, para que la estructura de la navegación sea consistente debe cubrir una serie de aspectos. Por ejemplo, el usuario debe tener la posibilidad de llegar a toda esta información desde un punto que podríamos definir como pantalla inicial.

2.4.3. Actividad 3: Definir los prototipos de interfaz.

Partiendo de los requisitos de interacción y de los elementos de navegación definidos en la actividad anterior se va a hacer una definición de la interfaz de la aplicación.

Estos prototipos de interfaz van a mostrar qué información se ofrece al usuario y en qué orden se ofrece, así como las navegaciones que se ofrecen.

Como resultado se generar un documento con el modelo conceptual del sistema.

Ejemplo de Prototipo de Interfaz

2.5. Cuarta fase Diseño

En el flujo de trabajo de diseño se parte del modelo de clases de análisis conseguido en el flujo anterior, así como del análisis hecho a los casos de uso y de la agrupación en paquetes de análisis. A partir de estos productos se realiza el diseño de la arquitectura del sistema, hacer un diseño de los casos de uso, conseguir un modelo de clases de diseño y realizar una división del sistema en subsistemas. Pero además, hay que recoger y diseñar los aspectos de navegación y de interfaz de usuario.

2.5.1. Actividad 1: Diseñar el modelo básico.

Tarea 1.1- Diseñar la arquitectura del sistema.
Tarea 1.2- Diseñar los casos de uso y los requisitos no funcionales. Tarea 1.3- Dividir el sistema en subsistemas.
Tarea 1.4- Diseñar el modelo de clases básico de análisis.

El diseño del modelo básico consistirá en obtener el diseño del sistema pero dejando al margen todo lo que serían los aspectos de navegación y de interfaz.

Para un mismo diseño básico puede haber muchos diseños de navegación y cada uno de estos puede tener definidas diferentes interfaces. De esta forma si cambiamos la interfaz o la navegación el modelo básico puede quedar igual.

En el diseño básico, una de las tareas a realizar es el diseño de la arquitectura. Para realizar el diseño de la arquitectura en un sistema de información global, hay que tener en cuenta que los soportes de estos sistemas, en la mayoría de los casos, se caracterizan por estar dispersos en la red, por estar implantados en sistemas heterogéneos y por la ausencia de estándares para el acceso a los mismos. Por ello, las propuestas de arquitecturas que se den en el marco de estos sistemas, deben contemplar los siguientes aspectos:

Almacenamiento Clasificación
Interfaces para la presentación de resultados Distribución del contenido de la biblioteca al usuario final Administración y control de acceso.

El diseño de la arquitectura no debe verse afectado por aspectos de navegación o interfaz.

Cuando diseñamos los casos de uso y conocemos las necesidades de almacenamiento que hay que tener, es hora de seleccionar la estrategia básica para implementar los almacenes de datos en términos de estructuras de datos, archivos y bases de datos.

Debido a la multiplicidad de medios y naturalezas que se usan, no siempre es posible encontrar un sistema capaz de dar soporte a todos ellos. Las bases de datos de última generación que permiten almacenar y recuperar imágenes o sonido de la misma forma que se recuperan los tipos básicos como el texto los booleanos, están sólo naciendo. Por ello, aquí podemos proponer múltiples sistemas de almacenamiento que den soporte a nuestras necesidades.

2.5.2. Actividad 2: Diseñar el modelo de navegación

Tarea 2.1- Diseñar el modelo de clases de navegación Tarea 2.2- Diseñar los contextos de navegación

A la hora de realizar un modelo de navegación, hay que tener en cuenta:
Qué objetos del modelo básico van a ser navegables. Esto va a estar íntimamente relacionado con los nodos definidos en la especificación de requisitos de navegación

Qué tipo de relaciones y estructuras de composición hay entre estos objetos navegables.

De los objetos navegables, habrá que estudiar en qué forma se mostrarán dependiendo del contexto en el que nos movamos.

Las conexiones entre los distintos objetos y las posibilidades de navegación de un objeto a otro. Esto va a estar bastante relacionado con los enlaces entre nodos que se detallan en la especificación de los requisitos navegación.

Se proponen dos modelos para recoger los aspectos de navegación: el esquema de Clases de navegación y el diagrama de contexto de navegación.

2.5.3. Actividad 3: Diseñar la interfaz abstracta.

El diseño de la interfaz abstracta se refiere a revisar los prototipos que se han definido en la fase de análisis y enriquecerlos con todos los nuevos aspectos de diseño conceptual y de navegación.

Tarea 3.1- Diseñar los prototipos de pantallas. Tarea 3.2- Diseñar el sistema dinámico de pantallas.

2.6. Quinta fase Desarrollo

La tarea de desarrollo consiste prácticamente en traducir a código lo que se ha diseñado, crear manuales de usuario, técnicos u otros.

Así por ejemplo, del diseño básico se extrae información sobre la arquitectura del sistema, los sistemas de almacenamiento a usar y la estructura de la base de datos que se le asociará.
Con el diseño de la interfaz abstracta y el diseño de la navegación se puede conocer qué pantallas hay que realizar, los eventos que la afectan, quién puede trabajar con ellas y cómo se puede acceder y salir de cada una de estas pantallas. Sí que habría que aplicar técnicas para hacer el diseño visual de estas pantallas, pero nada más.

Esta fase de desarrollo de acuerdo a políticas institucionales debe estar centralizado en el CISC.

2.6.1. Lenguajes de programación

Los sistemas informáticos deben llevarse a cabo en cualquiera de los lenguajes de programación siguientes:

2.6.2. Herramientas IDE

Las herramientas IDE que se establecen para el desarrollo de aplicaciones (editor de código, compilador, depurador y constructor de interface gráfica), comprenden las siguientes:

Escritorio/Movil
Web
Netbeans / Eclipse / Jdeveloper etc...

2.6.3. Servidores de Aplicación

Para la implementación de las aplicaciones en ambientes de pruebas y producción se debe utilizar alguno de los servidores de aplicación siguientes:

2.6.4. Sistemas operativos

Para proveer servicios para la ejecución de los sistemas informáticos, se establece cualquiera de los sistemas operativos siguientes:

- Escritorio
- Web
- Movil
- Windows
- Mc OS
- Windows Server Linux Red Hat
- Linux Centos
- Windows Android
- IOS

2.6.5. Interface de usuario

La interface de usuario del sistema informático desarrollado debe cumplir con las disposiciones relacionadas con la publicación de información del Instituto en intranet, Internet y las relativas a la imagen institucional. Los estándares para los elementos de imagen, audio y video para la presentación de información son:

- Imagen
- Audio y video
- Gif
- Jpeg Png
- Windows media video Mpeg-4

2.6.6. Herramientas de reportes

Las herramientas de reporte serán aquéllas que están incorporadas en los IDE a los que hace referencia este Manual. Como opciones se incluyen Jasper Report, iReports entre otros.

2.6.7. Manejadores BD

Los sistemas manejadores de bases de datos sugeridos en el estándar son los siguientes:

- MongoDB
- Oracle
- PostgreSQL
- MySql
- SqlServer
- SQLite

2.6.8. Nomenclatura de los objetos de Base de Datos. Rol.

Deben establecerse los privilegios de acceso a la bases de datos de acuerdo al perfil de usuario y/o funcionalidad de la aplicación.

Para identificar un rol se deberá seguir el patrón:
R_APL_XXX Donde:
“R_”: es el prefijo de identificación del rol
APL: siglas de la aplicación
XXX: nombre representativo del rol.

Tablas.
Generales.
Definir nombres claros, que describan el contenido de la entidad o tabla. La longitud máxima será de 30 caracteres.

La denominación debe ser un sustantivo en singular; sólo en aquellos casos en donde el singular no represente correctamente el contenido de la misma, se podrán utilizar nombres en plural.

Tipos de tablas (Nomenclatura de tablas).
Modelo relacional

“TR_” = Entidades que representan los registros con datos detallados

“TC_” = Entidades que representan catálogos que describen los valores de una variable de una entidad tipo TR_

“TI_” = Entidades que representan la relación muchos a muchos entre dos tablas.

En este caso el nombre de la tabla debe incluir los nombres de las tablas que relaciona separadas por guion bajo (“_”).

Metadatos
“TM_” = Entidades que representan metadatos.

Temporales
“TT_” = Entidades utilizadas temporalmente por uno o varios procesos.

Atributo.
El nombre del atributo debe ser claro y representativo al dato que contiene, con un tamaño máximo de 25 caracteres.

No debe contener caracteres especiales excepto el guion bajo (_).
El orden de los atributos al interior de la entidad, deben ser de acuerdo al orden de captación de la información correspondiente.

-Únicamente a los atributos que formen parte de la llave primaria se les agregará el prefijo “ID_” seguido del nombre de la entidad sin el prefijo (“TR_”, “TC_”, etc.).

-En tablas de catálogos debe utilizar el prefijo “DES_” para aquellos atributos que representan su descripción.

Procedimiento.
-El nombre del procedimiento iniciará con el prefijo “PR_” y será de la siguiente manera:
PR_NOMBRE Donde:
“PR_”: indica que es un procedimiento almacenado NOMBRE: es el nombre del procedimiento

Función.
-El nombre de la función iniciará con el prefijo “FN_” y será de la siguiente manera: FN_NOMBRE
Donde:
“FN_”: indica que es una función. NOMBRE: es el nombre de la función.
Paquete.
- El nombre del paquete iniciará con el prefijo “PQ_” y será de la siguiente manera:

PQ_NOMBRE Donde:
“PQ_”: indica que es un paquete. NOMBRE: es el nombre del paquete.

-Los nombres de procedimientos, funciones o paquetes deben ser claros y descriptivos a las tareas que realizarán. Dentro del script de creación del procedimiento o función se debe agregar como comentario lo siguiente:

Descripción: Texto que detalla la acción o finalidad del procedimiento o función. Parámetros: Valores que recibe el procedimiento o función. Para cada parámetro debe considerarse:

Vista.
Nombre. Tipo de dato.
Longitud (considerando el número de decimales). Si es de entrada y/o salida.
Resultado: En el caso de las funciones, el dato que se genera al ejecutarla
El nombre de la vista seguirá los estándares de nomenclatura de una tabla, con la variante de que en lugar de comenzar con “T” se comenzará con “V”.
Ejemplo, en lugar de usar “TR_” se usará “VR_”.
Índice.
En índices de campos que no son llave primaria o foránea, el nombre de un índice debe constituirse por el prefijo “I_”, seguido por las primeras letras de los nombres de cada una de las columnas que involucra, omitiendo cualquier prefijo, de la siguiente manera:
I_XXX_YYY Donde I = Índice
XXX = campo 1 YYY = campo n

Secuencia.
Para aquellas entidades donde no exista un atributo de llave primaria, se deberá agregar un atributo de secuencia que servirá como identificador único, nombrándolo
“SEC_”, seguido del nombre de la entidad, de la manera siguiente: SEC_NOMBRE Donde:
“SEC_”: indica que es una secuencia. NOMBRE: es el nombre de la secuencia

Sinónimo.
El nombre de un sinónimo debe constituirse por el prefijo “S_”, seguido por el nombre del objeto (tabla, procedimiento, vista, etc.) al cual hace referencia dicho sinónimo.
S_NOMBRE Donde:
“S_”: indica que es un sinónimo de un objeto de base de datos.
NOMBRE: es el nombre del objeto de bases de datos al cual hace referencia dicho sinónimo.

Diccionario de Datos
Es un listado organizado de todos los datos pertinentes al sistema con definiciones precisas y rigurosas para que tanto el usuario como el analista tengan un entendimiento en común de todas las entradas, salidas, componentes y cálculos.

Un diccionario de datos contiene las características lógicas de los datos que se van a utilizar en un sistema, incluyendo nombre, descripción, alias, contenido y organización.

El objetivo de un diccionario de datos es dar precisión sobre los datos que se manejan en un sistema, evitando así malas interpretaciones o ambigüedades.

Estos diccionarios se desarrollan durante el análisis de flujo de datos y su contenido también se emplea durante el diseño del proyecto en general.

Cada entrada en el diccionario de dato consiste en un conjunto de detalles que describen los datos utilizados o producidos en el sistema. Cada artículo se identifica por:

- Nombre de dato: Para distinguir un dato de otro, se deben asignar nombres significativos que se utilizan para tener una referencia de cada elemento a través del proceso total de desarrollo de sistemas.
- Descripción del dato: Establece brevemente lo que representa el dato en el sistema. Es importante que las descripciones se escriban suponiendo que la gente que la lea no conoce nada en relación del sistema. Por lo tanto deben evitarse términos especiales, para que todas las palabras sean entendibles para el lector.
- Sinónimo o alias: Es una alternativa de nombre para un campo. Con frecuencia el mismo dato puede conocerse con diferentes nombres, dependiendo de quién lo utilice.

- Longitud de campo: Cuando las características del diseño del sistema se ejecuten más tarde en el proceso de desarrollo del sistema, será importante conocer la cantidad de espacio que necesita para cada dato.

- Valores de datos: En algunos procesos solo se permiten valores de datos específicos. Por ejemplo, en muchas compañías con frecuencia los números de orden de compra se proporcionan con un prefijo de una letra para indicar el departamento del origen.

2.6.9. Estructura de Carpetas y Subcarpetas.

La estructura de carpetas y subcarpetas de desarrollo informático se organizará bajo las siguientes consideraciones:

Código fuente y librerías: Estas carpetas y sus subcarpetas se organizarán de acuerdo a la arquitectura de desarrollo que se esté utilizando.

Recursos: Esta carpeta contendrá los artefactos como textos, archivos de audio, iconos, imágenes, entre otros que utiliza el sistema informático.

Temporales: Esta carpeta contendrá los archivos transitorios para los procesos.

Configuración: Esta carpeta contendrá los archivos de parámetros requeridos por el sistema para su inicialización o para determinar comportamientos específicos.

Documentación: Esta carpeta contendrá los principales documentos de soporte al sistema.

Salidas: Esta carpeta contendrá aquellos productos que resulten de los diversos procesos de la aplicación que deban ser conservados. Esta carpeta debe estar ubicada en una trayectoria externa a la estructura del sistema.

2.6.10. Respaldos y restauraciones.

El respaldo de un proyecto de desarrollo de sistemas informáticos incluye:
Código fuente en su última versión conforme a la versión publicada en ambiente de producción.

Archivos de recursos, librerías, componentes y otros elementos utilizados por el sistema
Descripción de la estructura de carpetas del proyecto.
Descripciones de las estructuras de información que se utilizan.

Consideraciones y archivos que sean necesarios para la reconstrucción y restauración del sistema.

Los medios para el respaldo de lo definido en el inciso anterior deben ser externos al equipo de trabajo, como: servicios ftp, discos duros externos o sistemas SAN/NAS propios del Instituto.

Los respaldos y los procedimientos de restauración deben probarse conforme a los tiempos y períodos que defina el responsable de la información, para verificar que sean funcionales y que los medios utilizados continúen vigentes.

Los medios de almacenamiento deben encontrarse adecuadamente identificados, a través de una etiqueta que maneje como mínimo la fecha de generación del respaldo, nombre de la aplicación, tipo de información y periodo que se está respaldando.

Los procesos de respaldo deben coordinarse con los administradores de los servidores (de aplicación o base de datos) para que se ejecuten de forma programada.

Los respaldos generados deberán conservarse en al menos tres ciclos: diario, semanal, mensual.

Diario: Debe tomarse una copia de respaldo incremental de lunes a viernes. La copia de cada día tendrá una rotación quincenal.

Semanal: Debe tomarse una copia de respaldo el día domingo que contiene todo lo de la semana. La copia de cada semana tendrá una rotación mensual.

Mensual: Debe tomarse una copia de respaldo al final del mes la cual no tiene rotación alguna y se archiva.

Los formatos de los respaldos de bases de datos a utilizar son:
Oracle: DMP, TXT
PostgreSQL: .BACKUP .SQL .TXT

2.7. Sexta fase Pruebas e Implementación

2.7.1. Plan de pruebas

Aunque la definición concreta de las pruebas que se deben aplicar a un sistema deberá concretarse en el momento de realizar su diseño, toda propuesta metodológica debe definir las posibles estrategias a seguir, las técnicas a utilizar para realizarlas y el momento de hacer uso de cada una de ellas.

Se debe elaborar, diseñar e implementar el plan de pruebas. Este plan de pruebas no solo debe recoger las pruebas a realizar, además debe indicar el orden de realización.

Una vez realizado, hay que ejecutar el plan de pruebas y elaborar una memoria de resultados del mismo.

Un plan de pruebas incluye:

1. Identificador del plan.

Preferiblemente de alguna forma mnemónica que permita relacionarlo con su alcance, por ej. PP-01 (Plan de pruebas 01).

2. Alcance

Indica el tipo de prueba y las propiedades/elementos del software a ser probado.

3. Ítems a probar

Indica la configuración a probar y las condiciones mínimas que debe cumplir para comenzar a aplicarle el plan. Por un lado, es difícil y riesgoso probar una configuración que aún reporta fallas; por otro lado, si esperamos a que todos los módulos estén perfectos, puede que detectemos fallas graves demasiado tarde.

4. Estrategia

Describe la técnica, patrón y/o herramientas a utilizarse en el diseño de los casos de prueba. En lo posible la estrategia debe precisar el número mínimo de casos de prueba a diseñar.

5. Recursos

Especifica las propiedades necesarias y deseables del ambiente de prueba, incluyendo las características del hardware, el software de sistemas (p. ej. el sistema de operativo), cualquier otro software necesario para llevar a cabo las pruebas, así como los puntos específicos del software a probar.

2.7.2. Implementación

La fase de implementación de un sistema es la fase más costosa y que consume más tiempo de todo el ciclo de vida, es costosa porque muchas personas, herramientas y recursos, están involucrados en el proceso.

Esta etapa consume mucho tiempo porque se completa todo el trabajo realizado previamente durante el ciclo de vida.

Durante la implementación las especificaciones del diseño físico deben cumplir con el dicho diseño, el código es probado y la mayoría de los errores deben ser detectados y corregidos; posteriormente el sistema es instalado y las localidades de los usuarios son preparadas para el nuevo sistema y los usuarios deben acostumbrarse a éste.

Durante la implementación son muchos los aspectos organizacionales que deben ser considerados como:

- Instalación.
- Documentación.
- Capacitación.
- Soporte.

Puesto que la implementación de los sistemas de información acompaña inevitablemente la introducción de una nueva tecnología administrativa, es necesario tomar todas las precauciones posibles para que tenga éxito dicha fase de implementación.

La probabilidad de éxito en la implementación del sistema de información está directamente relacionada con la posición organizacional del patrocinador de más alta jerarquía. Por esta razón, se recomienda siempre asegurar el compromiso abierto de la alta gerencia para apoyar la implementación de este proyecto.

Cuando los altos niveles de la organización están directamente comprometidos con ello, existen mayores probabilidades de éxito.

El compromiso de la alta gerencia significa algo más que aprobación, supone participación en forma periódica para asegurar que los objetivos del proyecto se están alcanzando y que su filosofía e intenciones se reflejan en forma adecuada.

2.7.3. Mantenimiento y mejora continua

El mantenimiento se encarga de corregir las fallas detectadas durante la operación de un sistema de información, así como el de realizar las modificaciones pertinentes a los nuevos requerimientos que se van presentando.

Las principales funciones que se deben realizar en el mantenimiento de un sistema de información son las siguientes:

Planeación. Consiste en la detección (falla) o planteamiento del nuevo requerimiento.

Definición de los Ajustes a realizar. Debe realizarse un análisis del cambio a efectuar, considerando:

- Grado de Dificultad. Debe estimarse el esfuerzo, costo y tiempo que se llevaría la modificación, teniéndose en cuenta el impacto que pueda ocasionar los cambios

- Factibilidad. En base a los recursos requeridos y los disponibles, se decide la posibilidad de aplicar la modificación.

Ejecución de las Modificaciones. Se llevan a cabo las modificaciones necesarias para satisfacer al requerimiento planteado.

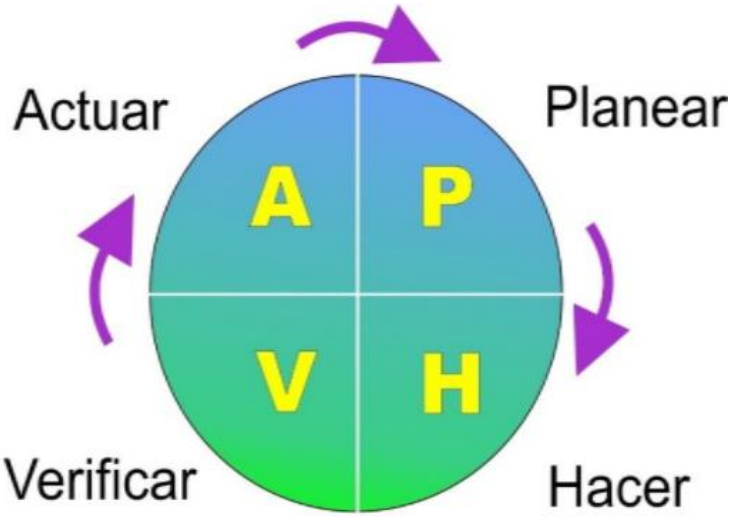
Pruebas de la Modificaciones. Se prueban exhaustivamente los cambios hechos al sistema, procurando detectar y corregir posibles errores.

Liberación. Se actualiza la documentación y se procede a informar y adiestrar al personal involucrado en la operación del sistema.

Para lograr un adecuado mantenimiento al sistema, se hacen las siguientes sugerencias:

• Tener un seguimiento del sistema, a efecto de evaluar la alimentación de datos, ejecución de procesos, revisión de resultados y respaldo de la información.
• Evaluar el empleo de recursos y tiempo de respuesta que ocupa el sistema en su operación, para optimizarla.
• Analizar detalladamente las fallas detectadas, así como los nuevos requerimientos antes de iniciar cualquier alteración.
• Incorporar las experiencias y avances tecnológicos para la optimización del sistema.
• Respaldar el sistema antes de comenzar los cambios.
• Durante la modificación, respetar las normas, estándares y procedimientos que han respaldado la construcción del sistema.
• Probar detenidamente las modificaciones realizadas

Esta metodología se encuentra enmarcada dentro del ciclo planificar-hacer-verificar-actuar.



Planificar: Establecer los objetivos y procesos necesarios para obtener los resultados, de conformidad con los requisitos del cliente y las políticas de la organización.
Hacer: Implementar procesos para alcanzar los objetivos.

Verificar: Realizar seguimiento y medir los procesos y los productos en relación con las políticas, los objetivos y los requisitos, reportando los resultados alcanzados.
Actuar: Realizar acciones para promover la mejora del desempeño del (los) proceso(s).

Como herramienta para la mejora continua en donde se desarrollan ciclos de mejora en todos los niveles, donde se ejecutan las funciones y los procesos de la organización, afectados por el sistema. Es una herramienta de simple aplicación y, cuando se utiliza adecuadamente, puede ayudar mucho en la realización de las actividades de una manera más organizada y eficaz.

Con la aplicación de esta modalidad circular y dinámica, el proceso o proyecto no termina cuando se obtiene el resultado deseado, sino que más bien, se inicia un nuevo desafío no sólo para el responsable de cada proceso o proyecto emprendido, sino también para la propia organización.

Normalización de Bases de Datos y Técnicas de diseño

Uno de los factores mas importantes en la creación de páginas web dinámicas es el diseño de las Bases de Datos (BD). Si tus tablas no estan correctamente diseñadas, te pueden causar un montón de dolores de cabeza cuando tengas de realizar complicadísimas llamadas SQL en el código PHP para extraer los datos que necesitas. Si conoces como establecer las relaciones entre los datos y la normalización de estos, estarás preparado para comenzar a desarrollar tu aplicación en PHP.

Si trabajas con MySQL o con Oracle, debes conocer los métodos de normalización del diseño de las tablas en tu sistema de BD relacional. Estos métodos pueden ayudarte a hacer tu código PHP mas fácil de comprender, ampliar, y en determinados casos, incluso hacer tu aplicación mas rápida.

Básicamente, las reglas de Normalización están encaminadas a eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas. Más tarde explicaré lo que esto significa mientras vemos los cinco pasos progresivos para normalizar, tienes que tener en cuenta que debes crear una BD funcional y eficiente. Tambien detallaré los tipos de relaciones que tu estructura de datos puede tener.

Digamos que queremos crear una tabla con la información de usuarios, y los datos a guardar son el nombre, la empresa, la dirección de la empresa y algun e-mail, o bien URL si las tienen. En principio comenzarias definiendo la estructura de una tabla como esta:

Formalización CERO

usuarios				
nombre	empresa	direccion_empresa	url1	url2
Joe	ABC	1 Work Lane	abc.com	xyz.com
Jill	XYZ	1 Job Street	abc.com	xyz.com

Diríamos que la anterior tabla esta en nivel de Formalizacion Cero porque ninguna de nuestras reglas de normalización ha sido aplicada. Observa los campos url1 y url2 -- ¿Qué haremos cuando en nuestra aplicación necesitemos una tercera url ? ¿ Quieres tener que añadir otro campo/columna a tu tabla y tener que reprogramar toda la entrada de datos de tu código PHP ? Obviamente no, tu quieres crear un sistema funcional que pueda crecer y adaptarse fácilmente a los nuevos requisitos. Hechemos un vistazo a las reglas del Primer Nivel de Formalización-Normalización, y las aplicaremos a nuestra tabla.

Primer nivel de Formalización/Normalización. (F/N)

1 Eliminar los grupos repetitivos de la tablas individuales.
2 Crear una tabla separada por cada grupo de datos relacionados.
3 Identificar cada grupo de datos relacionados con una clave primaria.

¿ Ves que estamos rompiendo la primera regla cuando repetimos los campos url1 y url2 ? ¿ Y que pasa con la tercera regla, la clave primaria ? La regla tres básicamente significa que tenemos que poner una campo tipo contador autoincrementable para cada registro. De otra forma, ¿ Qué pasaria si tuvieramos dos usuarios llamados Joe y queremos diferenciarlos. Una vez que aplicaramos el primer nivel de F/N nos encontrariamos con la siguiente tabla:

usuarios				
userId	nombre	empresa	direccion_empresa	url
1	Joe	ABC	1 Work Lane	abc.com
1	Joe	ABC	1 Work Lane	xyz.com
2	Jill	XYZ	1 Job Street	abc.com
2	Jill	XYZ	1 Job Street	xyz.com

Ahora diremos que nuestra tabla está en el primer nivel de F/N. Hemos solucionado el problema de la limitación del campo url. Pero sin embargo vemos otros problemas....Cada vez que introducimos un nuevo registro en la **tabla usuarios**, tenemos que duplicar el nombre de la empresa y del usuario. No sólo nuestra BD crecerá muchísimo, sino que será muy facil que la BD se corrompa si escribimos mal alguno de los datos redundantes. Aplicaremos pues el segundo nivel de F/N:

Segundo nivel de F/N

1 Crear tablas separadas para aquellos grupos de datos que se aplican a varios registros.
2 Relacionar estas tablas mediante una clave externa.

Hemos separado el campo url en otra tabla, de forma que podemos añadir más en el futuro si tener que duplicar los demás datos. Tambien vamos a usar nuestra clave primaria para relacionar estos campos:

usuarios			
userId	nombre	empresa	direccion_empresa
1	Joe	ABC	1 Work Lane
2	Jill	XYZ	1 Job Street

urls		
urlId	relUserId	url

1	1	abc.com
2	1	xyz.com
3	2	abc.com
4	2	xyz.com

Vale, hemos creado tablas separadas y la clave primaria en la **tabla usuarios**, userId, esta relacionada ahora con la clave externa en la **tabla urls**, relUserId. Esto esta mejor. ¿ Pero que ocurre cuando queremos añadir otro empleado a la empresa ABC ? ¿ o 200 empleados ? Ahora tenemos el nombre de la empresa y su dirección duplicandose, otra situación que puede inducirnos a introducir errores en nuestros datos. Así que tendrémos que aplicar el tercer nivel de F/N:

Tercer nivel de F/N.

1. Eliminar aquellos campos que no dependan de la clave.

Nuestro nombre de empresa y su dirección no tienen nada que ver con el campo userId, asi que tienen que tener su propio empresaId:

usuarios		
userId	nombre	relEmpresaId
1	Joe	1
2	Jill	2

empresas		
emprId	empresa	direccion_empresa
1	ABC	1 Work Lane
2	XYZ	1 Job Street

urls		
urlId	RelUserId	url
1	1	abc.com
2	1	xyz.com
3	2	abc.com
4	2	xyz.com

Ahora tenemos la clave primaria emprId en la **tabla empresas** relacionada con la clave externa recEmpresaId en la **tabla usuarios**, y podemos añadir 200 usuarios mientras que sólo tenemos que insertar el nombre 'ABC' una vez.

Nuestras tablas de usuarios y urls pueden crecer todo lo que quieran sin duplicación ni corrupción de datos. La mayoria de los desarrolladores dicen que el tercer nivel de F/N es suficiente, que nuestro esquema de datos puede manejar facilmente los datos obtenidos de una cualquier empresa en su totalidad, y en la mayoria de los casos esto será cierto.

Pero hechemos un vistazo a nuestro campo urls - ¿ Ves duplicación de datos ? Esto es perfectamente aceptable si la entrada de datos de este campo es solicitada al usuario en nuestra apliación para que teclee libremente su url, y por lo tanto es sólo una coincidencia que Joe y Jill teclearon la misma url. ¿ Pero que pasa si en lugar de entrada libre de texto usáramos un menú desplegable con 20 o incluso más urls predefinidas ? Entonces tendríamos que llevar nuestro diseño de BD al siguiente nivel de F/N, el cuarto, muchos desarrolladores lo pasan por alto porque depende mucho de un tipo muy específico de relación, la relación 'varios-con-varios', la cual aún no hemos encontrado en nuestra aplicación.

Relaciones entre los Datos

Antes de definir el cuarto nivel de F/N, veremos tres tipos de relaciones entre los datos: uno-a-uno, uno-con-varios y varios-con-varios. Mira la **tabla usuarios** en el Primer Nivel de F/N del ejemplo de arriba. Por un momento imaginámos que ponemos el campo url en una tabla separada, y cada vez que introducimos un registro en la **tabla usuarios** tambien introducimos una sola fila en la **tabla urls**. Entonces tendríamos una relacion uno-a-uno: cada fila en la tabla usuarios tendría exactamente una fila correspondiente en la **tabla urls**. Para los propósitos de nuestra aplicación no sería útil la normalización.

Ahora mira las tablas en el ejemplo del Segundo Nivel de F/N. Nuestras tablas permiten a un sólo usuario tener asociadas varias urls. Esta es una relación uno-con-varios, el tipo de relación más común, y hasta que se nos presentó el dilema del Tercer Nivel de F/N. la única clase de relación que necesitamos.

La relación varios-con-varios, sin embargo, es ligeramente más compleja. Observa en nuestro ejemplo del Tercer Nivel de F/N que tenemos a un usuario relacionado con varias urls. Como dijímos, vamos a cambiar la estructura para permitir que varios usuarios esten relacionados con varias urls y así tendremos una relación varios-con-varios. Veamos como quedarían nuestras tablas antes de seguir con este planteamiento:

usuarios		
userId	nombre	relEmpresaId
1	Joe	1
2	Jill	2

empresas		
emprId	empresa	direccion_empresa
1	ABC	1 Work Lane

2	XYZ	1 Job Street

urls	
urlId	url
1	abc.com
2	xyz.com

url_relations		
relationId	relatedUrlId	relatedUserId
1	1	1
2	1	2
3	2	1
4	2	2

Para disminuir la duplicación de los datos (este proceso nos llevará al Cuarto Nivel de F/N), hemos creado una tabla que sólo tiene claves externas y primarias url_relations. Hemos sido capaces de remover la entradas duplicadas en la tabla urls creando la tabla url_relations. Ahora podemos expresar fielmente la relación que ambos Joe and Jill tienen entre cada uno de ellos, y entre ambos, las urls. Así que veamos exáctamente que es lo que el Cuarto Nivel de F/N. supone:

Cuarto Nivel de F/N.

1. En las relaciones varios-con-varios, entidades independientes no pueden ser almacenadas en la misma tabla.

Ya que sólo se aplica a las relaciones varios-con-varios, la mayoria de los desarrolladores pueden ignorar esta regla de forma correcta. Pero es muy útil en ciertas situaciones, tal como esta. Hemos optimizado nuestra **tabla urls** eliminado duplicados y hemos puesto las relaciones en su propia tabla.

Os voy a poner un ejemplo prático, ahora podemos seleccionar todas las urls de Joe realizando la siguiente instrucción SQL:

SELECT nombre, url FROM usuarios, urls, url_relations WHERE url_relations.relatedUserId = 1 AND usuarios.userId = 1 AND urls.urlId = url_relations.relatedUrlId

Y si queremos recorrer todas las urls de cada uno de los usuarios, hariamos algo así:

SELECT nombre, url FROM usuarios, urls, url_relations WHERE usuarios.userId = url_relations.relatedUserId AND urls.urlId = url_relations.relatedUrlId

Quinto Nivel de F/N.

Existe otro nivel de normalización que se aplica a veces, pero es de hecho algo esotérico y en la mayoria de los casos no es necesario para obtener la mejor funcionalidad de nuestra estructura de datos o aplicación. Su principio sugiere:

1. La tabla original debe ser reconstruida desde las tablas resultantes en las cuales a sido troceada.

Los beneficios de aplicar esta regla aseguran que no has creado ninguna columna extraña en tus tablas y que la estructura de las tablas que has creado sea del tamaño justo que tiene que ser. Es una buena práctica aplicar este regla, pero a no ser que estes tratando con una extensa estructura de datos probablemente no la necesitarás.