UNIREMINGTON®
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

INGENIERÍA DE SOFTWARE II

INGENIERÍA DE SISTEMAS

FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA

Vicerrectoría de Educación a Distancia y virtual

2016

AUTOR

Piedad María Metaute Paniagua

Ingeniera de Sistemas, Especialista en Finanzas, Magister en Educación y Desarrollo Humano, diplomados en Competencias Pedagógicas, Diseño Curricular, seminarios sobre Formación en procesos y Técnicas de Investigación, sobre Negociación Electrónica, sobre Evaluación por Competencias pruebas ECAES, ICFES, SABER. Con amplia experiencia docente en los diferentes ciclos de educación desde Básica Secundaria, Técnica, Tecnológica, Profesional, Especialización en temáticas relacionadas con los sistemas, la computación y la informática, Asesora y Jurado de Proyecto de Grado, Coordinación de semilleros de investigación, Investigadora; así como experiencia en empresas del sector productivo en lo referente a la construcción de productos de software.

Piedad.metaute@uniremington.edu.co -pmetaute@gmail.com

Nota: el autor certificó (de manera verbal o escrita) No haber incurrido en fraude científico, plagio o vicios de autoría; en caso contrario eximió de toda responsabilidad a la Corporación Universitaria Remington, y se declaró como el único responsable.

RESPONSABLES

Jorge Mauricio Sepúlveda Castaño
Decano de la Facultad de Ciencias Básicas e Ingeniería
jsepulveda@uniremington.edu.co

Eduardo Alfredo Castillo Builes
Vicerrector modalidad distancia y virtual
ecastillo@uniremington.edu.co

Francisco Javier Álvarez Gómez
Coordinador CUR-Virtual
falvarez@uniremington.edu.co

GRUPO DE APOYO

Personal de la Unidad CUR-Virtual
EDICIÓN Y MONTAJE

Primera versión.
Segunda versión.
Tercera versión.
Cuarta versión 2016

TABLA DE CONTENIDO

Pág.

1 MAPA DE LA ASIGNATURA

INGENIERÍA DE SOFTWARE II

PROPÓSITO GENERAL DEL MÓDULO

La Ingeniería de Software II, se orienta hacia la identificación de procesos de Ingeniería de Software para la aplicación de diseños de software, administración de riesgos, gestión de configuración y medición, orientados a que las especificaciones de software se conviertan en productos funcionales de calidad, utilizando herramientas, métodos y las buenas prácticas que suministra el SWEBOK (Guía del cuerpo del conocimiento de la Ingeniería de Software). Teniendo en cuenta el proyecto que se viene trabajando desde la Ingeniería de Software I, se dará continuidad a dicho proyecto de software (documento y aplicación funcional), basado en ABP (Aprendizaje Basado en Problemas), desde el abordaje de una necesidad del contexto, con cliente real, bajo los principios de la Ingeniería de Software y los parámetros de la investigación formativa. Ver guía Ingeniería de Software II

OBJETIVO GENERAL

Identificar procesos de Ingeniería de Software para la aplicación de diseños de software, administración de riesgos, gestión de configuración y medición, orientados a que las especificaciones de software se conviertan en productos funcionales de calidad, utilizando herramientas, métodos y las buenas prácticas que suministra el SWEBOK (Guía del cuerpo del conocimiento de la Ingeniería de Software).

OBJETIVOS ESPECÍFICOS

- Definir arquitecturas de software, componentes e interfaces, para un sistema informático, utilizando herramientas propias de la Ingeniería de Software.
- Aplicar las mejores prácticas en el proceso de Gestión de Configuración de Software, analizando la viabilidad del cambio.
- Planificar procesos de administración de software en relación al análisis de riesgos y medición de procesos, con el fin de que se evalúe y prevea impactos negativos en el proceso de Ingeniería de Software.

UNIDAD 1

Diseño de Software

UNIDAD 2

Gestión de Configuración de Software

UNIDAD 3

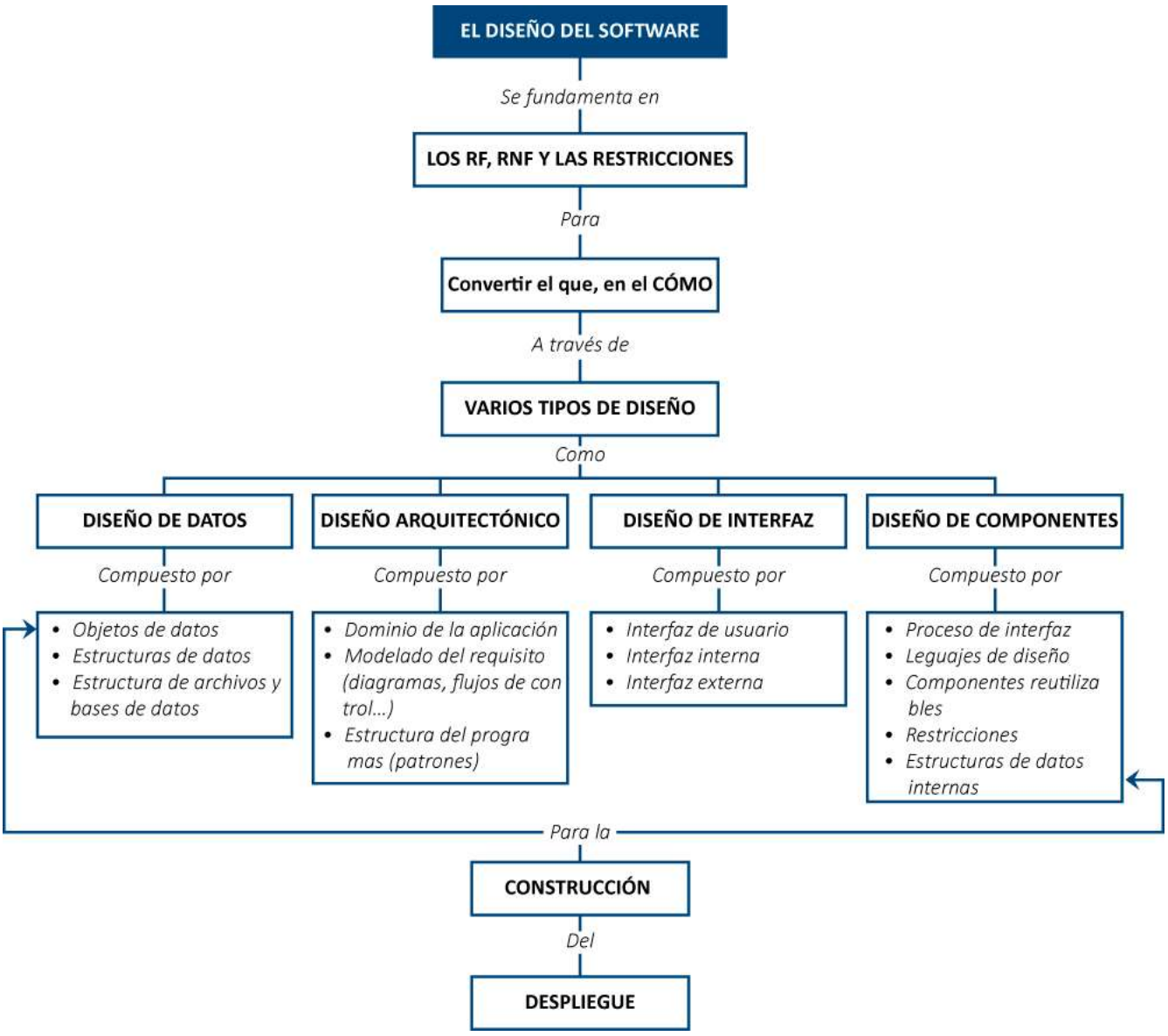
Administración de la Ingeniería de Software

UNIDAD 4

Proyecto de Ingeniería de Software (Aprendizaje Basado en Problemas e Investigación Formativa

2 UNIDAD 1: DISEÑO DE SOFTWARE

Mapa conceptual 1 (Fundamentos, estructura y arquitectura de diseño de software)



Fuente: Elaboración propia.

2.1 TEMA 1: FUNDAMENTOS, ESTRUCTURA Y ARQUITECTURA DE DISEÑO DE SOFTWARE.

2.1.1 SOBRE EL DISEÑO

Según **wordreference**, la palabra **diseño** significa:

> "Actividad creativa y técnica encaminada a idear objetos útiles y estéticos que puedan llegar a producirse en serie".

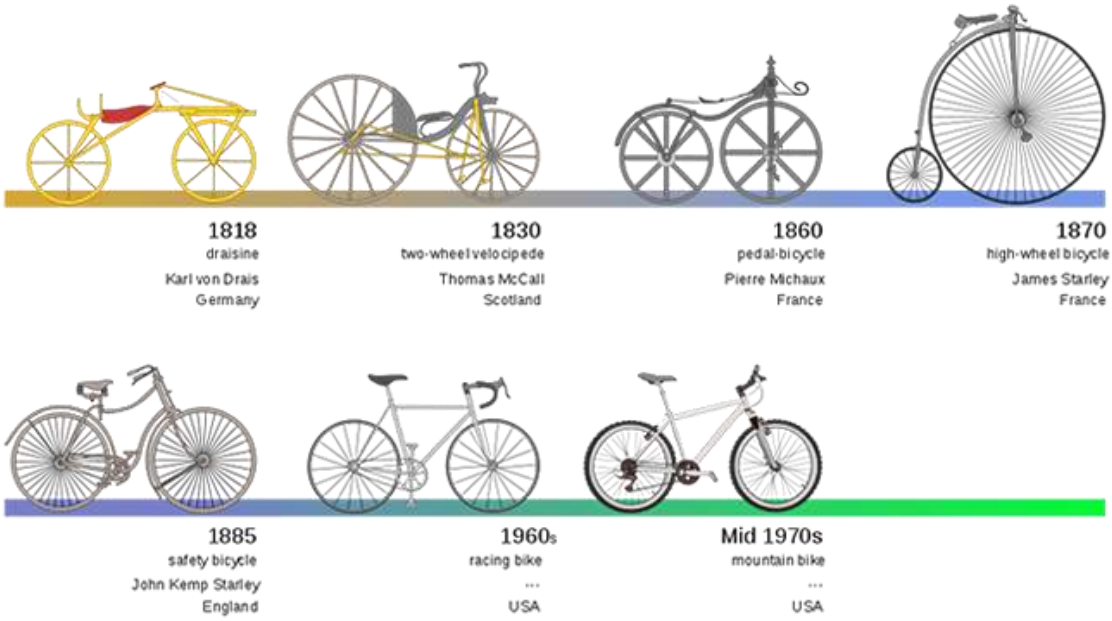
En el caso de la real academia española (2014), lo define como:

> **"Traza o delineación de un edificio o de una figura. Proyecto, plan. Diseño urbanístico. Concepción original de un objeto u obra destinados a la producción en serie. Diseño gráfico, de modas, industrial. Descripción o bosquejo verbal de algo. Disposición de manchas, colores o dibujos que caracterizan exteriormente a diversos animales y plantas".**

1Introducción a la Historia del Diseño Industrial Enlace

El **diseño evoluciona** con el tiempo, acorde a nuevos materiales, tendencias, estilos, formas, funcionalidad, entre otras características que muestran los **diferentes cambios** que se generan, acorde a necesidades, expectativas, en los negocios, lo cual motiva hacia la evolución de productos y servicios que buscan **satisfacer las diferentes necesidades sociales**. En la siguiente imagen, se puede observar la evolución que ha sufrido la bicicleta, en el tiempo, donde su puede observar diversos cambios entre ellas el diseño.

Imagen 1 Evolución en el diseño de la bicicleta



Fuente: http://tic.sepdf.gob.mx/micrositio/micrositio3/lineas.html

El diseño en el software a través de la historia, al igual que cualquier producto, **ha sufrido transformaciones**, en muchos aspectos, entre ellos **el diseño interno** y **externo**, pensado desde las mismas necesidades informacionales, donde cada vez se requiere de **software especializado** que además de funcional, se pueda adaptar a nuevas expectativas de clientes y usuarios. Una muestra de esa evolución, se puede ver claramente en **la empresa de desarrollo Microsoft**, la cual lleva en el mercado más de 40 años, suministrando gran variedad de productos diferenciados cada uno por sus **creativos diseños**, entre los que se pueden observar la línea del sistema operativo Windows. Ver la siguiente imagen.

Imagen 2 Evolución del sistema operativo Windows

Fuente: http://www.poderpda.com/wp-content/uploads/2014/08/Microsoft-Infografia-Evolucion-de-la-PC.jpg

De igual forma en empresas como **Apple**, **el diseño** ha sido una muestra de **creatividad** e **innovación** tanto en **hardware** como en **software**, lo que permite la captura de miles y miles de clientes, quienes a través de **la funcionalidad** de dichos productos, logran satisfacer sus **necesidades informacionales**, tanto en hardware como en software. Ver la siguiente imagen, donde se puede observar algunas proyecciones, que permiten visualizar la importancia que tiene el diseño para un usuario final.

Imagen 3 Evolución del IPhone

Fuente: http://www.ipadizate.es/2014/10/24/evolucion-iphone-apple-110799/

El diseño es definido en [IEEE610.12 - 90] como:

> "El proceso de definir la arquitectura, los componentes, interfaces y las otras características de un sistema o componente"

Es así como **todo producto de software** requiere de **diseños**, que consiste en **un conjunto de patrones** que se elaboran basados en **las especificaciones del software** y **las restricciones** que éste **debe cumplir**, donde se debe tener en cuenta además de **los requisitos funcionales**, los **no funcionales**, ya que el diseño dependerá del tipo de **necesidad informática** que se desee solucionar.

El diseño del software deberá permitir **integrar componentes** que lleven a cabo **tareas** con sus **interfaces**, facilitando **la comunicación** entre estos, es así que **la arquitectura de software** se relaciona con **el diseño** y **la implementación**, donde **su principal tarea** radica en obtener **la mayor funcionalidad** del sistema propuesto.

2.1.2 CONCEPTOS Y PRINCIPIOS DE DISEÑO DE SOFTWARE

El diseño, permite evolucionar **los requisitos del software**, relacionados con el **QUÉ debe funcionar en el sistema**, convirtiendo esos requisitos en modelos de software, con el fin de detallar el **CÓMO**, proporcionando detalles que tienen que ver con **la estructura de los datos**, la **arquitectura del software**, las **interfaces del sistema**, así como los **componentes** que se requieren para que la implementación del sistema cumpla con los **requisitos funcionales** y **no funcionales**.

En relación a la **arquitectura del software,** Pressman (2005), ésta tiene que ver con la **estructura jerárquica del programa,** la cual equivale a la forma como están **organizados** los módulos del software, además de la forma como éstos **interactúan**, así como **la estructura de datos** que van a utilizar sus componentes.

Para **el diseño arquitectónico**, se deberá tener en cuenta:

PROPIEDADES	CARACTERÍSTICAS
Estructurales	Hacen alusión a **la definición de los componentes** de un sistema por ejemplo **los módulos** y **la forma** como éstos **interactúan** con otros
Extra funcionales	Se refieren a **la capacidad** que debe tener **el diseño arquitectónico** en lograr **las características externas de calidad**, representado en **los requisitos no funcionales** como por ejemplo **el rendimiento**, **la fiabilidad**, **seguridad**, entre otras características que buscan **completitud** en el producto
Sistemas relacionados	En lo que respecta a **Sistemas Relacionados**, se debe tener cuidado que **el diseño arquitectónico** debe crearse pensando en **la capacidad** que éste debe tener de **reutilizarse.**

Fundamentos de diseño

Entre algunos fundamentos sobre el diseño, se tiene:

- **Abstracción:** se relaciona con **aislar un elemento** de **su contexto** o **del resto de los elementos** que **lo acompañan**, donde se puede observar **las características principales** de un objeto, de tal forma que **se diferencien** de otros objetos. Al relacionarlo con **el mundo real** la **abstracción** se puede **comparar,** como por ejemplo, cuando dentro de un grupo de personas, se observa un ser amado, sobre el cual **se centra toda la atención** siendo capaz de **describir algunas características** de **interés** que **le llame la atención** como por ejemplo el color del vestuario, la forma como lleva el cabello, entre otros aspectos que al observador le llame la atención, es así como si se le pregunta por otra persona que se encontraba al lado del ser amado, es posible que éste haya pasado desapercibido. Es así como **el diseño de software** debe permitir que el usuario pueda **abstraer objetos** y utilizarlos de **forma natural**.
- **Refinamiento:** El diseño de software debe **facilitar la evolución de las especificaciones**, de tal forma que permita observar claramente **los requisitos del software** de forma **más concreta** y a la vez permita **reconfigurar aspectos** que antes habían pasado desapercibidos y que con **los avances del diseño**, permite **incorporarlos** o fortalecer los existentes. En el caso del refinamiento, se puede relacionar con **una solicitud** para elaborar una tarjeta de presentación profesional, donde en el momento de presentar el diseño de dicha tarjeta, puede suceder que **existe información importante** que antes **no se había tenido en cuenta** y que sólo en **la etapa de diseño** se observa que es **necesario incorporarla**.
- **Modularidad:** El software deberá dividirse en **componentes** que se **identifiquen fácilmente**, a través de **un nombre** y una **ubicación específica**, denominados **módulos,** cuyo **conjunto de módulos** debidamente **integrada** componen **el sistema completo**, acorde a **los requisitos funcionales** que cubren **las necesidades informacionales** del cliente.

Un ejemplo que permite entender **la modularidad** es el caso de una empresa que se dedica a la producción de zapatos, dentro de ésta se encuentra **una estructura claramente definida**, existe **departamento de producción**, de **ventas**, **financiero**, **talento humano**, entre otros. Cada uno de ellos tiene funciones definidas, pero al **integrar los esfuerzos** de cada departamento, sale **una excelente producción** de zapatos de **alta calidad**, la cual es vendida a los clientes y **el dinero de retorno**, permite **fortalecer la empresa**.

2.2 TEMA 2. ARQUITECTURA DEL SOFTWARE

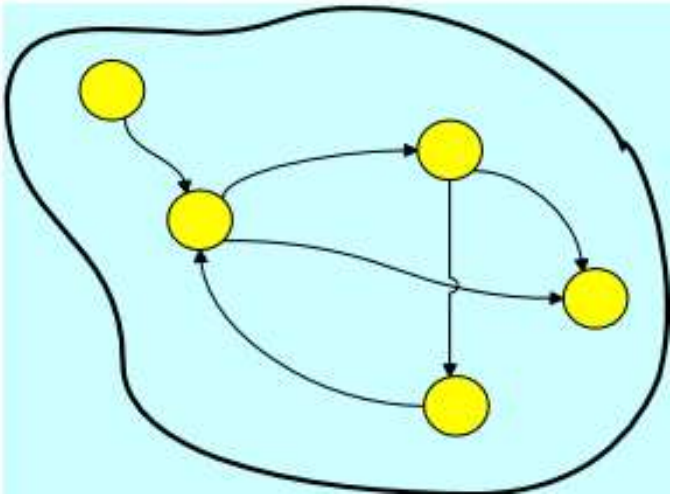
Define la forma como se relacionan los módulos o componentes del sistema. Según el estándar ISO 12207, la arquitectura del software se divide en dos tipos, como son Arquitectural y Detallado:

Arquitectural (de alto nivel), éste describe la estructura y organización de los subsistemas o componentes y sus relaciones. Es el primer paso que se debe hacer a nivel de diseño, antes de abordar el diseño detallado. Se basa en la especificación del requisito, aunque se puede desarrollar en paralelo con los requisitos. **Un buen diseño requiere de creatividad**, permitiendo que las actividades de diseño puedan ser modificadas a medida que avanza el desarrollo, en caso de que sea necesario realizar algún cambio.

Detallado: es el diseño que describe cada componente o subsistema y su comportamiento específico, de forma que pueda procederse a su construcción.

En la siguiente imagen se puede observar una arquitectura compuesta por varios nodos, donde los unos se encuentran relacionados con los otros, cada nodo representa un módulo del sistema con funciones específicas. Dicho comportamiento se refleja en el diseño arquitectural del software, donde cada componente tiene una función específica de acuerdo a los requisitos funcionales del sistema.

Imagen 4 Ejemplo de Arquitectura de un sistema



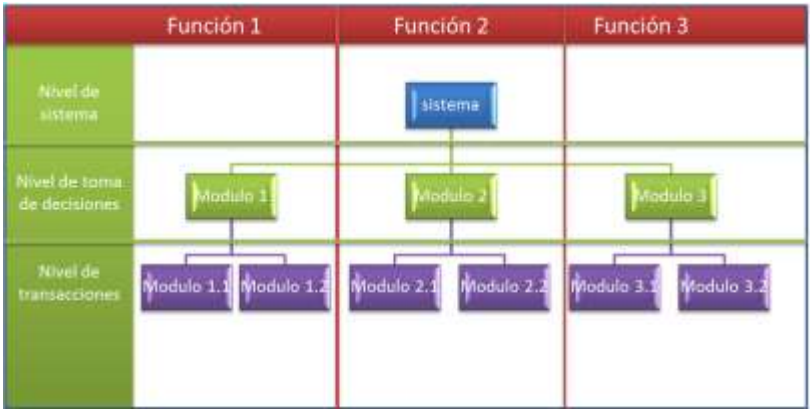
Fuente: Francisco Ruiz

2.2.1 DIVISIÓN ESTRUCTURAL DEL SOFTWARE

Corresponde a la forma en que se organizan los componentes del software, donde se puede observar desde una vista horizontal y una vertical. De forma vertical se observa las diferentes funciones que posee el software, de acuerdo a los requisitos funcionales y de forma horizontal la responsabilidad del software en cuanto al nivel de profundidad, donde en el nivel del sistema se puede observar el control que se ejerce desde la motivación del

usuario, en el nivel de toma de decisiones se puede visualizar las condiciones del sistema para ejercer su funcionalidad y en el último nivel la ejecución propiamente dicha de acuerdo a las solicitudes del usuario y funcionalidad del sistema. Ver imagen.

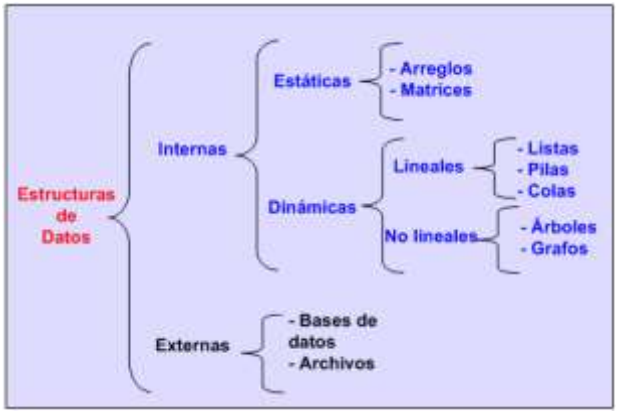
Imagen 5 Organización de los componentes del software



Fuente: http://sistemasumma.com/2011/03/15/conceptos-del-diseno/

- **Estructura de datos:** ésta estructura permite organizar la forma como se relacionan los datos, para su procesamiento y fluctuación a través de los diferentes componentes. Los datos normalmente se representan en estructuras como pilas, colas, árboles, listas, además de las bases de datos que pertenecen a otra estructura de almacenamiento, muy utilizada en los diferentes sistemas informáticos. Ver imagen.

Imagen 6 Estructura de los datos en los sistemas informáticos



Fuente: http://virtual.unach.edu.ec/pluginfile.php/72/course/summary/es1.png

2.2.2 ALGUNOS TIPOS DE ARQUITECTURAS EN EL DISEÑO DE SOFTWARE

De acuerdo a los diferentes estudios realizados sobre diseño de software, se encontró varios tipos de arquitecturas, cada una de ellas proporciona aportes a la temática de diseño, es así como se tomó tres de las arquitecturas más relevantes como son: Arquitectura de descomposición modular, arquitectura Cliente-servidor, arquitectura de tres niveles.

Arquitectura descomposición modular o modularización: consiste en descomponer un conjunto de elementos cumpliendo condiciones de bajo **acoplamiento** (donde los componentes deben ser independientes de los demás componentes) y alta **cohesión** (donde los componentes deben tener significado propio), por lo tanto se busca subdividir el sistema en varios subsistemas con tareas completas y con codificación independiente, de manera que los módulos puedan ser codificados, comprobados y depurados independientemente, pero que de igual forma puedan combinarse con otros módulos recibiendo de ellos información y generando información para ellos.

Arquitectura cliente servidor: Es un modelo de procesos independientes, donde se intercambia información, servicios y recursos, donde el **cliente** es quien solicita los recursos y el **servidor** es quien responde a las solicitudes, por lo tanto el servidor contiene todos los recursos a compartir con los diferentes clientes y el cliente posee los recursos particulares como por ejemplo, su interfaz de usuario, captura y validación de datos de entrada, generación de informes y consultas de acuerdo a su perfil. En el caso del servidor, éste tiene la función de compartir periféricos, bases de datos compartidas, administración de recursos, tiempos, concurrencia en los diferentes accesos, entre otros servicios.

Arquitectura de niveles o por capas: Ésta arquitectura, está compuesta por la **capa de la interfaz,** donde el usuario interactúa directamente y sirve de entrada de datos al sistema, **la capa de las reglas del negocio**, la cual contiene la funcionalidad del sistema y por último la **capa de los datos**, quien se encarga de la gestión de los almacenamientos de datos. Siendo la arquitectura más utilizada en el diseño del software. Ver imagen.

CAPA DE INTERFAZ	CAPA DE REGLAS DEL NEGOCIO	CAPA DE DATOS
Vistas de usuario, formularios	Componentes modulos, programación o codificación	Almacenamientos a través de las bases de datos

Fuente: elaboración propia.

La arquitectura en mención presenta beneficios como independencia entre capas, facilidad de sustitución con implementaciones alternativas de los mismos servicios básicos, minimización de dependencias entre capas, reutilización de capas por otros servicios de mayor nivel. A continuación se hace referencia a cada una de las tres capas:

Capa de interfaz o presentación: Permite la interacción entre el usuario y el software, a través de menú, vistas, pantallazos, donde su función principal consiste en facilitar el ingreso de datos, realizar algunas validaciones, automatizar órdenes a nivel de ejecución de procesos, generar información a través de informes y consultas, ofreciendo facilidad en la usabilidad del recurso.

Capa de reglas de negocio o diseño de componentes: Contiene todo el desarrollo lógico encargado del procesamiento de los datos con el fin de suministrar la funcionalidad al sistema, siendo responsable de los cálculos, procesos con los datos que recibe, validaciones, controlando así la ejecución de la capa de acceso a datos y servicios externos, coordinando el uso de los diferentes objetos y componentes lógicos, su estructuración y dinámica, acorde a las características de programación, sirviendo de enlace directo entre la interfaz y los almacenamientos de datos.

Capa de Datos: permite la comunicación con la capa de las reglas del negocio u otras aplicaciones que tienen la necesidad de acceder a los datos. Dicha capa normalmente se representa por bases de datos estructuradas para el almacenamiento permanente de datos y la generación de información.

El diseño de datos transforma los requisitos, creado durante el análisis y el modelamiento, en las estructuras de datos que se van a requerir para implementar el software. Las estructuras de datos clásicas son los elementos escalares, los arrays, las listas y los árboles. Según Wasserman (1996), “La actividad principal durante la fase de

diseño de datos es la selección de las representaciones lógicas de las estructuras de datos, identificados durante las fases de definición y especificación de requerimientos. Una actividad importante durante el diseño es la de identificar los módulos de programa que deben operar directamente sobre las estructuras de datos. De esta forma, puede restringirse el ámbito del efecto de las decisiones concretas de diseño de datos."

Según Pressman (2010) el diseño de datos transforma el modelo del dominio de información creado en el análisis en las estructuras de datos necesarias para la implementación del software. De igual forma la Influencia de la estructura de datos repercute fuertemente en la estructura del programa y en la complejidad de los procedimientos, donde la estructura de los datos bien diseñados conduce a mejorar, la modularidad efectiva, reducción de la complejidad procedimental, entre otros aspectos que contribuyen a la calidad de los requisitos del sistema. Ver otra imagen, sobre la arquitectura por capas.

Imagen 7 Arquitectura por capas desde el usuario

http://arevalomaria.files.wordpress.com/2010/12/ejemplo_mvc1.png?w=460&h=523

2.3 TEMA 3: ESTRATEGIAS, MÉTODOS Y MODELADO PARA EL DISEÑO DE SOFTWARE

Dentro del proceso de Ingeniería de Software, es importante el uso de estrategias, métodos y modelos adecuados, con el fin de conseguir diseños que se adapten a las diferentes necesidades informacionales, es así como durante el diseño se desarrollan, revisan y evalúan el diseño de datos, el diseño arquitectónico, el diseño procedimental, el diseño de la interfaz, diseño de componentes, entre otros, que buscan traducir con precisión

los requisitos del cliente en un producto que pueda ser probado, evolucionado e implementado en el ambiente del usuario final.

Uno de los modelos más importantes para la estructuración de base de datos, es el modelo relacional, del cual se realiza una inducción, ya que para el ejemplo que se trae, se necesita de dicha información.

2.3.1 CAPA DE DATOS BAJO EL MODELO RELACIONAL

Como soporte importante en el diseño de los datos, se cuenta con el modelo relacional, el que se fundamenta en la lógica de predicados y la teoría de conjuntos, propuesto por Edgar Frank Codd, a inicios de los años 70, pero aún hoy totalmente vigente, donde éste se basa en relaciones entre tuplas llamadas también (líneas o registros) y las columnas (campos), cuyo conjunto corresponde a una tabla compuesta por líneas y columnas. Así:

Imagen 8 Tabla (Registros y campos)

	Columnas o campos		
	Identificación	**Nombre**	**Edad**
Tuplas o líneas	100	Juan Camilo	15
	200	Isabella	7
	300	Carlos Mario	25

Fuente: elaboración propia.

El modelo relacional se ha convertido en una herramienta indispensable para la construcción de estructuras de bases de datos, sin importar el SGBD (Sistema de Gestión de Base de Datos) a utilizar, entre los más utilizados se encuentran SQL Server, My SQL, Oracle, Informix, ProgreSQL, los cuales toman dicho modelo para construir sus almacenamientos.

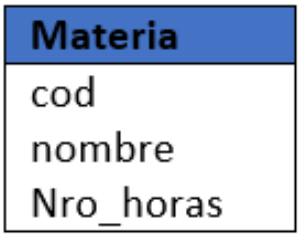
- Elementos del modelo relacional

Entidad: Corresponde a un objeto concreto o abstracto que posee características propias, el cual se representa con un rectángulo y se identifica por un nombre en singular. Cuando se habla de un objeto concreto se está haciendo alusión a todo aquello que tiene representación física como por ejemplo la entidad estudiante, profesor, empleado, automóvil, casa, libro, entre otras y cuando se refiere a un objeto abstracto, éste corresponde a los que no tienen representación física como por ejemplo materia, asesoría, vacaciones, préstamo, venta, pago, entre otros. Ver ejemplo de entidad concreta y entidad abstracta.

Imagen 9 Entidad estudiante (concreta)

Fuente: elaboración propia.

Imagen 10 Entidad abstracta (materia)



Fuente: elaboración propia.

Atributo: se refiere a aquellas características que poseen las entidades, donde por ejemplo en la entidad materia, equivale a cod, nombre, nro_horas.

Clave primaria (primary key): corresponde a uno de los atributos de la entidad, cuyo valor es único e identifica plenamente a los demás atributos de la misma entidad. La clave primaria se identifica con el símbolo número (#) y en la mayoría de las ocasiones corresponde al código de la entidad, en el caso de la entidad estudiante, se puede observar que la clave primaria es el atributo (ident). Ver ejemplo de entidad con clave primaria.

Imagen 11 Clave primaria (Primary Key)

Fuente: elaboración propia.

Al crear la estructura de la base de datos, la entidad, se convierte en una tabla que va a permitir almacenar datos. En la siguiente tabla se observa que en la clave primaria (ident), no se repiten los datos, ya que por ejemplo Juan Camilo y Juan Carlos deben tener una identificación diferente, ya que se trata de las personas diferentes, es así como la clave primaria permite que se pueda diferenciar la información que corresponde a cada uno de los estudiantes.

Imagen 12 Ejemplo de clave primaria (Primary Key)

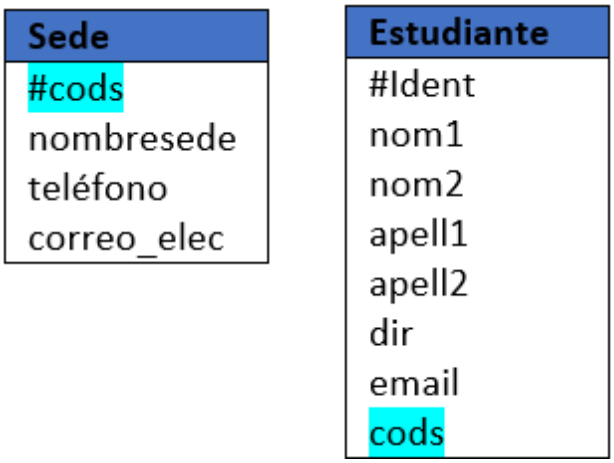
Estudiante						
Ident	**Nom1**	**Nom2**	**Apell1**	**Apell2**	**dir**	**Email**
001	Juan	Camilo	Ríos	Carmona	Cra 65 23-50	jrios@gmail.com
002	Sara	María	Cano	Yepes	Calle 34 54-21	smcano@gmail.com
003	Sara	María	Osorio	Restrepo	Cra 76 32-11	sosorio@gmail.com
004	Juan	Carlos	Ríos	Oquendo	Calle 66 34-22	jcrioso@gmail.com

Fuente: elaboración propia.

Clave foránea o extranjera (foreign key): corresponde a una clave primaria en otra entidad y permite hacer referencia a dos entidades, a través de sus relaciones, observando las entidades Sede y Estudiante, aparece el atributo (cods) como clave primaria marcado con el símbolo de número en la entidad (Sede), pero como clave foránea (cods) en la entidad (Estudiante), lo que indica que una entidad hace referencia a otra. Es importante aclarar que la clave foránea no es única y que al evolucionar el modelo relacional a tablas dentro

de la base de datos, la clave foránea se puede repetir, lo que no ocurre con la clave primaria. Ver clave foránea en la entidad estudiante.

Imagen 13 Clave Foránea (Foreign Key)



Fuente: elaboración propia.

En las siguientes tablas se puede ver claramente la diferencia entre clave primaria y foránea, donde en la tabla sede el código no se repite (clave primaria), y en la tabla estudiante el código de sede se puede repetir (clave foránea), mostrando que existen tres estudiantes que pertenecen a la sede Montería, uno a la sede de Cúcuta y dos a la sede de Villavicencio. Por lo tanto la clave primaria y la foránea, se utilizan para generar relación entre dos entidades. Ver diferencia entre clave primaria y foránea en las siguientes tablas.

Imagen 14 Ejemplo sobre clave primaria y clave foránea

Sede			
cods	**nombresede**	**teléfono**	**Correo_elec**
01	Montería	8902345	Monteria.cur@uniremington.edu.co
02	Cúcuta	4567896	Cucuta.cur@uniremington.edu.co
03	Villavicencio	9765432	Villavicencio.cur@uniremington.edu.co

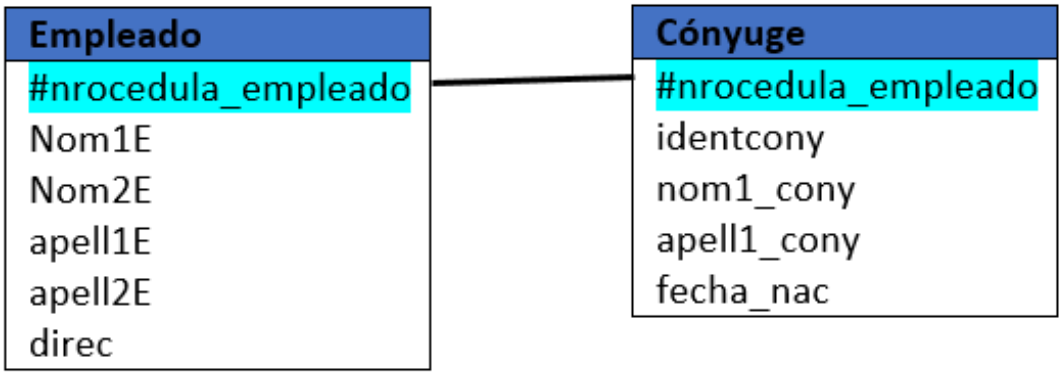
Estudiante							
Ident	**Nom1**	**Nom2**	**Apell1**	**Apell2**	**dir**	**Email**	**cods**
001	Juan	Camilo	Ríos	Carmona	Cra 65 23-50	jrios@gmail.com	01
002	Sara	María	Cano	Yepes	Calle 34 54-21	smcano@gmail.com	02
003	Sara	María	Osorio	Restrepo	Cra 76 32-11	sosorio@gmail.com	01
004	Juan	Carlos	Ríos	Rúa	Calle 66 34-22	jcrioso@gmail.com	03
005	Sofía		Rivera	Cano	Cra 56 43-67	srivera@gmail.com	03
006	Carlos		Mena	Manco	Calle 67 32-1	cmena@hotmail.com	01

Fuente: elaboración propia.

Relaciones: una relación es una asociación entre dos entidades, con el fin de referenciar una con otra. Existen tres tipos de relaciones como son de uno a uno (1:1), de uno a muchos (1:N) y de muchos a muchos (N:M)

- ✓ **Relación de uno a uno (1:1)**, ésta se da entre dos entidades, a través de sus claves primarias, lo que indica que en ninguna relación se pueden repetir registros. Éste tipo de relación es la que menos se da, sin embargo existen casos en los cuales se debe aplicar. Ver el siguiente ejemplo.

Imagen 15 Relación de Uno a Uno (1:1)



Fuente: elaboración propia.

Es así como un empleado, sólo puede tener registrado en su trabajo un cónyuge, y en esa empresa sólo debe aparecer un cónyuge, para cada empleado(a). En las siguientes tablas se muestra que el empleado Felipe, tiene como cónyuge a Rocío, el empleado Camilo a Claudia y la empleada María Elena no tiene cónyuge registrado.

Imagen 16 Ejemplo de Relación de Uno a Uno (1:1)

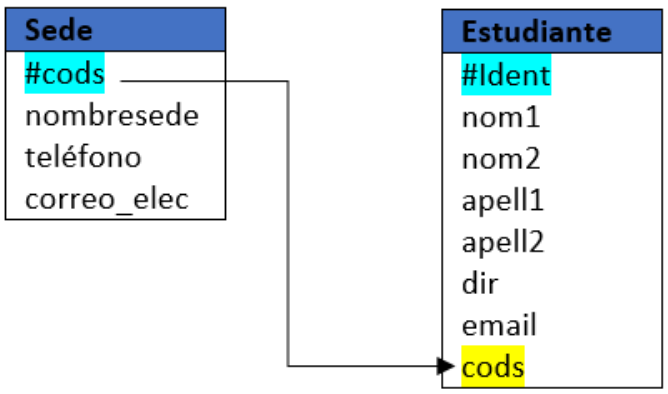
Empleado					
nrocedula	Nom1	Nom2	Apell1	Apell2	direc
1111	Felipe	Alonso	Rosales	Caro	Cll 34 56-78
2222	María	Elena	Mejía		Cra 10 45-66
3333	Camilo		Orrego		Cra 56 32-12

Cónyuge				
Nrocedula_empleado	identcony	Nom1_cony	Apelli1_cony	Fecha_nac
1111	100	Rocío	Vanegas	20-11-69
3333	456	Claudia	Rivera	24-08-75

Fuente: elaboración propia.

- ✓ **Relación uno a muchos (1:N):** Ésta relación es la más frecuente, donde siempre el lado uno de la relación corresponde a la clave primaria y el lado muchos corresponde a la clave foránea y se lee (a una sede pueden pertenecer varios estudiante y un estudiante sólo puede pertenecer a una sede). Ver la siguiente imagen.

Imagen 17 Relación de uno a muchos (1:N)



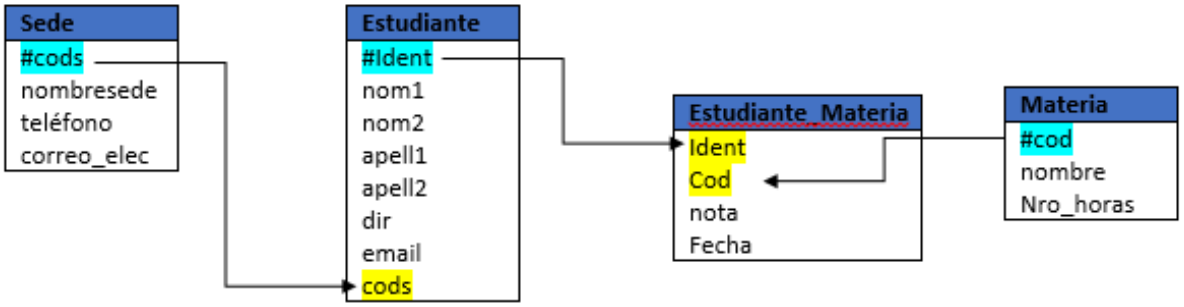
Fuente: elaboración propia.

- ✓ **Relación de muchos a muchos N:M:** cuando aparece éste tipo de relación, según el modelo relacional planteado, ésta se soluciona, con una nueva relación que sirva como entidad intermedia entre ambas. Ésta entidad intermedia llevará un nombre que haga alusión a las dos entidades que generan dicha

relación y contendrá como mínimo las claves primarias de las dos entidades, las cuales realmente serán en la entidad intermedia las claves foráneas que permitirán la relación entre las entidades de referencia, que en éste caso se refiere a (estudiante y materia), ya que un estudiante puede tomar una o varias materias y una misma materia puede ser vista por uno o varios estudiantes.

Por lo tanto, la relación de muchos a muchos, según el modelo relacional, no se puede manejar directamente, por lo que se crea la entidad (estudiante materia), la cual posee las claves primarias de la entidad estudiante y de la entidad materia. Cuando las claves son ubicadas dentro de la entidad intermedia, toman el nombre de claves foráneas o claves extranjeras. Observar el siguiente modelo relacional.

Imagen 18 Modelo Relacional



Fuente: elaboración propia.

Al llevar dicho modelo a una estructura de base de datos, apoyado en un SGBD (Sistema de Gestión de Base de Datos), tomará la forma que se muestra en las siguientes tablas, donde un estudiante puede tomar varias materias y una misma materia puede ser tomada por varios estudiantes y al mismo tiempo se le puede asignar otros datos como la nota que cada estudiante se sacó, así como la fecha en la cual se sacó la nota. Ver imagen, donde los campos que se encuentran sombreados de azul, corresponden a las claves primarias o principales y las que están sombreadas de amarillo corresponden a las claves foráneas, secundarias o extranjeras.

Imagen 19 Ejemplo de base de datos evolucionada hacia tablas

Sede			
cods	nombresede	teléfono	Correo_elec
01	Montería	8902345	Monteria.cur@uniremington.edu.co
02	Cúcuta	4567896	Cucuta.cur@uniremington.edu.co
03	Villavicencio	9765432	Villavicencio.cur@uniremington.edu.co

Estudiante							
Ident	Nom1	Nom2	Apell1	Apell2	dir	Email	cods
001	Juan	Camilo	Ríos	Carmona	Cra 65 23-50	jrios@gmail.com	01
002	Sara	María	Cano	Yepes	Calle 34 54-21	smcano@gmail.com	02
003	Sara	María	Osorio	Restrepo	Cra 76 32-11	sosorio@gmail.com	01
004	Juan	Carlos	Ríos	Rúa	Calle 66 34-22	jcrioso@gmail.com	03
005	Sofía		Rivera	Cano	Cra 56 43-67	srivera@gmail.com	03
006	Carlos		Mena	Manco	Calle 67 32-1	cmena@hotmail.com	01

Estudiante_materia			
ident	cod	nota	Fecha
001	10	3,5	19-01-16
002	10	4,2	20-01-16
003	20	5,0	30-01-16
001	30	3,8	28-01-16
004	20	2,8	21-01-16
006	20	4,2	22-01-16

Materia		
cod	nombre	Nro_horas
10	Ingeniería de Software	64
20	Cálculo I	64
30	Teoría de Sistemas	48
40	Electrónica	64

Fuente: elaboración propia.

Normalización del modelo relacional

Existen una serie de normas, llamadas formas normales que se diseñaron con el fin de obtener un conjunto de relaciones adecuadas, de tal forma que cumplan con ciertas características que permitan que la información posea ingredientes de calidad, desde la estandarización del modelo relacional, buscando que:

- La base de datos contenga el mínimo de atributos necesarios para soportar las necesidades informacionales del sistema.
- El acceso a los datos de la base de datos sea más sencillo.
- Evitar la redundancia de datos en la base de datos.
- Permitir actualización de los datos, utilizando el mínimo número de operaciones posibles.
- Evitar inconsistencias en los datos.
- Reducción de espacios de almacenamiento
- Reducción de costos de operación, entre otras.

Formas normales

En el siguiente video, se explican las primeras tres formas normales para estandarizar el modelo relacional

2Base de datos #13 | Normalización (1FN, 2FN y 3FN) Enlace

- **1NF (Primera Forma Normal)** Una tabla no puede tener múltiples valores en cada columna. Los datos son atómicos. Ejemplo, no debe existir un campo que se llame nombres, éste se debe dividir en dos campos, donde el uno almacenaría la información del primer nombre y el otro del segundo nombre. Esto facilita el manejo de la información dentro de las bases de datos.

- 2**NF (Segunda Forma Normal)** Dependencia Funcional. Una relación está en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir que no existan dependencias parciales. (Todos los atributos que no son clave principal deben depender únicamente de la clave principal), como por ejemplo la cédula de un cliente debe identificar los demás datos del cliente como su nombre, su apellido, su dirección, pero si le coloco, el nombre de un vendedor, éste no se identifica plenamente con la cédula del cliente.

- **3NF (Tercera Forma Normal)** La tabla se encuentra en 3FN si está en 2FN y si no existe ninguna dependencia funcional transitiva entre los atributos que no son clave. Un ejemplo que ilustra dicha forma puede ser la entidad estudiante, el cual posee los siguientes atributos (carnet, nom1, nom2, apell1, apell2, direc, tel, acudiente, fecha_nac). En el caso anterior, se viola la tercera forma normal, ya que acudiente

no se identifica plenamente con el carnet que es la clave primaria de la entidad estudiante, por lo tanto acudiente forma una entidad propia con los atributos de dichos acudientes como por ejemplo Acudiente (cedula, nom1, nom2, apell1, apell2, direc, tel), solucionando así la transitividad en la información.

Para abordar más a fondo las formas normales, éstas se tratarán en la asignatura Base de Datos.

- **Diccionarios de Datos:** herramienta que consiste en un conjunto de metadatos (datos altamente estructurados que describen información, contenido, calidad, condición, entre otras características, se refiere a información sobre información o datos sobre los datos) compuesta por estructuras lógicas y puntuales de los datos, entre algunas características tenemos: nombre, descripción, tipos de datos, alias, contenido, organización, entre otros aspectos. El diccionario de datos describe puntualmente la conformación de la estructura de la base de datos, herramienta que le facilita al diseñador de la base de datos su construcción. Ver el siguiente ejemplo:

Imagen 20 Ejemplo de diccionario de base de datos

Diccionario de datos Sede					
Nombre del campo	**Tipo dato**	**Tipo de campo**	**Requerido**	**Relación**	**Ejemplo**
Cods	Text(6)	Primary Key	Si	Con tabla estudiante	01
Nombresede	Text(30)	Normal	Si		Montería
Telefono	Text(12)	Normal	Si		8902345
Correo_elec	Text(30)	Normal	No		Monteria.cur@uniremington.edu.co

Diccionario de datos Estudiante					
Nombre del campo	**Tipo dato**	**Tipo de campo**	**Requerido**	**Relación**	**Ejemplo**
Ident	Text(10)	Primary Key	Si	Con tabla estudiante_materia	1001009021
Nom1	Text (15)	Normal	Si		Juan
Nom2	Text (15)	Normal	No		Camilo
Apell1	Text(15)	Normal	Si		Ríos
Apell2	Text(15)	Normal	No		Cano
dir	Text(30)	Normal	Si		Carrera 35 67-43
email	Text(30)	Normal	No		jcrios@gmail.com
cods	Text(6)	Foreign key	Si	Con tabla sede	01

Fuente: elaboración propia.

Es así que para efectos pedagógicos, se tomará la arquitectura por capas (Datos, reglas del negocio o diseño de componentes, interfaz de usuario o presentación), la que se aplicará al sistema de Proyecto de Grado. Por lo

tanto se evolucionarán las especificaciones del software abordadas en el módulo de Ingeniería de Software I al diseño de software. Recordemos la situación a abordar, así como el diagrama de casos de uso principal:

2.3.2 SITUACIÓN PROBLEMÁTICA, COMO EJEMPLO PRÁCTICO DEL DISEÑO DE SOFTWARE

En la facultad de Ciencias Básicas e Ingeniería de la Corporación Universitaria Remington, se tiene dificultad con la gestión de los Proyecto de Grado, ya que se presenta inconsistencias en la información, existen reproceso, pérdida de información, ineficiencia a la hora de dar respuestas sobre el estado de los proyectos, pérdida de tiempo. La forma como se viene manejando es a través de la hoja electrónica Excel complementado con planillas manuales, es importante tener presente que los Proyecto de Grado, pueden ser tipo Monografías, Investigación, Software entre otros, el equipo para desarrollar el trabajo puede ser máximo de tres estudiantes, cuyo equipo puede estar conformado por estudiantes de diferentes carreras. Además, se maneja la información de los Asesores de dichos proyectos, así como el tema o los temas sobre los cuales trata el proyecto (Facturación, El Negocio Virtual, Negocios del siglo XXII, Literatura Colombiana, Ecología, entre otros), se debe tener en cuenta que cada estudiante solo debe de figurar en un proyecto de grado y estar matriculado en una sola carrera. Debe registrarse la fecha, en la cual se le dio o dará asesoría al proyecto. En cualquier momento se requieren informes y consultas para el decano, asesores, estudiantes, secretaria de la facultad y no siempre se tiene la información a la mano, generándose inconformidad en muchas ocasiones.

Entre algunas de las solicitudes que se requieren en relación al Proyecto de Grado son:

- Nombre de los proyectos con sus Integrantes.
- Proyecto con sus respectivos asesores, y sus fechas de asesoría, tanto para los aprobados, como para los que se encuentran en proceso.
- Consulta por parte de los estudiantes para visualizar los proyectos por tema, por tipo de proyecto, para que sirva como referencia a proyectos posteriores.
- Consulta sobre las fechas de asesorías de los proyectos
- Consultar sobre proyectos aprobados, antes de realizar el proceso de graduación, entre otras.

Fuente: elaboración propia.

Caso de uso Principal o General

Se retoma diagrama principal de la situación “Proyecto de Grado”, en el que se muestran los requisitos funcionales del software a nivel general. Ver diagrama.

Imagen 21 Caso de uso principal (Proyecto de Grado)

Fuente: elaboración propia.

2.3.3 APLICACIÓN DE ARQUITECTURA POR CAPAS PARA EL DISEÑO DEL SOFTWARE (CASO DE ESTUDIO, PROYECTO DE GRADO)

Retomando el ejemplo propuesto sobre el manejo de Proyecto de Grado de los estudiantes que pertenecen a la Facultad de Ciencias Básicas e Ingeniería y tomando como base los requerimientos del cliente, evolucionado hacia el análisis de los requisitos del sistema, teniendo en cuenta las restricciones como, por ejemplo:

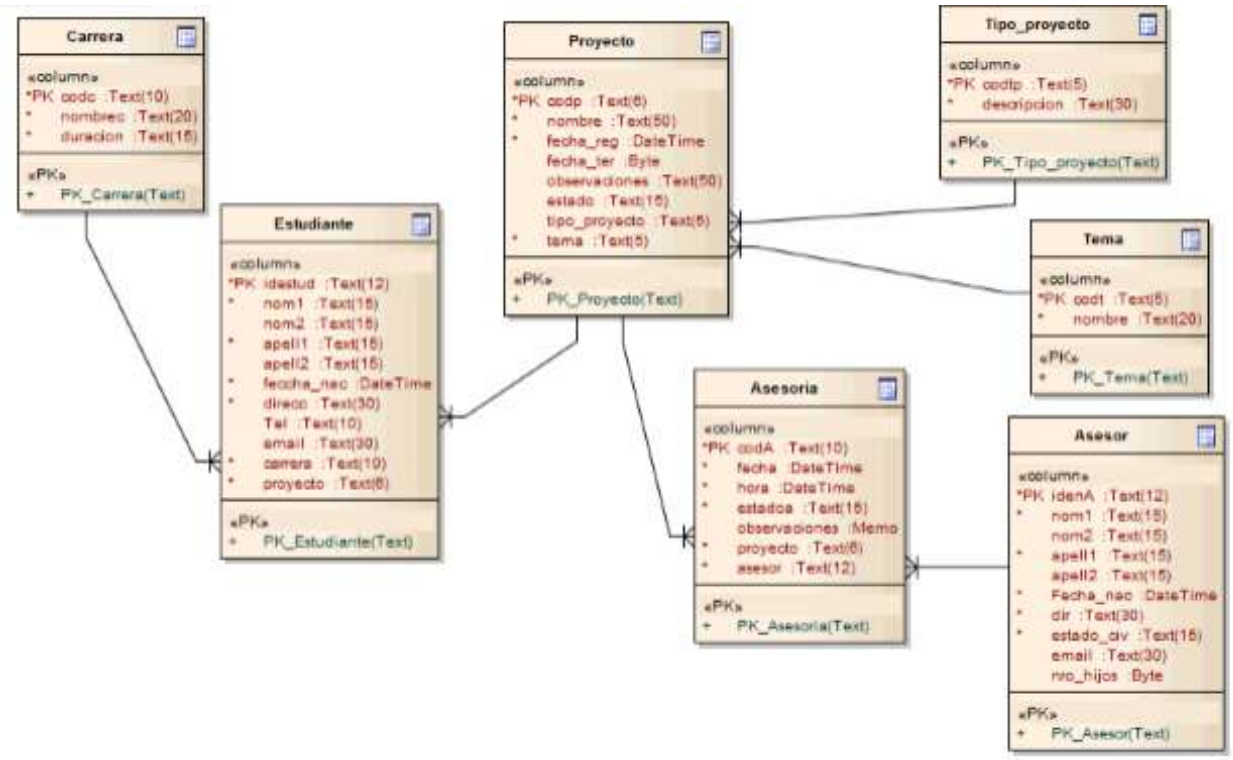
- -Un estudiante sólo puede pertenecer a un proyecto de grado.
- -Un estudiante sólo puede estar cursando una carrera.
- -Para un proyecto se debe definir un solo tema.
- -El proyecto sólo puede pertenecer a un solo tipo de proyecto como por ejemplo si es un proyecto de investigación, no puede ser un proyecto de desarrollo de software.

Para la especificación de los requisitos del sistema, se tomaron varias herramientas entre ellas cuadros, listados, diagramas, escenarios, entre otros, entendiéndose que no son las únicas herramientas que se pueden utilizar y que cada empresa, organiza sus líneas de trabajo, acorde a sus necesidades y expectativas.

2.3.4 CAPA DE DATOS (SITUACIÓN PROBLEMÁTICA, PROYECTO DE GRADO)

- Diseño de la base de datos resultado de dicho proceso: Para el diseño de la base de datos se tuvo en cuenta los parámetros dados por el modelo relacional, así como sus formas normales, de igual forma se utilizó software modelador Entreprise Architect, generándose de forma automática los diccionarios de la base de datos, cuyos datos depende de la forma como fue programado dicho modelador (los diccionarios normalmente son generados por los modeladores y los datos que éstos arrojen, depende del tipo de software). Ver modelo relacional (Proyecto de Grado) y diccionario de datos.

Imagen 22 Modelo Relacional (Proyecto de Grado)



Fuente: elaboración propia.

Realizando lectura al modelo de datos, se puede observar lo siguiente:

- A una carrera pueden pertenecer varios estudiantes y un estudiante, sólo puede elegir una carrera.
- A un tipo de proyecto pueden pertenecer varios Proyecto de Grado, pero a un proyecto de grado sólo puede asignársele un tipo de proyecto.
- Un tema puede figurar en varios proyectos, pero a un proyecto sólo se le puede asignar un solo tema.
- Un estudiante puede estar en un solo proyecto, pero en un proyecto puede existir uno o varios estudiantes (por código se valida que no pasen de tres estudiantes para un solo proyecto).

- Un asesor, puede asesorar varios proyectos y un proyecto puede tener varios asesores (en éste caso se crea una relación de muchos a muchos, la cual se soluciona con una entidad intermedia que en éste caso se llama asesoría, ya que, entre proyecto y asesor, no existe una relación directa, sólo se presenta con la entidad intermedia). Ésta se lee: Un asesor puede dar varias asesorías a varios proyectos y un proyecto puede recibir varias asesorías, que pueden darse por varios asesores cada una de ellas.

Lo anterior permite hacer una validación del modelo de datos, en relación a las especificaciones del requisito de software.

A continuación, se puede observar los diccionarios de datos generados de forma automática por Enterprise Architect.

Imagen 23 Diccionario de Datos (Proyecto de Grado)

Asesor

Database: MSAccess 2007, *Stereotype:* «table», *Package:* Data Model
Detail: *Created on* 20/01/2016. *Last modified on* 20/01/2016.
Notes:

Columns

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
True	idenA	Text	True	False	12				
False	nom1	Text	True	False	15				
False	nom2	Text	False	False	15				
False	apell1	Text	True	False	15				
False	apell2	Text	False	False	15				
False	Fecha_nac	DateTime	True	False					
False	dir	Text	True	False	30				
False	estado_civ	Text	True	False	15				
False	email	Text	False	False	30				
False	nro_hijos	Byte	False	False					

Fuente: elaboración propia.

Asesoria

Database: MSAccess 2007, *Stereotype:* «table», *Package:* Data Model
Detail: *Created on* 20/01/2016. *Last modified on* 20/01/2016.
Notes:

Columns

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
True	codA	Text	True	False	10				
False	fecha	DateTime	True	False					
False	hora	DateTime	True	False					
False	estadoa	Text	True	False	15				
False	observaciones	Memo	False	False					
False	proyecto	Text	True	False	6				
False	asesor	Text	True	False	12				

Fuente: elaboración propia.

Carrera

Database: MSAccess 2007, *Stereotype:* «table», *Package:* Data Model
Detail: *Created on* 20/01/2016. *Last modified on* 20/01/2016.
Notes:

Columns

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
True	codc	Text	True	False	10				
False	nombrec	Text	True	False	20				
False	duracion	Text	True	False	15				

Fuente: elaboración propia.

Estudiante

Database: MSAccess 2007, *Stereotype:* «table», *Package:* Data Model
Detail: *Created on* 20/01/2016. *Last modified on* 20/01/2016.
Notes:

Columns

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
True	idestud	Text	True	False	12				
False	nom1	Text	True	False	15				
False	nom2	Text	False	False	15				
False	apell1	Text	True	False	15				
False	apell2	Text	False	False	15				
False	feccha_nac	DateTime	True	False					
False	direcc	Text	True	False	30				
False	Tel	Text	False	False	10				
False	email	Text	False	False	30				
False	carrera	Text	True	False	10				
False	proyecto	Text	True	False	6				

Fuente: elaboración propia.

Proyecto

Database: MSAccess 2007, *Stereotype:* «table», *Package:* Data Model
Detail: *Created on* 20/01/2016. *Last modified on* 20/01/2016.
Notes:

Columns

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
True	codp	Text	True	False	6				
False	nombre	Text	True	False	50				
False	fecha_reg	DateTime	True	False					
False	fecha_ter	Byte	False	False					
False	observaciones	Text	False	False	50				
False	estado	Text	False	False	15				
False	tipo_proyecto	Text	False	False	5				
False	tema	Text	True	False	5				

Fuente: elaboración propia.

Tema

Database: MSAccess 2007, *Stereotype:* «table», *Package:* Data Model
Detail: *Created on* 20/01/2016. *Last modified on* 20/01/2016.
Notes:

Columns

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
True	codt	Text	True	False	5				
False	nombre	Text	True	False	20				

Fuente: elaboración propia.

Tipo_proyecto

Database: MSAccess 2007, *Stereotype:* «table», *Package:* Data Model
Detail: *Created on* 20/01/2016. *Last modified on* 20/01/2016.
Notes:

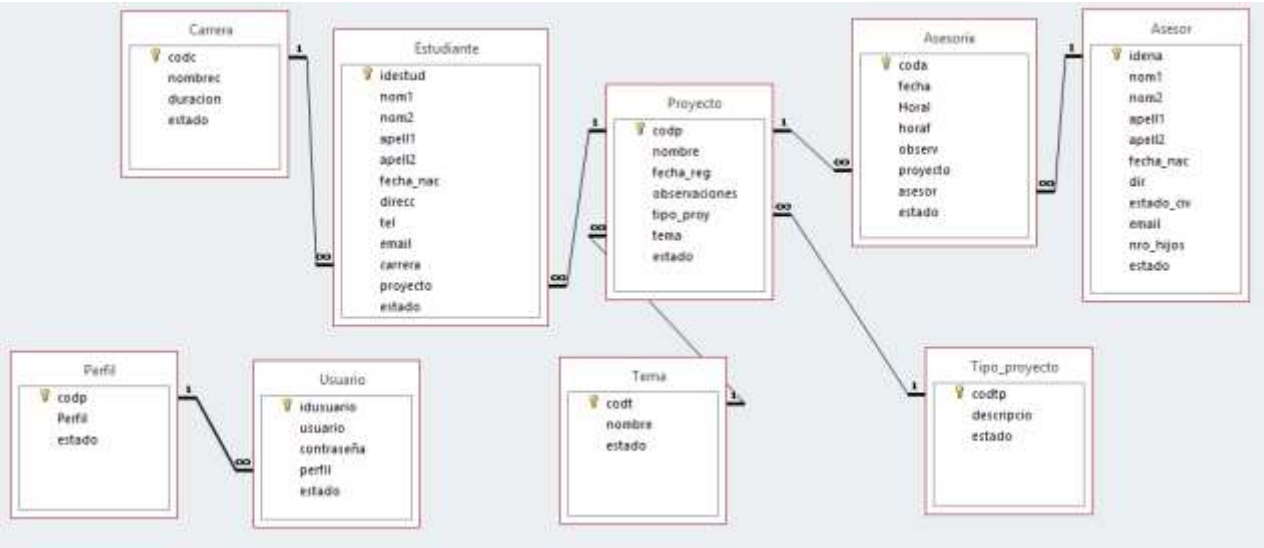
Columns

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
True	codtp	Text	True	False	5				
False	descripcion	Text	True	False	30				

Fuente: elaboración propia.

Otra herramienta, en la cual se pueden construir modelos de la base de dato puede ser Access, ya que posee potentes wizard (asistentes), para la generación de diseños, así:

Imagen 24 Modelo Relacional (Proyecto de Grado)



Fuente: elaboración propia

2.3.5 CAPA REGLAS DEL NEGOCIO O COMPONENTES

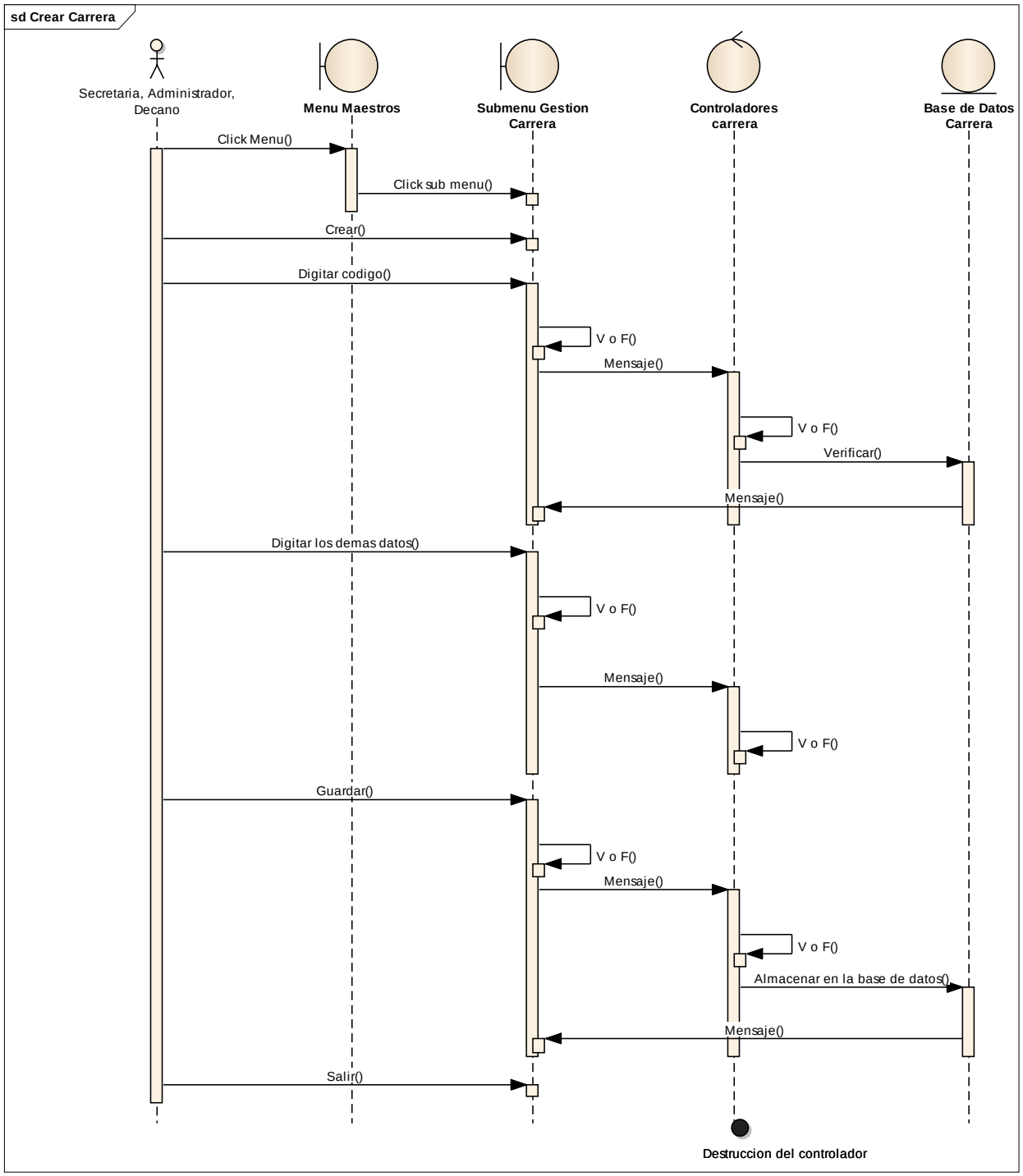
Ésta capa se relaciona con la programación, apoyada en el lenguaje de programación elegido (Java, C#, PHP, Python, entre otros). Integra la capa de la interfaz de usuario con la capa que se encarga de la administración de los datos (bases de datos). En ella se tiene en cuenta, las características que debe cumplir el estilo de programación, donde si se trata de POO (Programación Orientada a Objetos), habría que considerar características de herencia, polimorfismo, encapsulamiento, cohesión, acoplamiento, entre otras, buscando así la optimización de los diferentes procesos a nivel de comunicación, paso de mensajes, reutilización de código y componentes, ente otras, de manera que la integración de las tres capas cumpla a cabalidad con los requisitos funcionales y no funcionales del sistema. En las muchas herramientas que permiten entender dicho proceso, se encuentran los diagramas de secuencia.

- Los diagramas de secuencia:

Retomando el caso de estudio y los diagramas UML, en éste caso el diagrama de secuencia, se puede observar claramente la arquitectura por capas, donde el actor a través de las diferentes interfaces (menú, submenú, vistas, entre otras), ingresa datos, realiza peticiones al sistema, quien a través de los controladores procesa dichos datos y/o peticiones acorde a las reglas del negocio, que a su vez interactúa directamente con los datos ubicados en la diferentes estructuras y almacenamientos de bases de datos, requeridos para procesar la información, devolviendo al usuario a través de los mismos controladores (pueden ser varios) resultados procesados, acorde

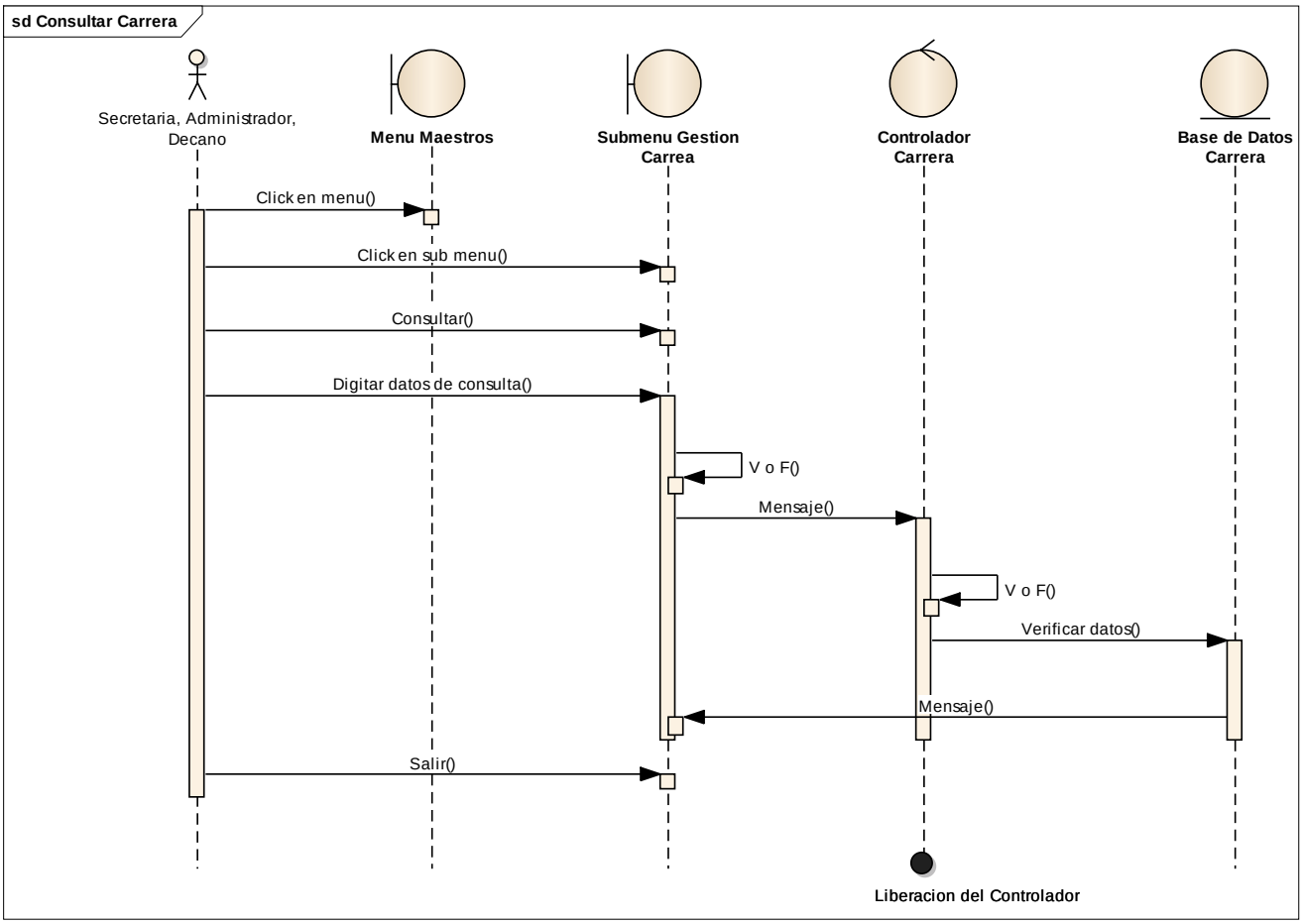
a funcionalidad de los diferentes componentes o módulos. En los siguientes diagramas de secuencia, se puede observar los requisitos funcionales (crear, consultar, modificar, Inhabilitar y cancelar), para la gestión carrera.

Imagen 25 Diagramas de secuencia (Proyecto de Grado)



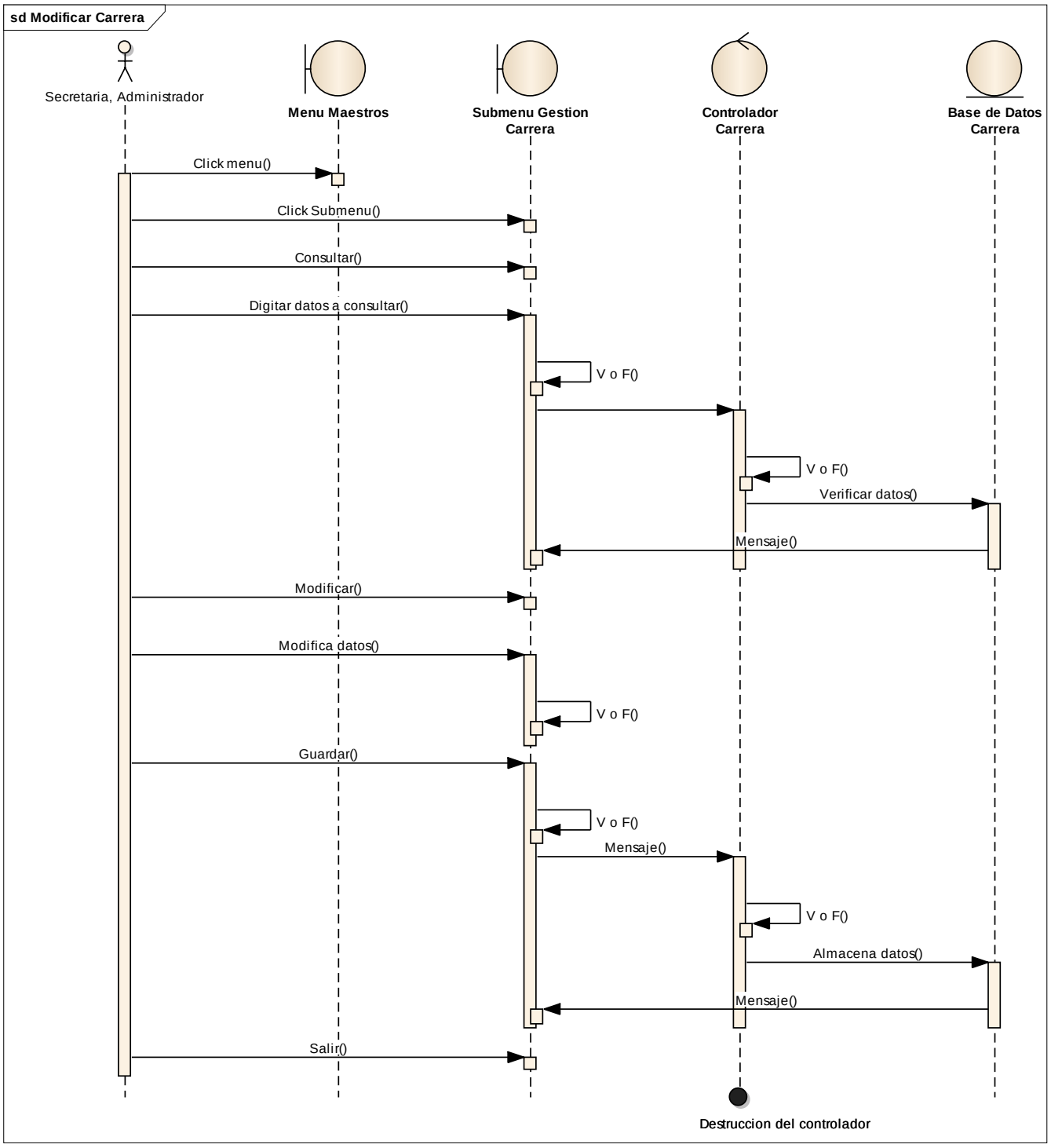
Fuente: Elaboración propia.

Imagen 26 Diagrama de secuencia Consultar Carrera (Proyecto de Grado)



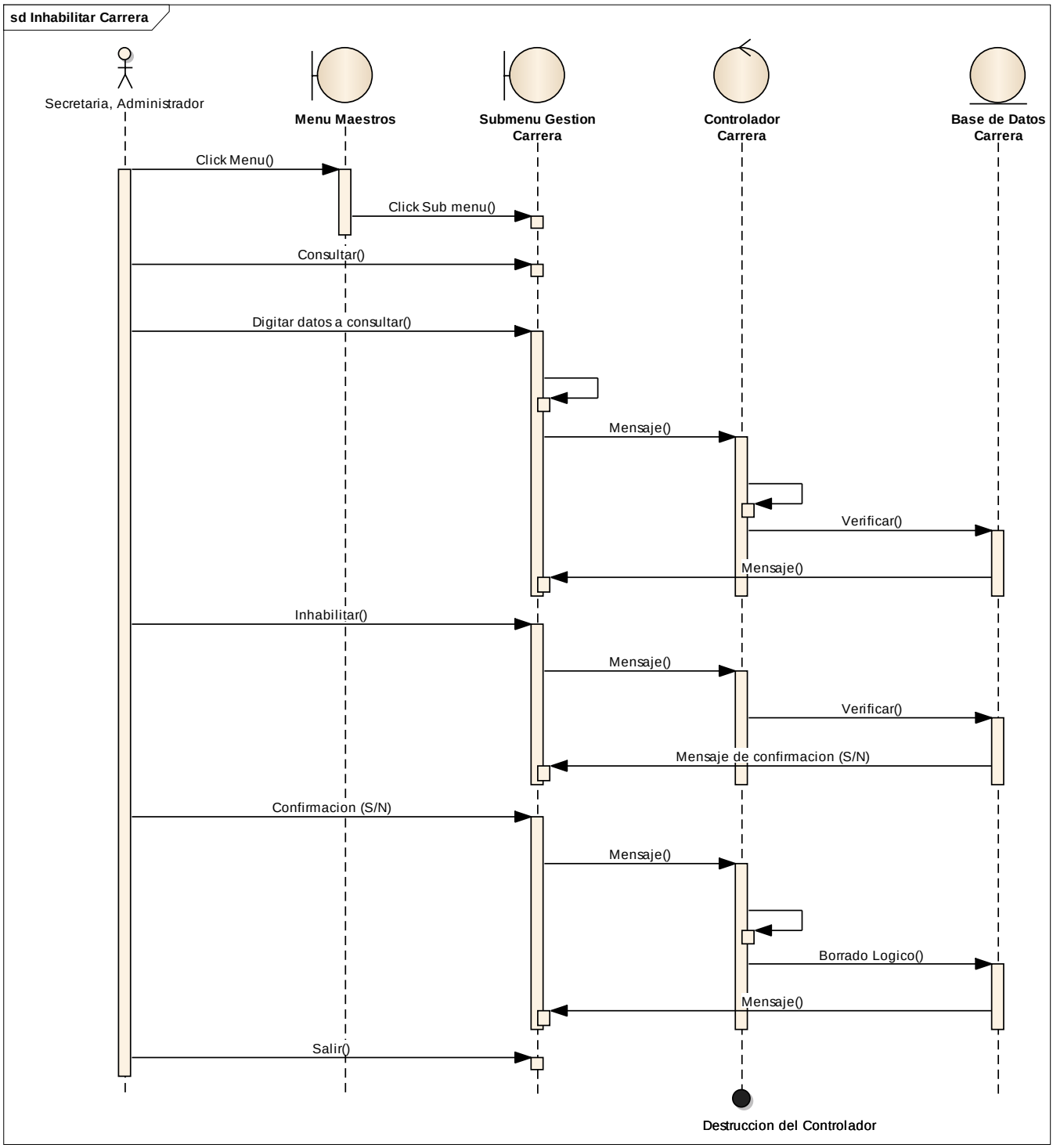
Fuente: Elaboración propia.

Imagen 27 Diagrama de secuencia Modificar Carrera (Proyecto de Grado)



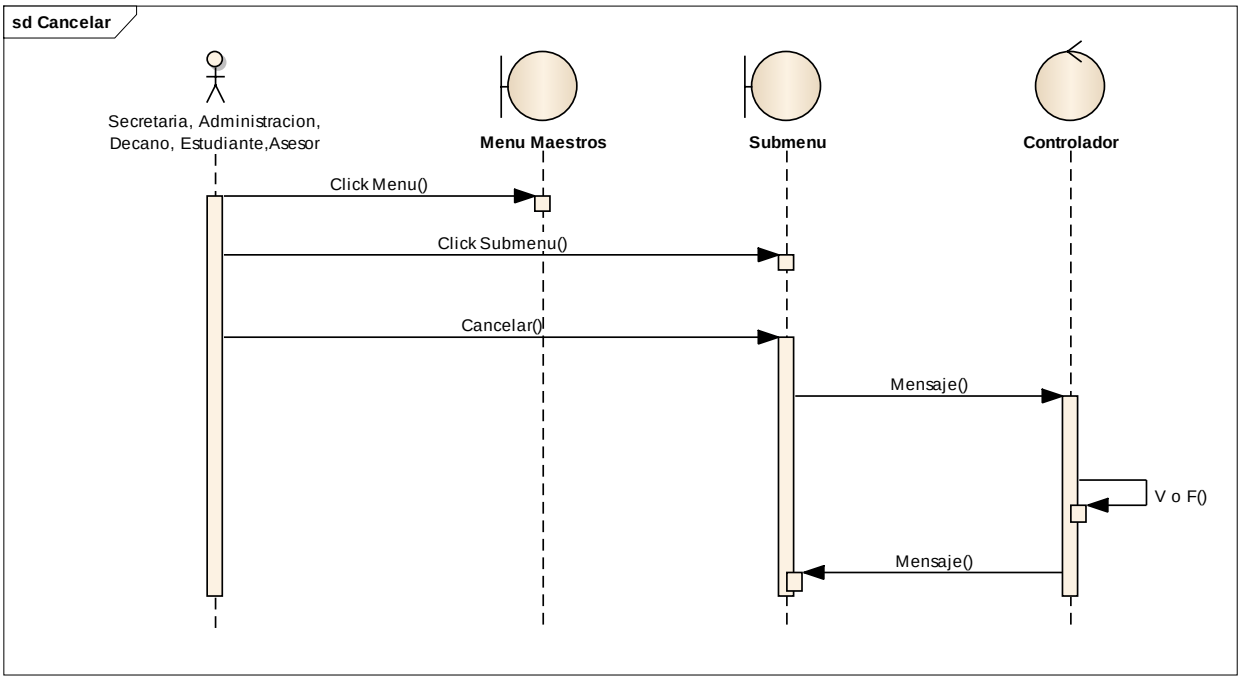
Fuente: Elaboración propia.

Imagen 28 Diagrama de secuencia Inhabilitar Carrera (Proyecto de Grado)



Fuente: Elaboración propia.

Imagen 29 Diagrama de secuencia Cancelar Carrera (Proyecto de Grado)

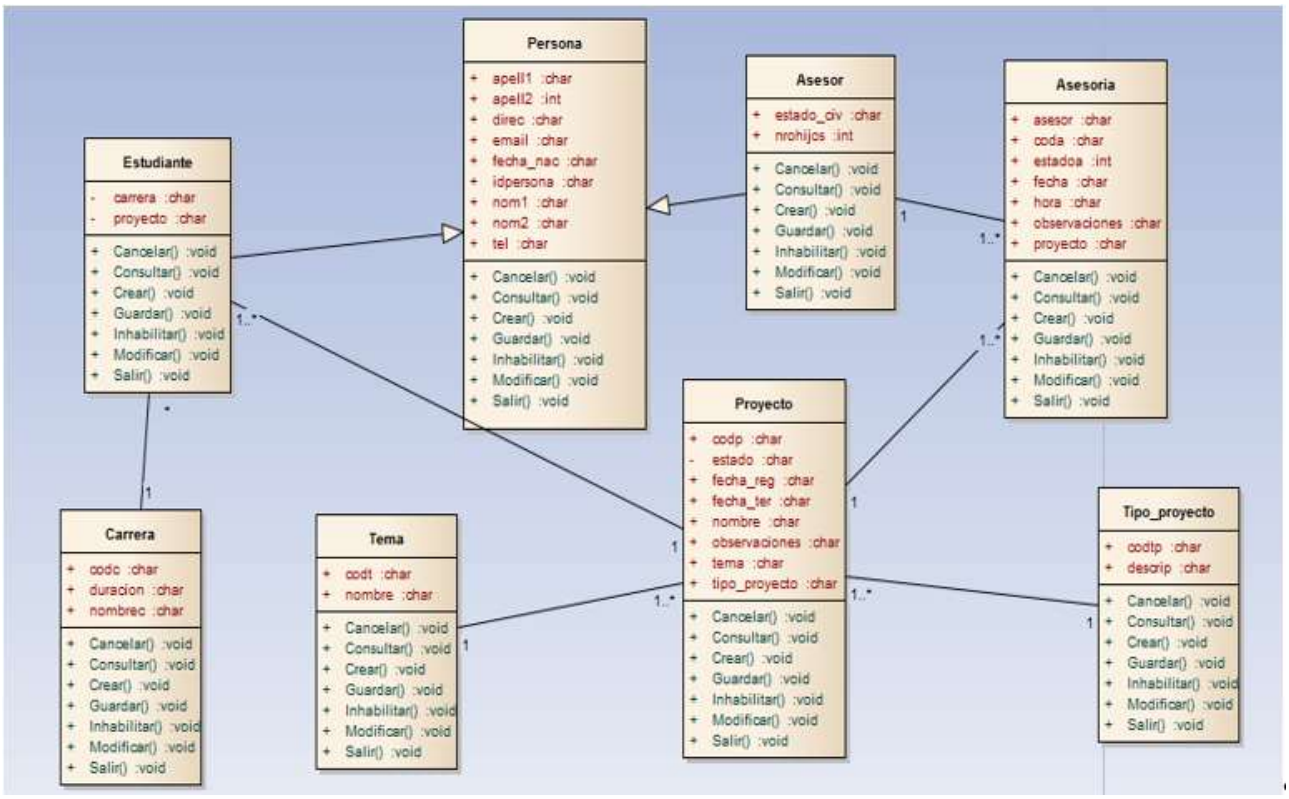


Fuente: Elaboración propia.

Modelado con estructuras estáticas

En relación a estructuras estáticas, se hace alusión en éste caso al diagrama de clases, donde se puede observar características de la POO (Programación Orientada a Objetos), como la herencia, la asociación entre clases, los diferentes atributos que se pueden utilizar en los objetos, así como los métodos que sobre cargados permiten el paso de mensajes hacia otros objetos o componentes. Es importante aclarar que los detalles en relación al lenguaje de programación, es abordado a profundidad en las diferentes materias (Lenguajes de Programación), que son vistas en varios semestres de la carrera. Ver diagramas de clases.

Imagen 30 Diagrama de Clases (Proyecto de Grado)



Fuente: elaboración propia.

A continuación, se muestra las plantillas que describen un diagrama de clase. Se debe elaborar, para cada clase una plantilla, en éste caso se ejemplificará mostrando la plantilla para la clase carrera.

Plantilla para el diagrama de Clases

La plantilla muestra cada uno de los elementos que se deben tener en cuenta antes de diseñar las clases, como soporte a los diferentes módulos dentro del desarrollo.

Imagen 31 Plantillas de diagramas de clase (Proyecto de Grado)

CLASE: CARRERA						
ATRIBUTO	**TIPO**	**VISIBILIDAD**	**DESCRIPCIÓN**			
codc	String	Public	Código de la carrera			
nombrec	String	Public	Nombre de la Carrera			
Duración	String	Public	Duración de la carrera			
Método	**Visibilidad**	**Parámetros de entrada**		**Valores que retorna**		**Descripción**
		Tipo	**Descripción**	**Tipo**	**Descripción**	
GuardarCarrera	Public	string, string, string,	Codc Nombrec Duración	int		Devuelve 1 si crea el usuario, 0 de lo contrario
ConsultarCarrera	Public	String String	Codc Nombrec			Devuelve la carrera en caso de que exista de lo contrario retorna null
ModificarCarrera	Public	string, string, string,	Codc Nombrec Duración	Boolean	…	Retorna true si se realiza la actualización, false de los contrario
InhabilitarCarrera	Public	string, string, string,	Codc Nombrec Duración	Boolean	…	Retorna true si se realiza la inactivación del registro (borrado lógico), false de lo contrario

Fuente: elaboración propia

Es importante, elegir herramientas que ayuden a optimizar procesos para la construcción del software, desde etapas tempranas. El código que se muestra en la siguiente tabla obedece a código generado por el modelador Enterprise Architect, el cual permite elegir el lenguaje de programación a utilizar, de tal forma que el código se adapte sin ninguna dificultad al lenguaje con el que se esté construyendo el software. Ver tabla.

- Generación de código para la construcción de las clases

Imagen 32 Código generado sobre clases (Proyecto de Grado)

Codificación de algunas clases (clase carrera, persona y su herencia para estudiante y asesor)

```
public class Carrera {
        public char codc;
        public char duracion;
        public char nombrec;
        public Estudiante m_Estudiante;
        public Carrera(){
        }
        public void finalize() throws Throwable {
        }
        public void Cancelar(){
        }
        public void Consultar(){
        }
        public void Crear(){
        }
        public void Guardar(){
        }
        public void Inhabilitar(){
        }
        public void Modificar(){
        }
        public void Salir(){
        }
}
```

```
public class Persona {
        public char apell1;
        public int apell2;
        public char direc;
        public char email;
        public char fecha_nac;
        public char idpersona;
        public char nom1;
        public char nom2;
        public char tel;
        public Persona(){
        }
        public void finalize() throws Throwable {
        }
        public void Cancelar(){
        }
        public void Consultar(){
        }
        public void Crear(){
        }
        public void Guardar(){
        }
        public void Inhabilitar(){
        }
```

	public void Modificar(){
	}
	public void Salir(){
	}
	}

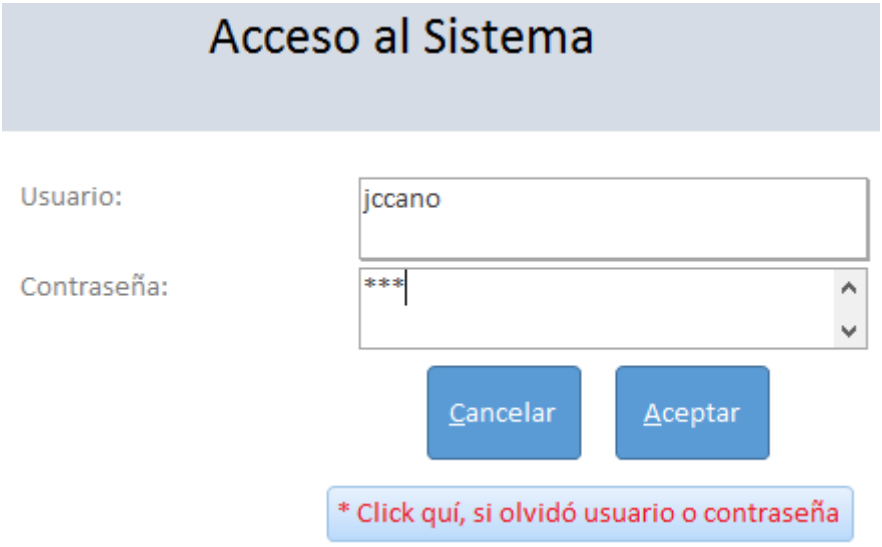
public class Estudiante extends Persona {
private char carrera;
private char proyecto;
public Proyecto m_Proyecto;
public Estudiante(){
}
public void finalize() throws Throwable {
super.finalize();
}
public void Cancelar(){
}
public void Consultar(){
}
public void Crear(){
}
public void Guardar(){
}
public void Inhabilitar(){
}
public void Modificar(){
}
public void Salir(){
}
}
	public class Asesor extends Persona {
	public char estado_civ;
	public int nrohijos;
	public Asesoria m_Asesoria;
	public Asesor(){
	}
	public void finalize() throws Throwable {
	super.finalize();
	}
	public void Cancelar(){
	}
	public void Consultar(){
	}
	public void Crear(){
	}
	public void Guardar(){
	}
	public void Inhabilitar(){
	}
	public void Modificar(){
	}
	public void Salir(){
	}
	}

Fuente: Código Generado en el proceso de IS

2.3.6 CAPA INTERFAZ DE USUARIO O PRESENTACIÓN

La interfaz de usuario tiene como objetivo ofrecer una interacción intuitiva, de tal forma que el diseño se centre en el usuario, facilitando que éste se comunique de forma eficiente, el diseño debe ofrecer claridad, elegancia, facilidad de entender, de accionar, dando respuestas oportunas, acorde a solicitudes del usuario. Actualmente las interfaces pueden utilizarse a través de varios medios como por ejemplo el teclado, mouse, lápiz óptico, táctil, voz, entre otras formas que permiten esa interacción entre usuario y sistema. A continuación se presenta una propuesta sencilla, sobre diseño, aplicada al sistema (Proyecto de Grado).

Imagen 33 Interfaz de acceso al sistema (Proyecto de Grado)



Fuente: elaboración propia.

Los menús más usuales son los que se muestran en los siguientes diseños:

Menú colgante o desplegable: Se puede observar en la parte superior (Aspirantes, Estudiantes, Docentes…)

Imagen 34 Menú colgante o desplegable

Fuente: http://www.uniremington.edu.co/

Menú de Botones de Comando: Se puede observar (Uniremington, Sedes, Programas…)

Imagen 35 Menú de Botones de Comando

Fuente: http://www.uniremington.edu.co/

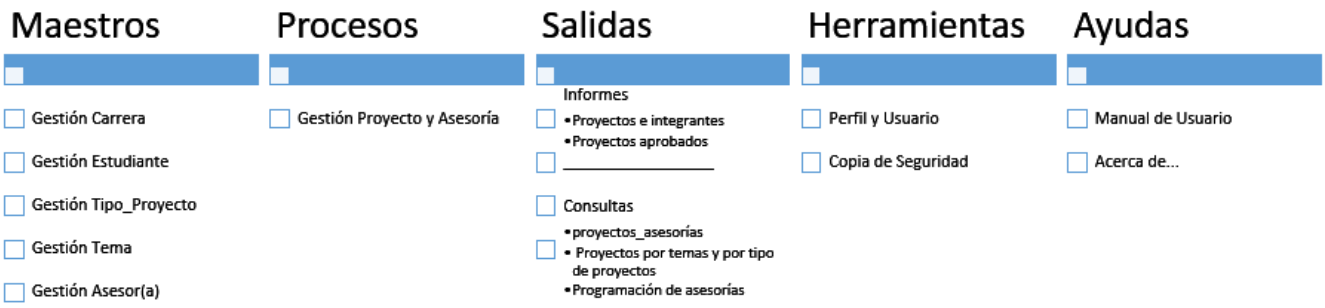
Menú en forma de árbol: Se puede observar en Cursos, donde al activarlo despliega su submenú, o sea que se expande y al tocarlo nuevamente de comprime.

Imagen 36 Menú en forma de árbol

Fuente: https://virtual.uniremington.edu.co/home/

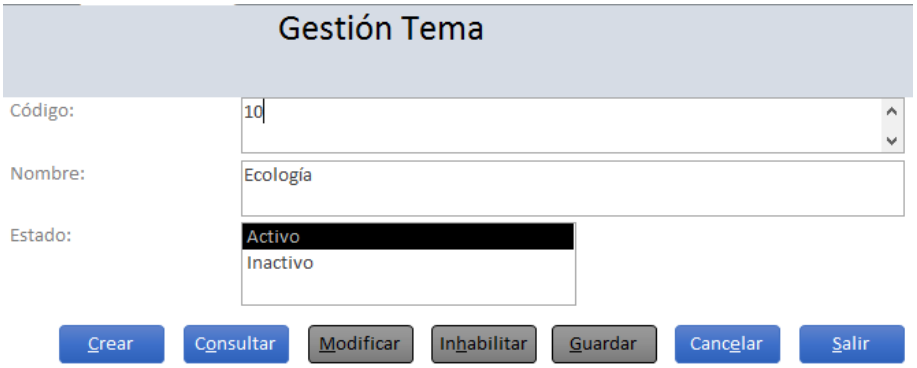
Continuando con el ejemplo del Proyecto de grado, se muestra un estilo de menú tradicional (Colgante o desplegable).

Imagen 37 Menú principal con todos los permisos de usuario (Proyecto de Grado)



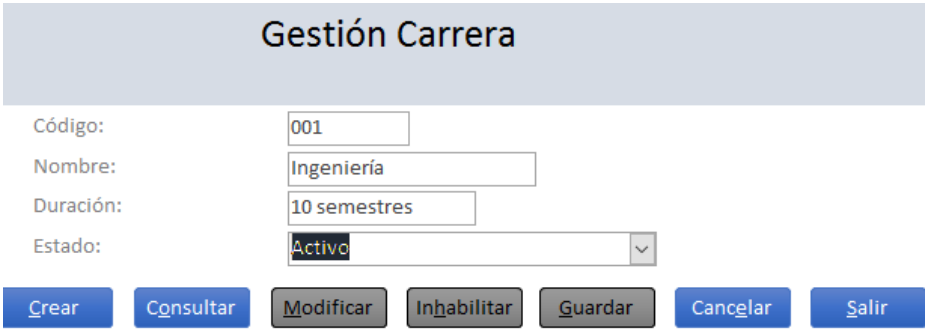
Fuente: elaboración propia.

Imagen 38 Interfaz módulo (Gestión Tema)



Fuente: elaboración propia.

Imagen 39 Interfaz módulo (Gestión Carrera)



Fuente: elaboración propia.

Imagen 40 Interfaz módulo (Gestión Estudiante)

Fuente: elaboración propia.

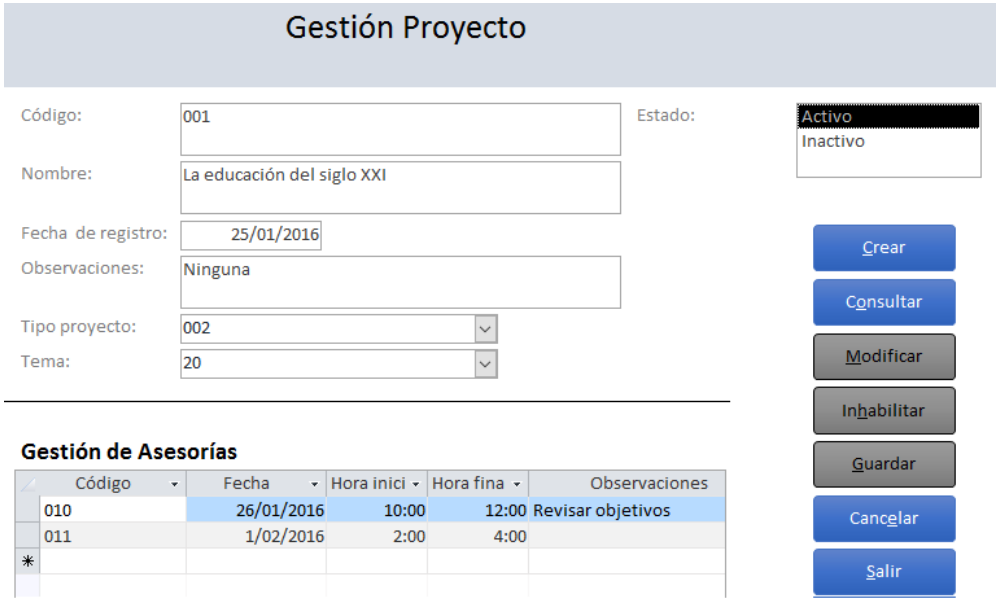
Imagen 41 Interfaz módulo (Gestión Proyecto)

Fuente: elaboración propia.

Imagen 42 Interfaz módulo (Gestión Asesor)

Fuente: elaboración propia.

Imagen 43 Interfaz módulo (Gestión Proyecto y asesorías)

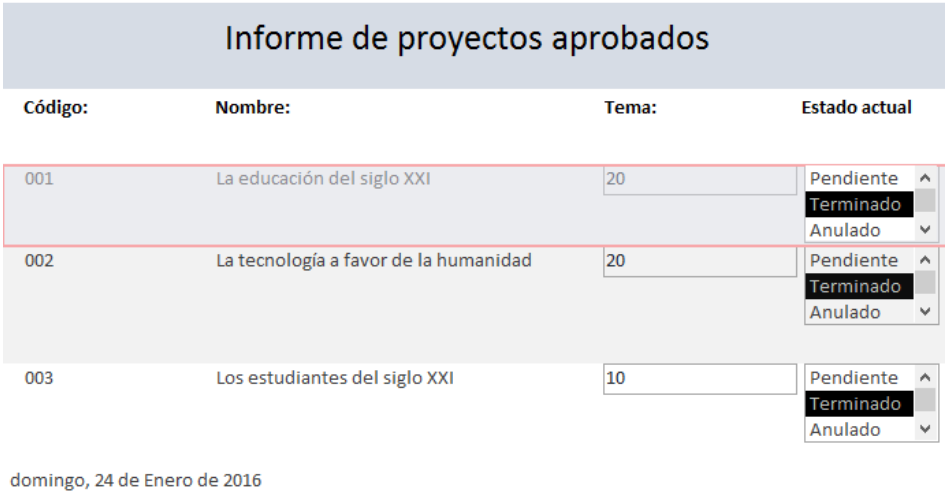


Fuente: elaboración propia.

Imagen 44 Interfaz módulo (Administración Perfil y Usuarios)

Fuente: elaboración propia.

Imagen 45 Interfaz Informe (Proyectos aprobados)



Fuente: elaboración propia.

Imagen 46 Interfaz Informe (Proyectos y sus integrantes)

Informe de proyectos y sus integrantes

Código:	Nombre:	Tema:	Nombre:	Primer nombre:	Segundo nombre:	Primer apellido:
001	La educación del siglo XXI	20	Educación			
				Juan	Camilo	Rios
				Sara	Elena	Castro
				Felipe	Esteban	Hoyos
002	La tecnologia a favor de la humanidad	20	Educación			
				Santiago		Vélez
				Clara		Hincapié

domingo, 24 de Enero de 2016 Página 1 de 1

Fuente: elaboración propia.

El diseño de la interfaz, puede variar de acuerdo al tipo de proyecto, tipo de cliente, tipo de empresa desarrolladora de software, por lo tanto el que se mostró anteriormente, no es una camisa de fuerza para los diferentes diseños.

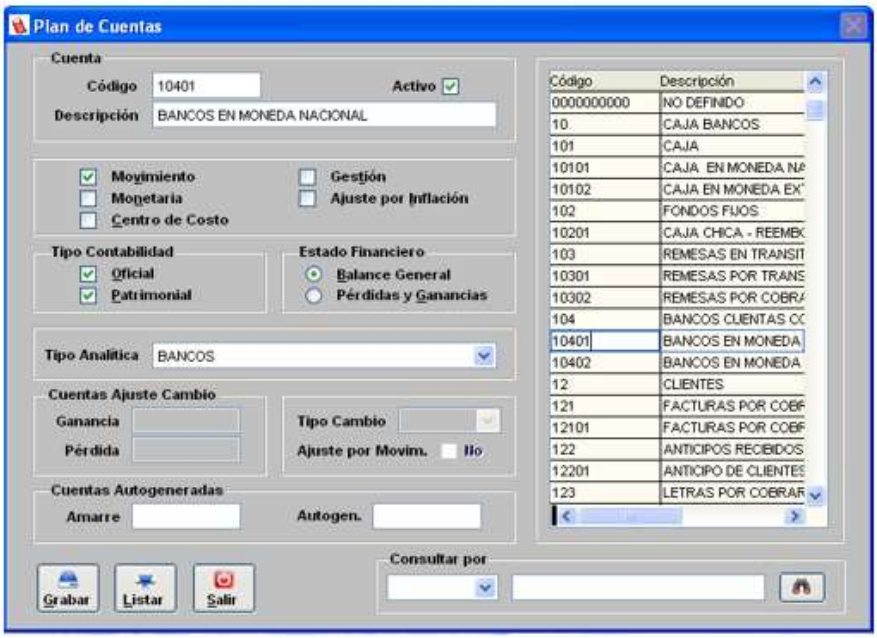
2.3.7 OTROS EJEMPLOS DE DISEÑO DE INTERFAZ

Ejemplo de interfaz para ingreso al sistema

Imagen 47 Otros ejemplos para diseño de software (Interfaz de usuario)

Fuente: Varios pantallazos de programas.

Imagen 48 Ejemplo de software contable



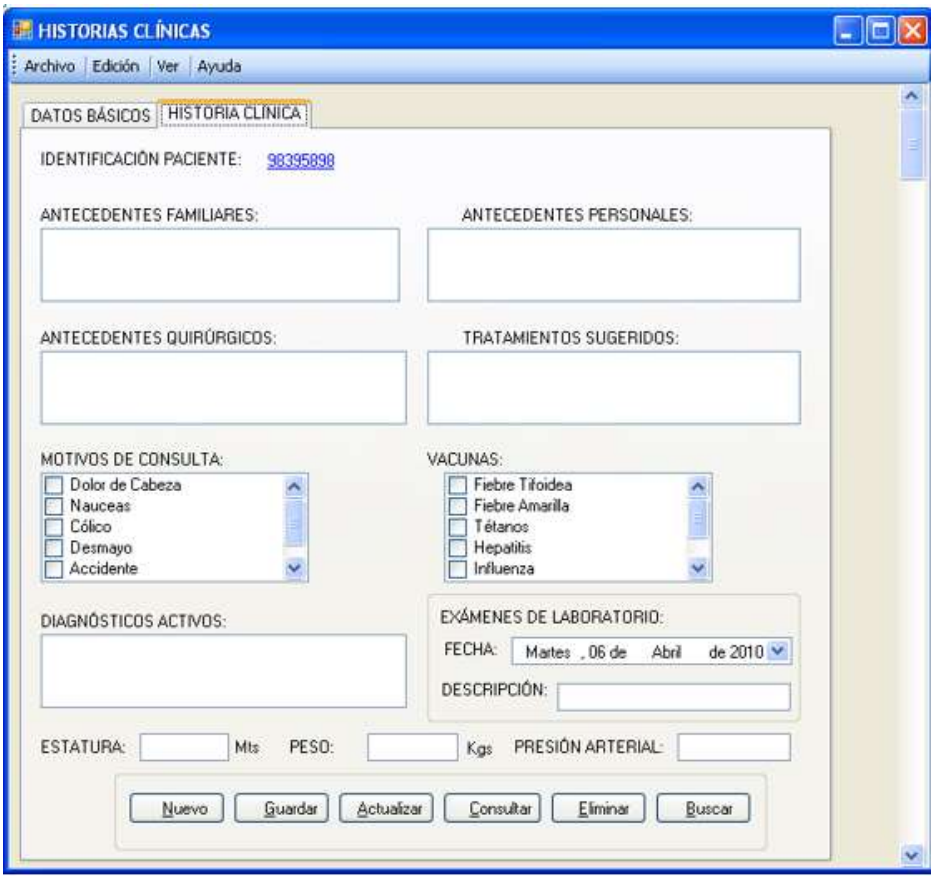
Fuente: programa contable

Imagen 49 Ejemplo Software educativo

Fuente: programa educativo

Fuente: programa educativo

Imagen 50 Ejemplo software Historias Clínicas



Fuente: Software historias clínicas.

Una interfaz de usuario debe ser coherente debe integrar la interacción del usuario y la presentación de la información. En la siguiente tabla, se muestra algunas sugerencias para la obtención de diseños de calidad.

Imagen 51 Algunos consejos para el diseño

Estilo de interacción	Principales ventajas	Ejemplo de aplicación
Manipulación Directa	• Interacción rápida e intuitiva. • Fácil de aprender.	Videojuegos.
Selección de Menús	• Evita errores del usuario. • Se requiere digitar un poco.	Muchos software de propósito general.
Llenado de Formularios	• Introducción de datos sencilla. • Fácil de aprender.	Control de almacén. Procesamiento de préstamos en una biblioteca.
Lenguaje de Comandos	• Flexible	Sistemas bibliotecarios de recuperación de la información.
Lenguaje Natural	• Accesible a usuarios casuales. • Fácil de ampliar.	Sistemas para el manejo de horarios. Sistemas web de recuperación de información.

Fuente. Ingeniería de Software. Ian Somerville

2.4 TEMA 4: ANÁLISIS Y EVALUACIÓN DEL DISEÑO DE SOFTWARE

Durante el proceso del ciclo de vida para la construcción del software, se hace indispensable la evaluación de cada uno de sus procesos, siendo necesario realizar análisis detallado del diseño de software, ya que éste deberá recoger todos los requisitos, deberá proporcionar altos niveles de comprensión para los procesos siguientes al diseño, deberá permitir la integración de todos sus componentes, deberá permitir la realización de pruebas durante su proceso de construcción, terminación e implementación, entre otras características que deberá cumplir un diseño bajo lineamientos de calidad.

Para realizar validaciones al diseño de software, Juristo, Moreno & Vegas (2006), propone listas de chequeo, con una serie de preguntas que ayudan con la verificación de proceso y de su funcionalidad, las que se relacionan a continuación.

2.4.1 EJEMPLO DE LISTA DE COMPROBACIÓN PARA EL DISEÑO ARQUITECTÓNICO

• ¿Es la organización del sistema clara, incluyendo una buena visión general de la arquitectura y su justificación?

• ¿Están todos los módulos bien definidos, incluyendo su funcionalidad e interfaces con otros módulos?

• ¿Se cubren todas las funciones que aparecen en los requisitos?

• ¿Se han descrito y justificado todas las estructuras de datos más importantes?

• ¿Se han ocultado todas las estructuras de datos con funciones de acceso?

• ¿Se ha especificado la organización y contenidos de la base de datos?

• ¿Se han descrito y justificado los objetos principales?

• ¿Se ha modularizado la interfaz de usuario de tal forma que los cambios en ella no afectarán al resto del programa?

• ¿Se ha descrito alguna estrategia para gestionar la entrada del usuario?

• ¿Se han definido los aspectos principales del interfaz de usuario?

• ¿Se describe y justifica una estrategia para el manejo de entradas y salidas?

• ¿Se incluye una estrategia de manejo de errores coherente?

• ¿Se ha diseñado la arquitectura para acomodar cambios probables?

• ¿Encaja conceptualmente cada parte de la arquitectura para formar un todo?

2.4.2 EJEMPLO DE LISTA DE COMPROBACIÓN PARA DISEÑO DE ALTO NIVEL

• ¿Es el diseño del sistema actual es consistente con el diseño de sistemas relacionados?

• ¿Gestiona adecuadamente el diseño asuntos que fueron identificados y postergados al nivel de la arquitectura?

• ¿Es satisfactoria la forma en la que el programa se ha descompuesto en módulos u objetos?

• ¿Es satisfactoria la forma en que los módulos se han descompuesto en rutinas?

• ¿Se han definido bien todas las fronteras de los subsistemas?

• ¿Se han diseñado los subsistemas para la interacción mínima de unos con otros?

• ¿Tiene sentido el diseño recorriéndolo tanto de arriba abajo como de abajo a arriba?

- ¿Distingue el diseño entre componentes pertenecientes al dominio del problema, componentes de interfaz de usuario, componentes de gestión de tareas y componentes de gestión de datos?
- ¿Tiene el diseño complejidad baja?
- ¿Será el sistema fácil de mantener?
- ¿Reduce el diseño las conexiones entre subsistemas a la mínima cantidad?
- ¿Permite el diseño extensiones futuras al sistema?
- ¿Están diseñados los subsistemas de tal modo que se pueden usar en otros sistemas?
- ¿Será sencillo portar el diseño a otro entorno?
- ¿Es el diseño balanceado? ¿Son todas sus partes estrictamente necesarias?
- ¿Está estratificado el diseño en niveles?
- ¿Usa el diseño técnicas estándar y evita elementos exóticos, difíciles de entender?

Es importante resaltar, que cada empresa desarrolladora de software, posee sus propias líneas base, donde define sus propios procesos, metodologías, estrategias y técnicas para la construcción de productos de calidad.

2.4.3 TALLER DE ENTRENAMIENTO UNIDAD 1

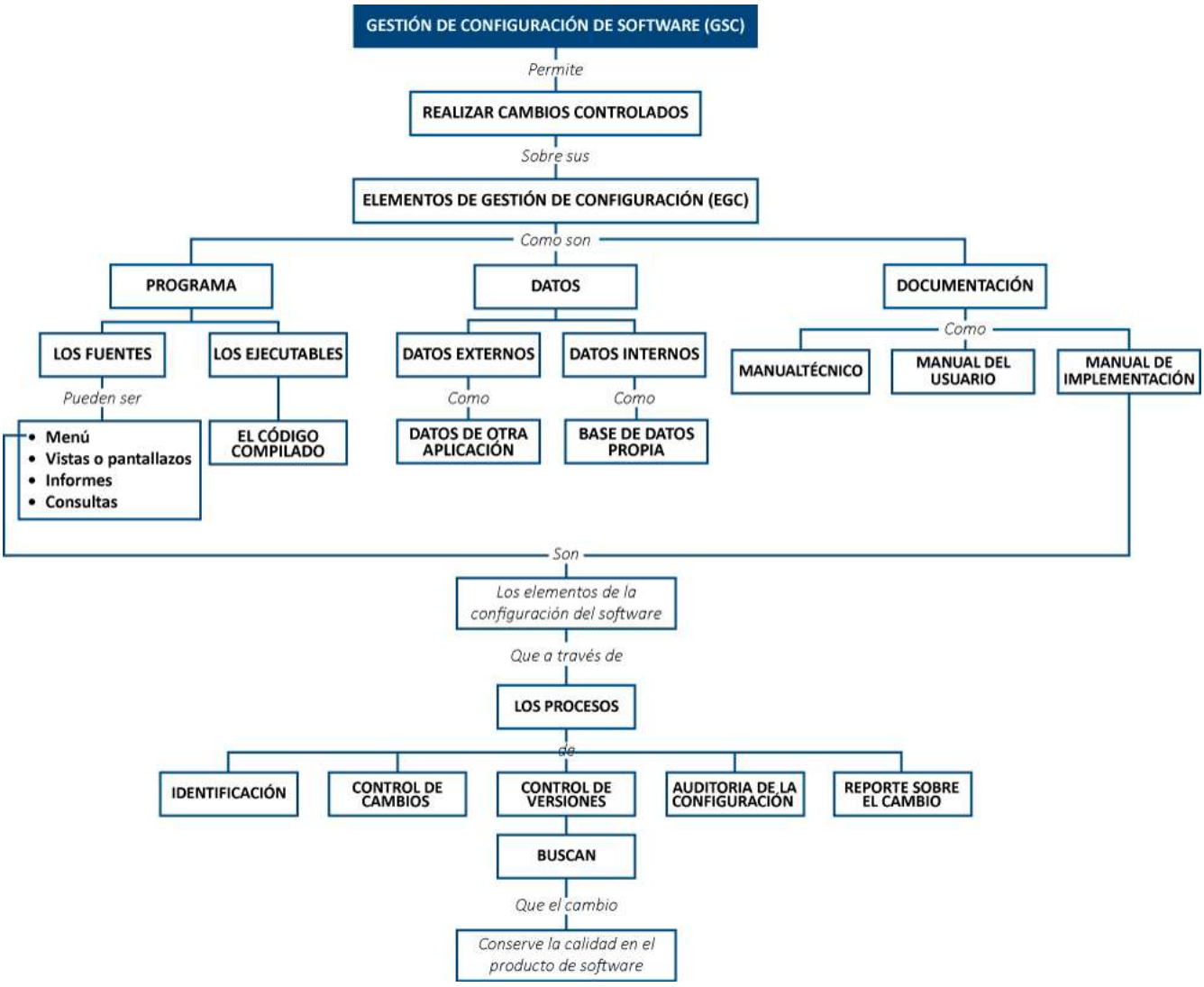
Nombre del taller: Taller 1	Modalidad de trabajo: Aprendizaje basado en problemas
Actividad previa: Estudio de la Unidad 1 del módulo	
Describa la actividad: **De acuerdo a la situación problemática que se presenta:** 1. Elaborar la tabla General para Casos de Uso. 2. Elaborar el modelo relacional (identificar entidades, asignar atributos, identificar claves primarias, relacionar, aplicar primeras tres formas normales). 3. Realizar el diseño de las interfaces o vistas de usuario (Menú, Acceso al sistema, formularios, informes). **Situación problemática (Trabajada desde el módulo Ingeniería de Software I)** Un parqueadero ubicado en el centro de la ciudad, tiene dificultades con el control de ingreso y salida de vehículos, lo que genera pérdida de tiempo y retraso tanto a clientes como a empleados, generando cogestión, stress, desorganización entre otros, es así como se requiere sistematizar su servicio de parqueos. El software	

debe manejar la información de los vehículos que ingresan al parqueadero, así como el respectivo registro que se genera, el cual debe hacerse al ingreso de este, también debe imprimir el recibo para el pago por horas. El parqueadero también ofrece mensualidades. Dependiendo del tipo de vehículo que pueden ser camiones, buses, busetas, automóviles será el costo de la hora o de la mensualidad. Para el parqueadero también es importante tener la información de los clientes que llevan sus vehículos, así como la información del empleado que registra el servicio y expide el recibo de pago.

Es de anotar que la forma como se está llevando actualmente el manejo del parqueadero es manual, en un cuaderno, donde en ocasiones se pierde la información o no se encuentra en el momento oportuno.

La persona encargada de manejar todo el sistema, será una secretaria que se encuentra ubicada en la cabina de entrada del parqueadero.

3 UNIDAD 2 GESTIÓN DE CONFIGURACIÓN DE SOFTWARE



Fuente: elaboración propia.

3.1 TEMA 1: IDENTIFICACIÓN, CONFIGURACIÓN Y CONTROL DE LA GESTIÓN DE CONFIGURACIÓN DEL SOFTWARE.

Tanto el proyecto, proceso, como el producto de software, durante el ciclo de vida, es decir desde que aparece la necesidad o requerimiento, hasta que el producto sale del mercado, está susceptible a cambios,

ya sea porque cambian las condiciones comerciales, por reorganización, crecimiento o reducción del negocio, restricciones de presupuesto o de planificación que provocan una redefinición del sistema o producto, entre otras variables que se pueden presentar, hace que dentro del proceso de Ingeniería de Software, existan herramientas que permitan controlar esos cambios inevitables que deben asegurar que el producto de software, se pueda adaptar, asegurando que el software cumpla con los requisitos funcionales y no funcionales como lo plantea SWEBOK (cuerpo del conocimiento de la ingeniería de software), donde además aborda los fundamentos de la GCS (Gestión de Configuración del Software), como por ejemplo: identificar, controlar, garantizar e informar el cambio aplicado tanto al proyecto, como al proceso o producto de software. Ver documento Swebok (Versión 3 de 2014).

3.1.1 CONCEPTUALIZACIÓN SOBRE LA GESTIÓN DE CONFIGURACIÓN DEL SOFTWARE (GCS)

La gestión de configuración del software es una actividad de autoprotección que se aplica para controlar los cambios a lo largo del ciclo de vida del software. Sus objetivos son:

- Identificar los cambios
- Controlar los cambios
- Control de versión
- Auditar e informar de los cambios realizados a los interesados.
- Realizar reportes o informes sobre los cambios

3.1.2 LO QUE PUEDE GENERAR UN CAMBIO

Un cambio se puede aplicar por muchas causas, algunas de ellas pueden ser debido a:

- Nuevos negocios o condiciones comerciales que dictan los cambios en los requisitos del producto o en las normas comerciales.
- Nuevas necesidades del cliente que requieren la modificación de los datos producidos por el sistema.
- Reorganización y crecimiento o reducción del negocio que provoca cambios en las prioridades del proyecto o en la estructura del equipo de Ingeniería de Software.
- Restricciones de presupuesto o de planificación que provocan una redefinición del sistema o producto.

3.1.3 DIFERENCIA ENTRE MANTENIMIENTO Y GESTIÓN DE CONFIGURACIÓN DE CAMBIOS

Es importante aclarar que existe diferencia entre mantenimiento de software y gestión de configuración del software, el **mantenimiento** es un conjunto de actividades de Ingeniería de Software que se producen después de que se ha entregado al cliente y esté en funcionamiento, en relación a l**a gestión de configuración del software** es un conjunto de actividades de seguimiento y control que comienza cuando se inicia el proyecto de ingeniería de software y termina sólo cuando el software queda fuera de circulación.

3.1.4 ELEMENTOS DE LA CONFIGURACIÓN DE SOFTWARE (ECS)

Dentro del proceso de Gestión de Configuración del Cambio, se tienen varios elementos, los cuales pueden ser susceptibles a cambios como son los datos, los programas y la documentación:

- **Datos:** Todo lo relacionado a los datos internos y externos que se relacionen con el sistema como por ejemplo la estructura de la base de datos (tablas nuevas o modificadas, relaciones, datos provenientes de otras aplicaciones, entre otros)

- **Programas (Fuentes y Ejecutables):** Se relaciona con todo cambio que se encuentre comprometido, con interfaces (vistas de usuario, menú, informes, consultas, reglas del negocio, configuración del sistema para compatibilidad con otros sistemas como por ejemplo los sistemas operativos, en los cuales opera el sistema).

- **Documentación:** Compuesto por los manuales técnicos, manuales de usuario y manual de implementación. En el manual técnico debe figurar la información que tiene que ver con la construcción del sistema, diagramas, esquemas, modelos, diseños, entre otros donde se deje evidencia de cada cambio que se realizó, en lo referente al manual del usuario, los nuevos componentes deben tener orientación clara para facilitar al usuario final el manejo del componente creado o adaptado al proceso de configuración. Es importante además que el manual de implementación posea los pasos claros sobre la forma como el usuario debe hacer el proceso de iniciación del producto de software, tanto a nivel de requerimientos técnicos como instrucciones de instalación. La siguiente imagen muestra los elementos de configuración de software susceptibles de cambios o nuevas implementaciones.

Fuente: elaboración propia.

GESTIÓN DE CONFIGURACIÓN DEL SOFTWARE		
Programas de computadora (frente y ejecutable)	Documentos que describen los programas de computadora (manual técnico y manual de usuario, manual de implementación)	Datos (contenidos en el programa o externos a el)
Elementos de configuración del software (ECS)		

3.1.5 PROCESO DE GESTIÓN DE CONFIGURACIÓN DE SOFTWARE

Para aplicar el proceso de GCS, se sugiere la ejecución de procesos que buscan garantizar la calidad en cada uno de los cambios solicitados por el cliente. Dichos procesos, se pueden apreciar en la siguiente imagen y se describen a continuación.

Imagen 52 Proceso de Gestión de Configuración de Software (GCS)

Fuente: elaboración propia.

A continuación se describe cada uno de los procesos a aplicar en la GCS:

Identificar los cambios: Consiste en identificar los objetos básicos y/o objetos compuestos con sus características o atributos, relaciones entre objetos, evoluciones, módulos, listados, nombres de variables, entre otros sobre los cuales recaen actualizaciones o modificaciones. Siendo necesario documentarlos adecuadamente.

Controlar los cambios: Esta actividad combina los procedimientos humanos y las herramientas automáticas. A nivel de Ingeniería de Software es preocupante el cambio, ya que una diminuta perturbación en el código puede crear un gran fallo en el producto, pero también puede reparar un gran fallo o habilitar excelentes capacidades nuevas. El cambio incontrolado lleva rápidamente al caos.

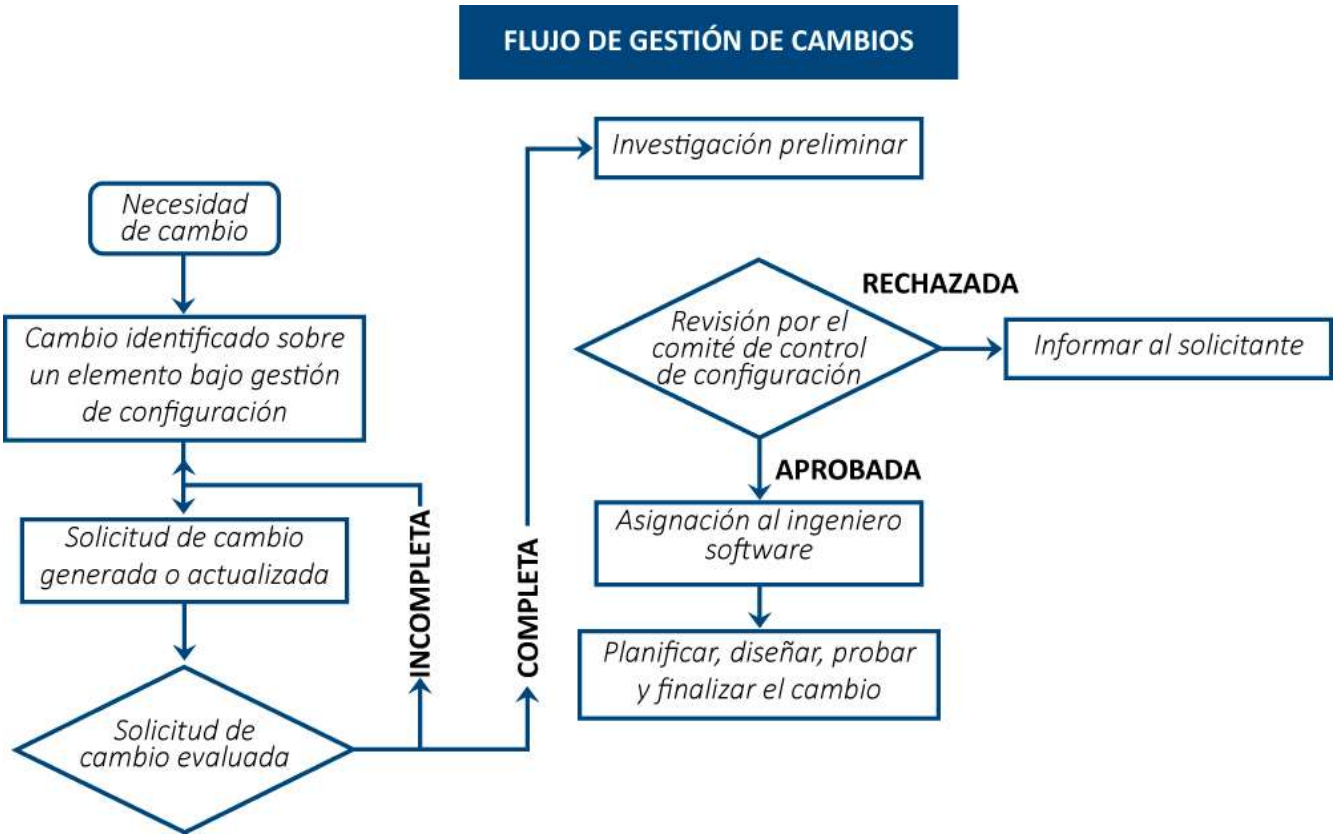
Para controlar el cambio, se proponen los siguientes pasos:

- Recibir la solicitud o petición de cambio por parte del cliente.

- Evaluar la Petición de Cambio (Esfuerzo técnico, efectos secundarios, impacto global sobre otras funciones del sistema y sobre otros objetos de configuración).
- Informe de cambios a la Autoridad de Control de Cambios (ACC) quien toma la decisión, evaluando impacto del cambio en software, hardware, rendimiento, calidad, fiabilidad entre otras)
- Orden de Cambio de Ingeniería (OCI) (cambio a realizar, restricciones que deben respetar, criterios de revisión y de Auditoría).

En la siguiente imagen se puede visualizar el proceso para controlar el cambio.

Imagen 53 Flujo de Gestión de Cambios



Controlar las versiones: Una nueva versión se define cuando se realizan cambios significativos en uno o más objetos del software. El control de versiones combina procedimientos y herramientas para gestionar las versiones de los objetos con configuraciones alternativas asociando atributos particulares a cada versión, creados durante el proceso del software. Es así que, para construir la variante, de una determinada versión de un programa, a cada componente (objetos compuestos u objetos básicos), se le asigna un registro que define si se ha de utilizar el componente cuando se va a construir una determinada versión. Es importante

tener presente que si el cambio solicitado por el cliente es de dimensiones mayores, se deberá ser claro con él, explicándole detalladamente sobre los efectos que dicha solicitud podría generar en el resto de los componentes del software, donde en muchas ocasiones sería mejor esperar una nueva versión, donde el cliente tendría beneficios tanto económicos, como de calidad en el producto versionado.

3.2 TEMA 2: AUDITORÍA Y ADMINISTRACIÓN DE LA CONFIGURACIÓN DEL SOFTWARE

La verificación del proceso de Gestión de Configuración de Software, permite comprobar que efectivamente el producto que se está construyendo es lo que pretende ser, ya que cualquier cambio debe someterse a actividades de verificación y validación, siendo apropiado que dicha auditoría se realice por personal externo al proceso de cambio que se esté aplicando.

De igual forma cualquier cambio que sea aceptado, debe ser corroborado, con el fin de evidenciar la satisfacción del requisito, su adecuación, completitud, precisión acorde a la línea base establecida para tal proceso y especialmente, si el o los elementos de Configuración del Software se comportarán correctamente una vez que éstos se encuentran en el ambiente del usuario final.

En relación a la administración del cambio, éste permite identificar la configuración teniendo en cuenta los tiempos, los cambios realizados, teniendo en cuenta la integridad y la rastreabilidad de la configuración a través del ciclo de vida del sistema, donde es importante tener presente las restricciones, los cambios de versión , los procesos para realizar la GCS, estado de la configuración, el manejo de líneas base, entre otros aspectos que garanticen que el cambio fue debidamente administrado.

Continuando con los pasos para el Proceso de Gestión de Configuración de Software (GCS) como son identificar los cambios, controlar los cambios, control de versión, auditar e informar de los cambios realizados a los interesados, realizar reportes o informes sobre los cambios, se describen a continuación la auditoría y el reporte o informe de gestión del cambio.

3.2.1 AUDITAR LOS CAMBIOS REALIZADOS

El control de cambios ayuda al equipo de desarrollo de software a mantener un orden, que de no ser así llevaría a una situación caótica y sin salida. Se debe:

Realizar Revisiones Técnicas Formales: La cual se centra en la corrección técnica del elemento que ha sido modificado, la evaluación del ECS, para determinar consistencia con otros ECS, las omisiones o los posibles efectos secundarios.

Auditorías de Configuración de Software: Además de lo anterior se plantean preguntas que permitan verificar que el proceso efectivamente se realizó y que además de ello se obtuvo buenos resultados. Las preguntas podrían ser similares a las siguientes:

- ¿Se ha hecho el cambio específico en la OCI?
- ¿Se ha incorporado modificaciones adicionales?
- ¿Se ha llevado a cabo una revisión técnica formal para evaluar la corrección técnica?
- ¿Se ha seguido el proceso o línea base establecida para la gestión de cambio?
- ¿Se ha aplicado adecuadamente los estándares de ingeniería de Software?
- ¿Se ha resaltado los cambios en el ECS?
- ¿Se ha especificado la fecha de cambio y el autor?
- ¿Reflejan los cambios los atributos del objeto de configuración?
- ¿Se ha seguido procedimientos de GCS para señalar el cambio, registrarlo y divulgarlo?
- ¿Se ha actualizado adecuadamente todos los ECS relacionados?

3.2.2 INFORMAR LOS CAMBIOS REALIZADOS A LOS INTERESADOS

Proporciona información sobre cada cambio realizado al sistema, así como los demás elementos del software que se vieron comprometidos, ellos con el fin de llevar información histórica sobre las modificaciones realizadas en el proceso de construcción del software o en el producto terminado. El informe o reporte de gestión de cambios, debe cumplir mínimo con los siguientes interrogantes:

- ¿Qué pasó?
- ¿Quién lo hizo?
- ¿Cuándo pasó?

- ¿Qué más se vio afectado?

Para registrar dicha evidencia, se recomienda hacer uso de una base de datos interactiva, donde se registre cada vez que:

- Se asigna un ECS (Elemento de Configuración de Software)
- La ACC (Autoridad de Control de Cambios) aprueba un cambio.
- Se expide una OCI (Orden de Cambio de Ingeniería).
- Se lleva a cabo una auditoria de configuración.

Buscan con ello que dicha GCS, se encuentre lo suficientemente documentada, para ser utilizada en el momento que se requiera.

3.3 TEMA 3. FORMATOS PROPUESTOS PARA APLICAR PROCESOS DE GESTIÓN DE CONFIGURACIÓN DEL SOFTWARE

Para sistematizar el proceso de gestión de configuración de software, se pueden aplicar diversas herramientas, las cuales son adaptadas de acuerdo a las diferentes necesidades de las empresas, proyectos y clientes. Para el caso de estudio, se trabajará con los siguientes formatos, los que facilitan el proceso de Gestión de Configuración de Software:

3.3.1 FORMATO ORDEN DE SOLICITUD DE CAMBIO

Estudio orden de solicitud del cambio
Fecha:_____**Hora:**_____ **Responsable**:_____**Cargo**:_____
Requerimiento (situación actual, justificación, descripción
Situación actual: Estado del proyecto o proceso actual.
Justificación: Detallar la importancia que tiene el cambio en el mejoramiento del proyecto o proceso.
Descripción: Detallar la necesidad que refleja el cliente.

Fuente: elaboración propia.

3.3.2 FORMATO EVALUACIÓN Y ANÁLISIS DE SOLICITUD DEL CAMBIO

Evaluación y análisis de solicitud del cambio
Fecha:_____**Hora:**_____ **Responsable**:_____**Cargo**:_____
Análisis de datos: (Agregación de tablas, modificación de las ya existentes, relaciones, manejo de datos existentes, migración de datos, manejo de estructuras por fuera de la base de datos entre otros). Se debe mostrar modelos actuales y modelo propuesto, donde se resalte las mejoras y los procesos a tener en cuenta.
Análisis de Aplicativo y Ejecutables: Se detalla la actualización en formularios, consultas, informes, menú, módulos, componentes, sistemas operativos u otros sistemas que intervengan en la aplicación, cambios en ejecutables.
Documentación **Manual Técnico:** (como nuevos modelos, script de migración, y población de datos, script de generación de ejecutables, configuraciones a nivel de parametrización a nivel de sistemas operativos, hardware, base de datos entre otros.) **Manual del usuario:** Describir la funcionalidad del sistema informático con los nuevos cambios. **Manual de implementación**: Define las actividades con su orden y secuencia que se debe desarrollar para la correcta implementación del sistema informático. En el orden se incluirá las actividades y prerrequisitos que permitan garantizar la correcta operatividad de sistema. Estas se deberán especificar, una vez se tenga definido el diseño detallado del sistema configurado.

Fuente: elaboración propia.

3.3.3 FORMATO TIEMPO PRESUPUESTADO PARA LA GCS

Tiempo Presupuestado	
Fecha:_____ **Hora:**_____	
Responsable:_____ **Cargo**:_____	
ACTIVIDAD	**TIEMPO (horas, días, semanas…)**
Evaluación	
Diseño	
Construcción	
Pruebas	
Documentación	
Tiempo total	

Fuente: elaboración propia.

3.3.4 FORMATO EVALUACIÓN ESTUDIO DE LA ORDEN DE CAMBIO

Evaluación del estudio de la orden de cambio			
Fecha:_____ **Hora:**_____			
Responsable:_____ **Cargo**:_____			
Nota: la ACC (Autoridad de Control de Cambios evalúa (impacto del cambio en software, hardware, rendimiento, calidad, fiabilidad, aprueba los ajustes definidos, si están dentro del producto y cumplen con la lista de chequeo)			
Nro.	Ítems	Estado	
		Si	No
1	Se tiene identificada la arquitectura en la que estará soportado el sistema: Servidor de base de datos, estaciones de trabajo.		
2	Se tiene las características de hardware donde se va a ejecutar el sistema, lógica del negocio, servicios de almacenamiento.		
3	Se describen los requerimientos de software sobre los cuales se construirá el sistema: Sistemas operativos, bases de datos y herramientas de desarrollo.		
4	Se tienen los diagramas de clases, de colaboración, el modelo de datos, las interfaces, consultas, informes y archivos requeridos para cada servicio a desarrollar.		
5	Se describe las características que han sido identificadas durante la etapa de diseño.		
6	Se elaboró cronograma de trabajo.		
7	Se tiene documentado el proceso de diseño.		
8	Se tiene identificado los usuarios del sistema.		
9	Se tienen identificado los procesos.		
10	Se identificaron las relaciones entre los usuarios y los procesos		
11	Se describen las entradas del sistema		
12	Se describen las salidas del sistema		
13	Se identificaron las interfaces para las pantallas		
14	Se identificaron las interfaces para las consultas		
15	Se identificaron las interfaces para los informes		
16	Se identificaron las interfaces para los archivos		
17	Se identificaron los posibles errores y excepciones.		
18	Se tienen documentos que soporten el proceso de análisis		

Fuente: elaboración propia.

3.3.5 FORMATO ORDEN DE CAMBIO DE INGENIERÍA (OCI)

orden de cambio de ingeniería (OCI)

Fecha:_____**Hora:**_____

Responsable:_____**Cargo**:_____

Nota: Se valora los tiempos presupuestados, se asignan los roles por actividad, se genera el contrato respectivo el cual debe ser firmado por las dos partes donde se consigna la fecha de inicio, y la de entregas parciales (si son acordadas), la fecha de entrega definitiva. De igual forma se define el tiempo de garantía del cambio y el cronograma.

Actividad	Responsable	Tiempo (estimado, real en horas, días, semanas…)	
		TE	TR

Nota: Después de generarse el cambio, se debe chequear el desarrollo de la OCI
Aplicación de RTF (Revisión Técnica Formal)

Fuente: elaboración propia

3.3.6 FORMATO INFORME GERENCIAL SOBRE EL CAMBIO DE GESTIÓN DE CAMBIOS

Informe gerencial sobre el cambio de gestión de cambios

Fecha:_____**Hora:**_____

Responsable:_____**Cargo**:_____

Actividad	Responsable	Fecha	Otros cambio
¿Qué paso?	¿Quién lo hizo?	¿Cuándo paso?	¿Qué más se vio afectado?

Fuente: elaboración propia.

3.4 TEMA 4. EJEMPLO PRÁCTICO SOBRE GESTIÓN DE CONFIGURACIÓN DE SOFTWARE (SITUACIÓN PROBLEMÁTICA, PROYECTO DE GRADO)

Situación práctica

Para la situación que se viene abordando, en relación al sistema para la gestión de los Proyectos de Grado, se aplicará un proceso de Gestión de Configuración de Software:

Necesidad de Cambio:

Después de tener implementado el sistema, el decano requiere que se tenga la información sobre la especialidad de los asesores, con el fin de asignar los asesores de acuerdo a la cualificación de éstos, buscando que las asesorías de los proyectos tengan mayor calidad en relación a las temáticas que se estén desarrollando. (Es importante que el cliente realice una solicitud escrita, donde clarifique el cambio que requiere para su sistema).

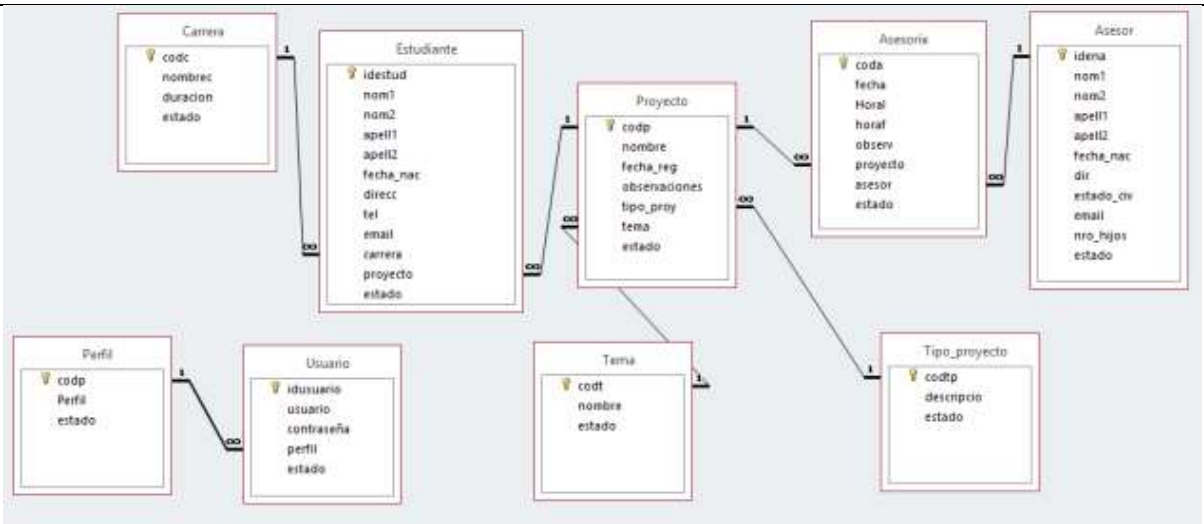
3.4.1 ESTUDIO SOLICITUD DE CAMBIO (PROYECTO DE GRADO)

ESTUDIO ORDEN DE SOLICITUD DEL CAMBIO
Fecha: 21-01-16 **Hora: 02:00pm** **Responsable**: Juan Camilo Ríos **Cargo**: Líder de proyectos
Requerimiento (situación actual, justificación, descripción
Situación actual: Estado del proyecto o proceso actual. El software se encuentra activo en la máquina del cliente, cumpliendo con los requisitos funcionales y no funcionales establecidos, desde su construcción, acorde a necesidad relacionada con el manejo de las asesorías para los Proyecto de Grado de la Facultad de Ciencias Básicas e Ingeniería (caso que se viene abordando desde el módulo Ingeniería de Software I). El cliente requiere que se tenga la información sobre la especialidad de los asesores, con el fin de asignar asesores de acuerdo a la cualificación de éstos, buscando que las asesorías de los proyectos tengan mayor calidad en relación a las temáticas que se estén desarrollando.

Justificación: Detallar la importancia que tiene el cambio en el mejoramiento del proyecto o proceso. Con la evolución de los diferentes procesos, manejo de información, agilidad, versatilidad, consistencia en los datos e información. Es importante mantener actualizado el software, de acuerdo a los diferentes cambios de los procesos, es el caso de los Proyecto de Grado, siendo importante el manejo de la cualificación del asesor, ya que esto permite que las asesorías, estén orientadas por personal idóneo, lo que permite que los Proyecto de Grado, logren mejorar su nivel académico, tanto a nivel interno de la facultad como de toda la universidad.
Descripción: Detallar la necesidad que refleja el cliente. El cliente requiere que el sistema esté en capacidad de manejar la información relacionada con la cualificación académica del docente (asesor), con el fin de asignarle proyectos que se relacionen con las temáticas que éste maneja, buscando mayor eficiencia en las diferentes asesorías que permitan la evolución de los Proyecto de Grado de los estudiantes.

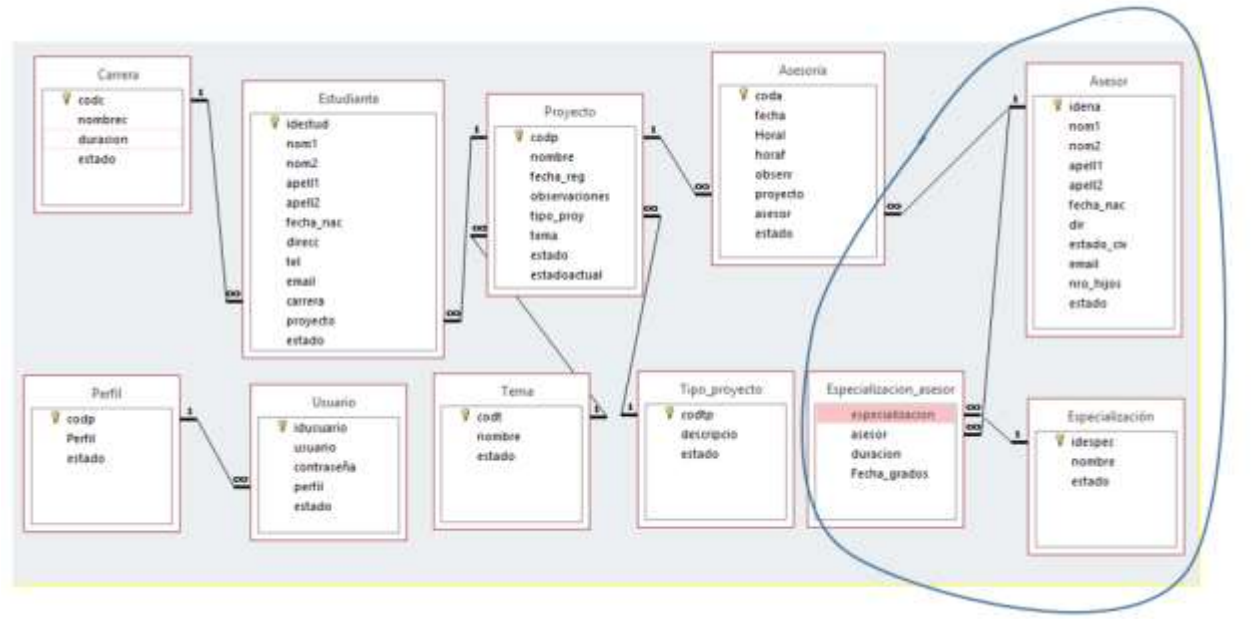
3.4.2 EVALUACIÓN Y ANÁLISIS DE SOLICITUD DE CAMBIO (PROYECTO DE GRADO)

EVALUACIÓN Y ANÁLISIS DE SOLICITUD DEL CAMBIO
Fecha: 25-01-16 **Hora: 02:00pm** **Responsable**: Carlos Mario Marín **Cargo**: Analista de Software
Análisis de datos: (Agregación de tablas, modificación de las ya existentes, relaciones, manejo de datos existentes, migración de datos, manejo de estructuras por fuera de la base de datos entre otros). Se debe mostrar modelos actuales y modelo propuesto, donde se resalte las mejoras y los procesos a tener en cuenta.
1. Se realiza revisión de las bases de datos relacionadas directamente con el sistema actual, encontrándose la siguiente:



2. Se analiza protección de datos (copia de seguridad), con el fin de garantizar que la información que se posee actualmente almacenada en la base de datos, no se pierda o sufra alteraciones.
3. Al revisar el requerimiento del cliente, se encuentra que:

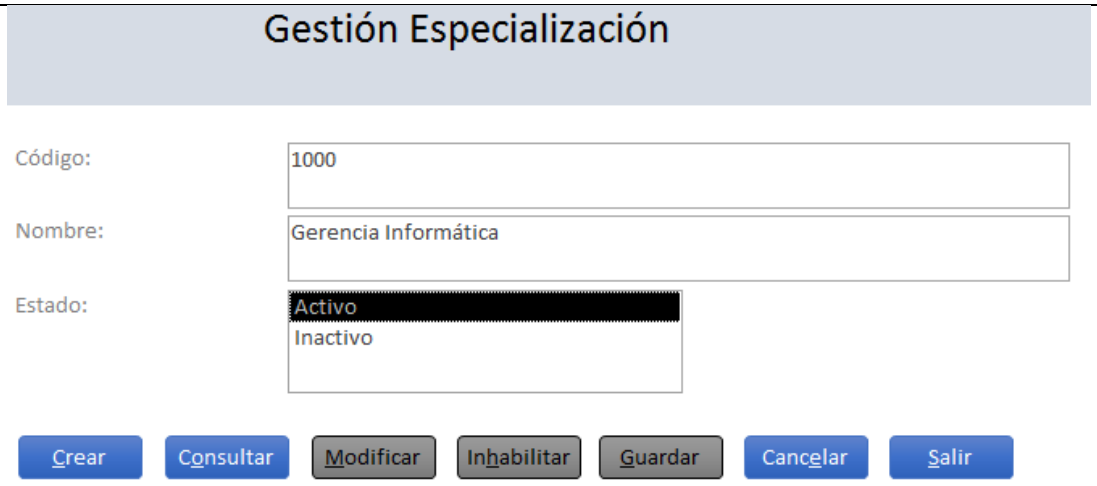
- Se requiere crear una nueva tabla llamada Especialización con los siguientes campos (código de la especialización, nombre de la especialización)
- Existe relación de muchos a muchos entre especialización y asesor, por lo tanto se requiere crear una tabla intermedia, que puede llamarse especialización asesor con los siguientes campo (clave primaria de asesor, clave primaria de especialización y otros campos complementarios como duración, fecha de graduación, entre otros), conservando la integridad de acuerdo a las formas normales para estandarización de las bases de datos.
- Realizar las respectivas validaciones de la estructura de la base de datos.
- Recuperar la información, desde la copia de seguridad hacia la nueva estructura.
- Modelo actualizado de la base de datos.



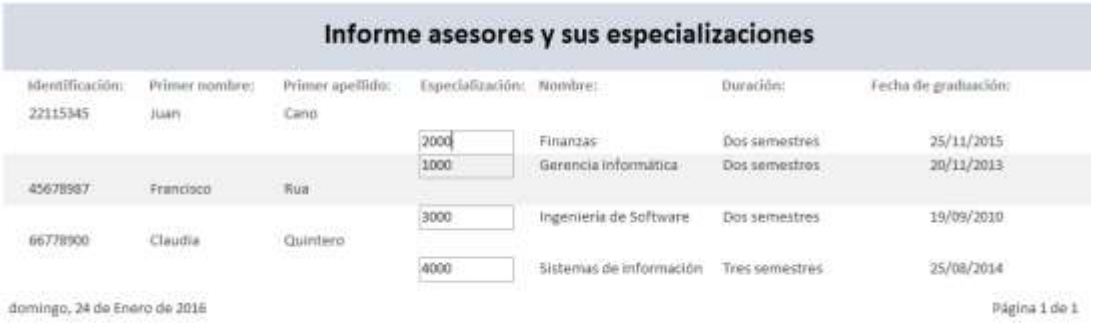
Análisis de Aplicativo y Ejecutables: Se detalla la actualización en formularios, consultas, informes, menú, módulos, componentes, sistemas operativos u otros sistemas que intervengan en la aplicación, cambios en ejecutables.

1. La vista de usuario asesor, sufrirá modificación, ya que se incorpora la información relacionada con los estudios de los asesores:

2. Se crea una vista de usuario nueva llamada especialización, donde se aplica toda la validación de la vista, la programación de los diferentes controles, así como enlaces con la base de datos.



3. Se diseña informe asesor y estudios



4. Se modifica el menú del sistema, agregando dos menú (Gestión Especialización e Informe Asesores y sus especializaciones).

5. Se realizan las diferentes pruebas al software, tanto pruebas unitarias que corresponden a las realizadas por los desarrolladores, como pruebas de integración relacionadas con las posibles variaciones que pueden generar éstos cambios al resto de los elementos funcionales del sistema informático.

6. Se empaqueta el producto, revisando su funcionalidad, teniendo en cuenta los diferentes sistemas operativos, que el cliente puede utilizar.
7. Despliegue del producto en la máquina del cliente, con pruebas por parte del usuario, para la aceptación de los cambios.

Documentación

Manual Técnico: (como nuevos modelos, script de migración, y población de datos, script de generación de ejecutables, configuraciones a nivel de parametrización a nivel de sistemas operativos, hardware, base de datos entre otros.)

Realizar la especificación del requisito

Tabla general para casos de uso

Gestiones	Acciones	Actores				
		Secretaria	Decano	Estudiante	Asesor	Administrador
Gestión Especialización	Crear	X				X
	Consultar	X	X			X
	Modificar	X				X
	Guardar	X				X
	Inhabilitar	X				X
	Cancelar	X	X			X
	Salir	X	X			X

Diseñar los casos de uso

Identificación de los requisitos de usuario

Requisitos de Usuario (RU)			
Id Requisito	**Nombre del requisito**	**Descripción del requisito**	**Usuario**
RU-012	Gestión Especialización	El sistema permitirá la gestión de la información de las especializaciones para Crear, Modificar, Inhabilitar, Consultar, Cancelar y Salir	Secretaria, Administrador, Decano

Análisis de los Requisitos Funcionales

Requisitos Funcionales (RF)				
Id Requisito	**Nombre del requisito**	**Descripción del requisito**	**Usuario**	**Id Requisito de usuario**
RF-068	Crear Especialización	Permite registrar la información de las especializaciones con los siguientes campos código, nombre y estado de la especialización. A través de opción guardar.	Secretaria, Administrador	RU-012
RF-069	Consultar Especialización	Permite consultar la información de las especializaciones por los campos código y nombre.	Secretaria, Administrador, Decano	RU-012
RF-070	Modificar Especialización	Permite modificar la información de las especializaciones a excepción del código, utilizando la opción guardar.	Secretaria, Administrador	RU-012
RF-071	Inhabilitar Especialización	Permite habilitar o inhabilitar una especialización, con previa confirmación de realizar el proceso.	Secretaria, Administrador	RU-012
RF-072	Cancelar Especialización	Permite cancelar el proceso que se esté realizando y regresar al estado inicial.	Secretaria, Administrador, Decano	RU-012
RF-073	Salir Especialización	Permite cerrar el formulario	Secretaria, Administrador, Decano	RU-012

Construcción de los escenarios

NOMBRE	CREAR ESPECIALIZACIÓN	
Descripción	Permite registrar las Especializaciones con los siguientes datos: Código, nombre, duración y estado de la Especialización.	
Actor	Secretaria, Administrador, Decano.	
Precondiciones	El usuario debe estar logueado en el sistema	
FLUJO BÁSICO		
Pasos	Actor	Sistema
	1. El usuario ingresa al menú de Maestros / Gestión Especialización.	2. El sistema despliega la interfaz con las opciones: Código, nombre, y estado de la Especialización; deshabilitados, con los botones crear, consultar y salir activos y los demás inactivos.
	3. El usuario da clic en botón Crear.	4. El sistema activa todos los campos y envía el cursor a Código y activa los botones cancelar y guardar.
	5. El usuario ingresa el código da enter.	El sistema verifica si el código existe.
	5. El usuario ingresa los demás datos de la Especialización y va dando enter, para llenarlos todos.	6. El sistema valida los campos requeridos.
	7. El actor da clic en guardar.	8. El sistema almacena y arroja un mensaje de transacción exitosa.
	9. Clic en salir.	
FLUJO ALTERNATIVO		
Pasos	Actor	Sistema

		10. El sistema confirma que hay inconsistencia en los datos y arroja el mensaje de alerta.
	1. El usuario da click en el botón cancelar	2. El sistema ubica el cursor en el código
	13. El usuario ingresa los datos correctamente y da clic en guardar.	14. El sistema verifica los datos nuevamente.
		15. El sistema almacena y arroja un mensaje de transacción exitosa.
	16. Clic en Salir	
Post-Condiciones	Existe una nueva Especialización Registrada.	
Requisito Funcional	RF-068	

Fuente: Elaboración propia

Tabla 1 Escenario Consultar Especialización (Sistema Proyecto de Grado)

NOMBRE	CONSULTAR ESPECIALIZACIÓN	
Descripción	Permite consultar la información de las Especializaciones.	
Actor	Secretaria, Administrador, Decano	
Precondiciones	El usuario debe estar logueado en el sistema.	
FLUJO BÁSICO		
Pasos	Actor	Sistema
	1. El usuario ingresa al menú de maestros/ Gestión Especialización	2. El sistema despliega la interfaz con las opciones Código, nombre, y estado de la Especialización; deshabilitados, con los botones crear, consultar y salir activos y los demás inactivos
	3. El usuario da clic en botón consultar.	4. El sistema activa los campos Código, nombre, de la Especialización para que el usuario realice la búsqueda por

		cualquiera de estos campos. Activa los botones modificar, inhabilitar y cancelar
	5. El usuario digita los datos para la búsqueda y da enter.	6. El sistema valida que los datos ingresados sean de una Especialización existente.
		7. El sistema muestra los datos y todos los campos en estado inactivo.
	8. Clic en salir.	
FLUJO ALTERNATIVO		
Pasos	Actor	Sistema
		9. El sistema confirma que los datos ingresados no son de una Especialización existente y arroja el mensaje de alerta.
	10. El usuario ingresa los datos correctamente y da enter.	11. El sistema valida los datos nuevamente.
		12. El sistema muestra los datos y todos los campos en estado inactivo.
	13. Clic en salir	
Post-condiciones	Se consultó una Especialización existente.	
Requisito funcional	Rf-069	

Fuente: Elaboración propia

Tabla 2 Escenario Modificar Especialización (Sistema Proyecto de Grado)

NOMBRE	MODIFICAR ESPECIALIZACIÓN
Descripción	Permite modificar la información de las Especializaciones
Actor	Secretaria, administrador
Precondiciones	El usuario debe estar logueado en el sistema y haber dado clic en el botón consultar.
FLUJO BÁSICO	

Pasos	Actor	Sistema
	1. El usuario ingresa al menú de maestros/ Gestión Especialización.	2. El sistema despliega la interfaz con las opciones Código, nombre y estado de la Especialización; deshabilitados, con los botones crear, consultar y salir activos y los demás inactivos.
	3. El usuario da clic en botón consultar.	4. El sistema activa los campos Código, nombre para que el usuario realice la búsqueda por cualquiera de estos campos y activa los botones modificar, inhabilitar, cancelar.
	5. El usuario digita los valores para la búsqueda y da enter.	6. El sistema valida que los datos ingresados sean de una Especialización existente.
		7. El sistema muestra los datos y todos los campos en estado inactivo.
	8. El usuario da clic en el botón modificar.	9. El sistema activa los campos nombre, duración y estado de la Especialización, envía el cursor al campo nombre y activa el botón guardar.
	10. El usuario modifica los datos y va dando enter.	11. El sistema valida los datos.
	12. El usuario da clic en el botón guardar.	13. El sistema almacena y arroja un mensaje de transacción exitosa.
	14. Clic en salir.	

FLUJO ALTERNATIVO		
Pasos	Actor	Sistema
		15. El sistema confirma que hay inconsistencia en los datos y arroja el mensaje de alerta.

	16. El usuario ingresa los datos correctamente y da clic en guardar.	17. El sistema confirma los datos nuevamente.
		18. El sistema almacena y arroja un mensaje de transacción exitosa.
	19. Clic en salir	
Post-condiciones	Se modificó una Especialización.	
Requisito funcional	RF-070	

Fuente: Elaboración propia

Tabla 3 Escenario Inhabilitar Especialización (Sistema Proyecto de Grado)

NOMBRE	INHABILITAR ESPECIALIZACIÓN	
Descripción	Permite activar o desactivar una Especialización.	
Actor	Secretaria, Administrador	
Precondiciones	El usuario debe estar logueado en el sistema y con los permisos necesarios y haber dado clic en el botón consultar.	
FLUJO BÁSICO		
Pasos	Actor	Sistema
	1. El usuario ingresa al menú de maestros / Gestión Especialización.	2. El sistema despliega la interfaz con las opciones Código, nombre y estado de la Especialización; deshabilitados, con los botones crear, consultar y salir activos y los demás inactivos.
	3. El usuario da clic en botón consultar.	4. El sistema activa los campos código, nombre para que el usuario realice la búsqueda por cualquiera de estos campos. Activa los botones modificar, inhabilitar, cancelar.
	5. El usuario digita los valores para la búsqueda y da enter.	6. El sistema valida que los datos ingresados sean de una Especialización existente.

		7. El sistema muestra los datos y todos los campos en estado inactivo.
	8. El usuario da clic en el botón inhabilitar.	9. El sistema muestra advertencia y valida si está seguro de inhabilitar la Especialización, dando opciones si/no
	10. El usuario selecciona la opción sí.	11. El sistema cambia el estado de la Especialización y arroja un mensaje de transacción exitosa.
	12. Clic en salir.	
FLUJO ALTERNATIVO		
Pasos	Actor	Sistema
	13. El usuario selecciona la opción no.	14. El sistema muestra los datos y todos los campos en estado inactivo. El sistema inactiva los botones modificar e inhabilitar
	15. Clic en salir	
Post-condiciones	Se cambió el estado de una Especialización.	
Requisito funcional	RF-071	

Fuente: elaboración propia.

Así como demás diagramas que permitan validar la construcción del software (secuencia, componentes, clases, entre otros).

Manual del usuario: Describir la funcionalidad del sistema informático con los nuevos cambios.

Se debe construir un documento, donde se explique el paso a paso de cómo se maneja la nueva funcionalidad del sistema (Gestión Especialización y Gestión asesor, acorde a las modificaciones).

Manual de implementación: Define las actividades con su orden y secuencia que se debe desarrollar para la correcta implementación del sistema informático. En el orden se incluirá las actividades y prerrequisitos que permitan garantizar la correcta operatividad de sistema. Estas se deberán especificar, una vez se tenga definido el diseño detallado del sistema configurado.

De igual forma se deberá diseñar un manual de implementación, donde debe aparecer el tipo de hardware que se necesita, conectividad, entre otras especificaciones para su correcta instalación.

3.4.3 PROYECCIÓN DEL TIEMPO ESTIMADO PARA LA REALIZACIÓN DE LA GESTIÓN DE CONFIGURACIÓN (PROYECTO DE GRADO)

TIEMPO PRESUPUESTADO	
Fecha: 27-01-16 **Hora: 03:00pm** **Responsable**: Juan Camilo Ríos **Cargo**: Líder de Proyectos	
Actividad	**Tiempo (horas, días, semanas…)**
Evaluación	10 horas
Diseño	10 horas
Construcción	30 horas
Pruebas	4 horas
Documentación	10 horas
Tiempo total	**64 horas**

3.4.4 EVALUACIÓN DEL ESTUDIO DE LA ORDEN DE CAMBIO

EVALUACIÓN DEL ESTUDIO DE LA ORDEN DE CAMBIO			
Fecha: 29-01-16 **Hora: 08:00am** **Responsable**: Mario Enrique Cano **Cargo**: Auditor en Ingeniería de Software Nota: la ACC (Autoridad de Control de Cambios evalúa (impacto del cambio en software, hardware, rendimiento, calidad, fiabilidad, aprueba los ajustes definidos, si están dentro del producto y cumplen con la lista de chequeo)			
Nro.	Ítems	Estado	
		Si	No

1	Se tiene identificada la arquitectura en la que estará soportado el sistema: Servidor de base de datos, estaciones de trabajo.	X	
2	Se tiene las características de hardware donde se va a ejecutar el sistema, lógica del negocio, servicios de almacenamiento.	X	
3	Se describen los requerimientos de software sobre los cuales se construirá el sistema: Sistemas operativos, bases de datos y herramientas de desarrollo.	X	
4	Se tienen los diagramas de clases, de colaboración, el modelo de datos, las interfaces, consultas, informes y archivos requeridos para cada servicio a desarrollar.	X	
5	Se describe las características que han sido identificadas durante la etapa de diseño.	X	
6	Se elaboró cronograma de trabajo.	X	
7	Se tiene documentado el proceso de diseño.	X	
8	Se tiene identificado los usuarios del sistema.	X	
9	Se tienen identificado los procesos.	X	
10	Se identificaron las relaciones entre los usuarios y los procesos	X	
11	Se describen las entradas del sistema	X	
12	Se describen las salidas del sistema	X	
13	Se identificaron las interfaces para las pantallas	X	
14	Se identificaron las interfaces para las consultas		X
15	Se identificaron las interfaces para los informes	X	
16	Se identificaron las interfaces para los archivos	X	
17	Se identificaron los posibles errores y excepciones.	X	
18	Se tienen documentos que soporten el proceso de análisis	X	

3.4.5 ORDEN DE CAMBIO DE INGENIERÍA (OCI) (PROYECTO DE GRADO)

ORDEN DE CAMBIO DE INGENIERÍA (OCI)	
Fecha:02-02-16	**Hora:09:00am**
Responsable: Juan Camilo Ríos	**Cargo**: Líder de proyectos

Nota: Se valoran los tiempos presupuestados, se asignan los roles por actividad, se genera el contrato respectivo el cual debe ser firmado por las dos partes donde se consigna la fecha de inicio, y la de entregas parciales (si son acordadas), la fecha de entrega definitiva. De igual forma se define el tiempo de garantía del cambio y el cronograma.			
Actividad	**Responsable**	**Tiempo (estimado, real en horas, días, semanas…)**	
		TE	TR
Evaluación	Carlos Mario Marín	10 horas	
Diseño	Carlos Mario Marín	10 horas	
Construcción	Carlos Mario Marín	30 horas	
Pruebas	Mario Enrique Cano	4 horas	
Documentación	Sandra Elena Osorio	10 horas	
	Número de horas	64	

Nota: Después de generarse el cambio, se debe chequear el desarrollo de la OCI

Aplicación de RTF (Revisión Técnica Formal)

3.4.6 INFORME GERENCIAL SOBRE EL CAMBIO DE GESTIÓN DE CAMBIOS (PROYECTO DE GRADO)

INFORME GERENCIAL SOBRE EL CAMBIO DE GESTIÓN DE CAMBIOS			
Fecha: 19-02-16 **Hora: 03:00pm** **Responsable**: Juan Camilo Ríos **Cargo**: Líder de Proyectos			
Actividad	**Responsable**	**Fecha**	**Otros cambio**
¿Qué paso?	¿Quién lo hizo?	¿Cuándo paso?	¿Qué más se vio afectado?
Se realizó evaluación, sobre la viabilidad para incorporar dentro del software la gestión Especialización, se realizó	Juan Camilo Ríos, Líder de Proyecto Carlos Mario Marín, Analista	25-01-16	

la especificación del requerimiento.			
Se realizó estudio técnico sobre diseñó la interfaz gráfica (vistas, informes, base de datos), apoyada en la especificación del requisito	Juan Camilo Ríos, Líder de Proyecto Carlos Mario Marín, Analista	21-01-16	
Se programó las actividades y tiempo estimado para desarrollarlas	Juan Camilo Ríos, Líder de Proyecto	27-01-16	
Se realizó evaluación del estudio de la orden de cambio.	Mario Enrique Cano, Auditor de Ingeniería de Software	29-01-16	
Se aprueba orden de cambio de Ingeniería		02-02-16	
Se realizó la programación del requisito (Gestión especialización, actualización de gestión asesor, informe sobre asesor con sus especializaciones, configuración menú, perfiles, empaquetamiento)	Carlos Mario Marín	08-02-18	Actualización base de datos, actualización de aplicación, sin efectos negativos hacia otros componentes.
Se aplicaron pruebas unitarias y de integración	Mario Enrique Cano	10-02-16	

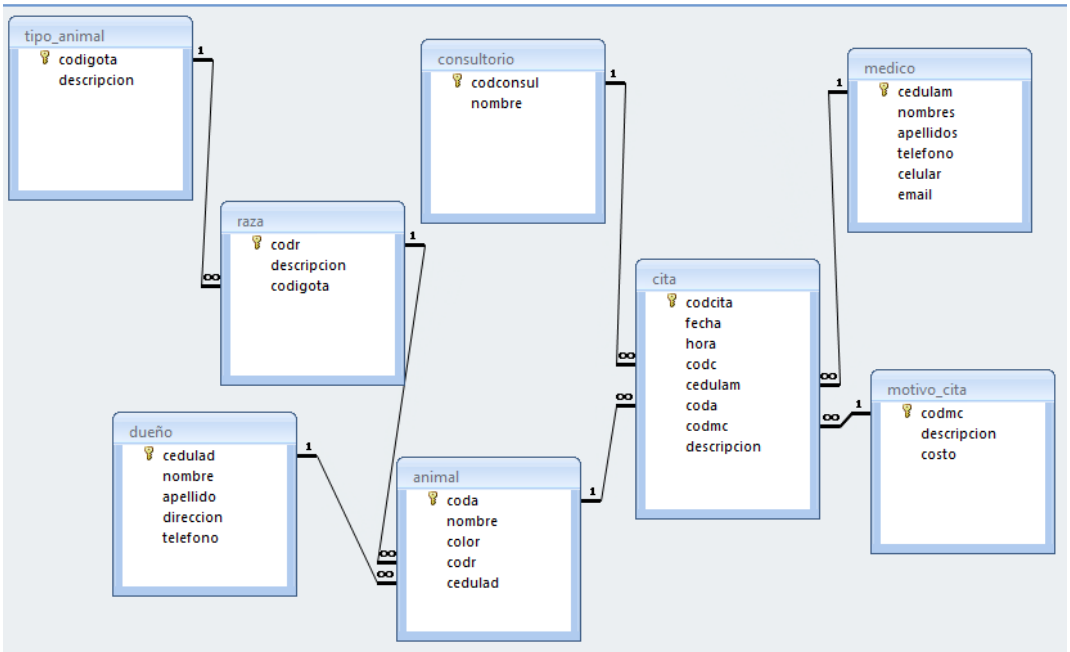
para asegurar la calidad de la gestión del cambio.			
Se documentó a través del manual técnico, manual técnico.	Sandra Elena Osorio	10-02-16	
Se implementó en la máquina del cliente	Mario Enrique Cano	15-02-16	
Observaciones: No se presentaron novedades.			

3.4.7 TALLER DE ENTRENAMIENTO UNIDAD 2

Nombre del taller: Taller 2	Modalidad de trabajo: Aprendizaje basado en problemas.
Actividad previa: Lectura de la unidad 1 y Unidad 2 del módulo	
Describa la actividad: **Situación Problemática:** Se tiene un software para manejar la información de sobre el control de citas de un centro veterinario, el cual atiende varios tipos de animales, los cuales pueden ser de varias razas, cuando el animal ingresa la secretaria pide los datos del animal, los del dueño y le asigna un médico veterinario, así como el consultorio, donde le toca la consulta, el animal puede ingresar a la consulta por varios motivos, lesiones, chequeo general, entre otros. El médico puede revisar su agenda de citas vía Web y observar la historia de cada animal. El Director del centro en cualquier momento puede consultar el movimiento de citas en determinado rango de tiempo. Para la información de los médicos el sistema permite sólo el manejo que muestra el siguiente formulario, ya que cuando se levantó los requerimientos no se consideró relevante los estudios de los médicos.	

La Dirección, requiere tener a la mano la información sobre los estudios de cada uno de los médicos, con el fin de ir registrándolos o de realizar la respectiva consulta e informe cuando lo considere pertinente.

El modelo de la base de datos actual es el siguiente:



Realizar La Gestión de Configuración del Software (GCS), para obtener el resultado que el cliente espera, así:

Se pide:

Aplicar GCS (gestión de configuración del software) al anterior caso utilizando los formatos que se muestran a continuación (agregar esquemas, gráficas, diseño al estudio con el fin de entender mejor el proceso)

3. Valor 0,5

Estudio orden de solicitud del cambio

Fecha:_____**Hora:**_____

Responsable:_____**Cargo**:_____

Requerimiento (Situación actual, justificación, descripción
Situación actual: Estado del proyecto o proceso actual
Justificación: Detallar la importancia que tiene el cambio en el mejoramiento del proyecto o proceso.
Descripción: Detallar la necesidad que refleja el cliente
1. Valor 1.5
Evaluación y análisis
Análisis de datos: (Agregación de tablas, modificación de las ya existentes, relaciones, manejo de datos existentes, migración de datos, manejo de estructuras por fuera de la base de datos entre otros). Se debe

mostrar modelos actuales y modelo propuesto, donde se resalte las mejoras y los procesos a tener en cuenta.
Análisis de Aplicativo y Ejecutables: Se detalla la actualización en formularios, consultas, informes, menú, módulos, componentes, sistemas operativos u otros sistemas que intervengan en la aplicación, cambios en ejecutables.
Documentación **Manual Técnico:** (como nuevos modelos, script de migración, y población de datos, script de gereración de ejecutables, configuraciones a nivel de parametrización a nivel de sistemas operativos, hardware, base de datos entre otros.) **Manual del usuario:** Describir la funcionalidad del sistema informático con los nuevos cambios. **Manual de implementación**: Define las actividades con su orden y secuencia que se debe desarrollar para la correcta implementación del sistema informático. En el orden se incluirá las actividades y prerrequisitos que permitan garantizar la correcta operatividad de sistema. Estas se deberán especificar, una vez se tenga definido el diseño detallado del sistema configurado.
2. Valor 0,5
Tiempo Presupuestado

Actividad	**Tiempo (realizarlo en horas…)**
Evaluación	
Diseño	
Construcción	
Pruebas	
Documentación	
Tiempo total	

4. **Valor 0,5**

Evaluación del estudio de la orden de cambio

Fecha:_____**Hora:**_____

Responsable:_____**Cargo**:_____

Nota: la ACC (Autoridad de Control de Cambios evalúa (impacto del cambio en software, hardware, rendimiento, calidad, fiabilidad, aprueba los ajustes definidos, si están dentro del producto y cumplen con la lista de chequeo)

NRO	Ítems	ESTADO (SI, NO)

1	Se tiene identificada la arquitectura en la que estará soportado el sistema: Servidor de base de datos, estaciones de trabajo.		
2	Se tiene las características de hardware donde se va a ejecutar el sistema, lógica del negocio, servicios de almacenamiento.		
3	Se describen los requerimientos de software sobre los cuales se construirá el sistema: Sistemas operativos, bases de datos y herramientas de desarrollo.		
4	Se tienen los diagramas de clases, de colaboración, el medelo de datos, las interfaces, consultas, informes y archivos requeridos para cada servicio a desarrollar.		
5	Se describe las características que han sido identificadas durante la etapa de diseño.		
6	Se elaboró cronograma de trabajo.		
7	Se tiene documentado el proceso de diseño.		
8	Se tiene identificado los usuarios del sistema.		
9	Se tienen identificado los procesos.		
10	Se identificaron las relaciones entre los usuarios y los procesos		
11	Se describen las entradas del sistema		
12	Se describen las salidas del sistema		
13	Se identificaron las interfaces para las pantallas		
14	Se identificaron las interfaces para las consultas		
15	Se identificaron las interfaces para los informes		
16	Se identificaron las interfaces para los archivos		
17	Se identificaron los posibles errores y excepciones.		
18	Se tienen documentos que soporten el proceso de análisis		

5. Valor 0,5

Orden de cambio de ingeniería (OCI)

Fecha:_____**Hora:**_____

Responsable:_____**Cargo**:_____

Nota: Se valora los tiempos presupuestados, se asignan los roles por actividad, se genera el contrato respectivo el cual debe ser firmado por las dos partes donde se consigna la fecha de inicio, y la de entregas parciales (si son acordadas), la fecha de entrega definitiva. De igual forma se define el tiempo de garantía del cambio y el cronograma.

Actividad	**Responsable**	**Tiempo (estimado, real en horas, dias, semanas…)**	
		TE	TR

Nota: Despues de generarse el cambio, se debe chequear el desarrollo de la OCI

Aplicación de RTF (Revisión Técnica Formal)

6. Valor 1.0

Informe gerencial sobre el cambio de gestión de cambios

Fecha:_____**Hora:**_____

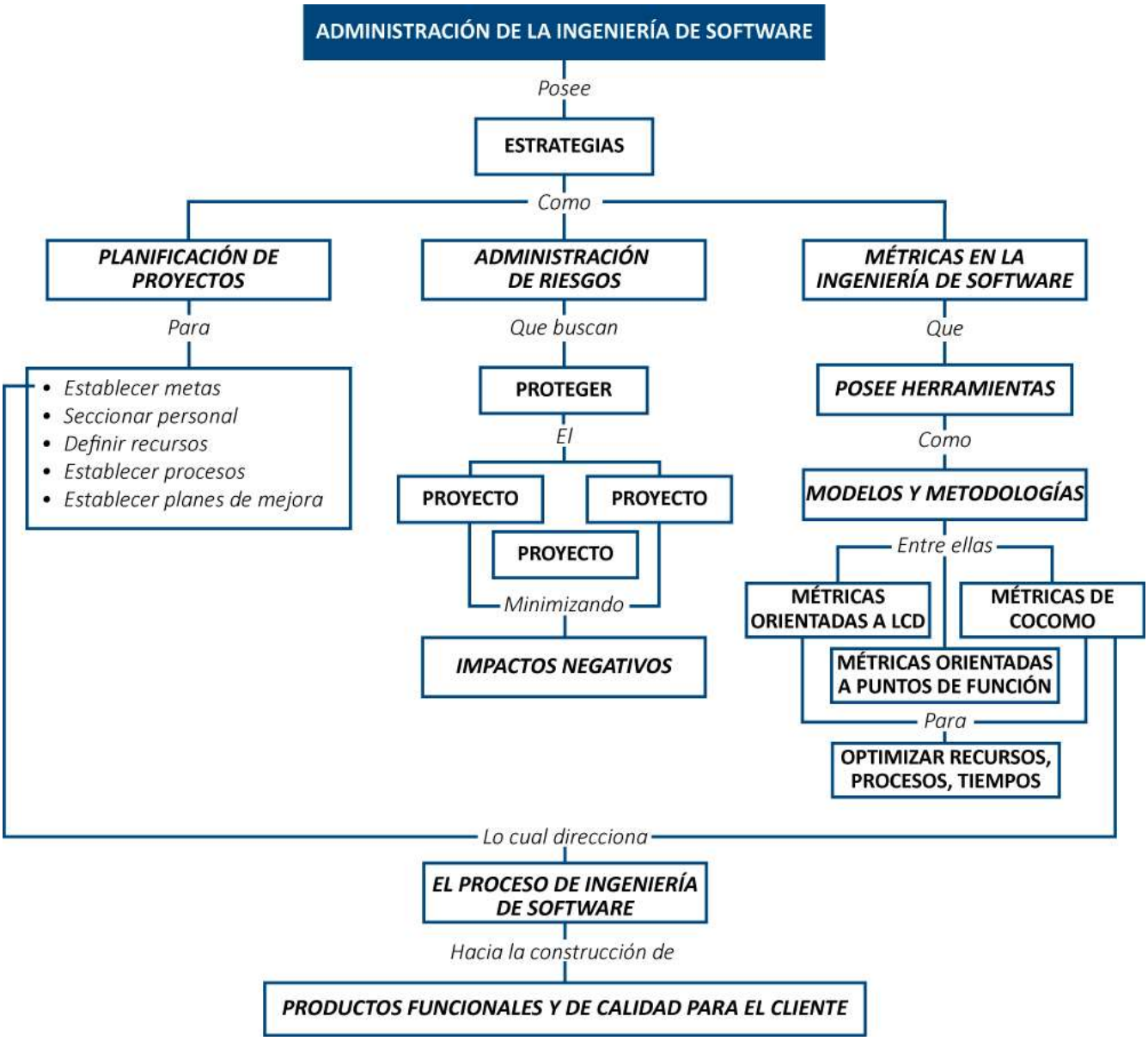
Responsable:_____**Cargo**:_____

Actividad	**Responsable**	**Fecha**	**Otros cambio**
Qué paso?	Quién lo hizo?	Cuándo paso?	Qué mas se vió afectado?

1. **Valor 0,5**

Cómo configuraría el cambio de versión en caso de ser necesaria?

4 UNIDAD 3: ADMINISTRACIÓN DE LA INGENIERÍA DE SOFTWARE



Fuente: elaboración propia.

4.1 TEMA 1: PLANIFICACIÓN DE PROYECTOS DE SOFTWARE

Los proyectos de software al igual que cualquier otro proyecto se debe planificar teniendo en cuenta los recursos, tiempo, estimación, objetivos y metas que integradas se orienten hacia la construcción de dicho producto de

software. Según Pressman, la gestión de proyectos de software se orienta hacia las cuatro P (personal, producto, proceso y proyecto), aplicando actividades de medición que ayudan a la toma de decisiones con el fin de estimar el proyecto y los procesos, aplicar controles de calidad, evaluación a la productividad y control al proyecto. Ver Video Gestión de Proyectos Informáticos en:

3UTPL GESTIÓN DE PROYECTOS [(INFORMÁTICA)(GESTIÓN DE PROYECTOS INFORMÁTICOS)] *Enlace*

Una buena planificación y gestión de proyectos permite detectar a tiempo procesos incorrectos e identificar riesgos que puedan ser administrados a tiempo, minimizando así eventos negativos que puedan afectar el proyecto, por lo tanto, es importante prever dichos acontecimientos a través de procesos sistemáticos como la planificación, identificación, análisis, soluciones y control de los posibles riesgos, presentando un plan de continuidad que permitan dar solución a dichos imprevistos. Ver video Gestión de proyectos y riesgos

4Entrevista con Liliana Buchtik, especialista en gestión de proyectos y riesgos. *Enlace*

4.2 TEMA 2: ADMINISTRACIÓN DEL RIESGO EN LOS PROYECTOS DE SOFTWARE

4.2.1 CONCEPCIONES BÁSICAS

En relación al riesgo, según el diccionario ABC, éste se define como "amenaza concreta de daño que yace sobre nosotros en cada momento y segundos de nuestras vidas, pero que puede materializarse en algún momento o no".

En el caso de Charette (1989), hace alusión a que el riesgo "en primer lugar, afecta a los futuros acontecimientos. El hoy y el ayer están más allá de lo que nos pueda preocupar, pues ya estamos cosechando lo que sembramos previamente con nuestras acciones del pasado. La pregunta es, podemos por tanto, cambiando nuestras acciones actuales, crear una oportunidad para una situación diferente y, con suerte, mejor para nosotros en el futuro. Esto significa, en segundo lugar, que el riesgo implica cambio, que puede venir dado por cambios de opinión, de acciones, de lugares. En tercer lugar, el riesgo implica elección y la incertidumbre que entraña la elección. Por tanto, el riesgo, como la muerte, es una de las pocas cosas inevitables de la vida.

Cuando se considera el riesgo en el contexto de la Ingeniería de Software, los tres pilares conceptuales de Charette se hacen continuamente evidentes. El futuro es lo que nos preocupa, ¿qué riesgos podrían hacer que nuestro proyecto fracasara? El cambio es nuestra preocupación ¿cómo afectarán los cambios en los requisitos del cliente, en las tecnologías de desarrollo, en los ordenadores a las que van dirigidas, el proyecto y todas las entidades relacionadas con él, al cumplimiento de la planificación temporal y al éxito en general? Para terminar, nos enfrentamos con elecciones ¿qué métodos y herramientas deberíamos emplear, cuánta gente debería estar implicada, qué importancia hay que darle a la calidad?

El riesgo se debe preverse, desde antes que ocurra, lo cual minimiza el impacto en caso de que se genere, lo ideal es no esperar hasta que suceda, donde en dicho caso, se deberá buscar mecanismos para corregir el problema existente, lo cual puede generar sobre costos en recursos, ineficiencia en tiempos, siendo costoso reparar las repercusiones que puedan acaecer ya que es posible que todo el sistema se vea afectado. Los riesgos están presentes en todos los ambientes, desde el familiar, personal social, cultural, político,

económico, laboral, empresarial, etc, donde en el caso del software, el riesgo se puede presentar desde la planeación del proyecto, proceso de construcción, proceso de utilización, hasta que éste sale del mercado.

Es así como un software mal gestionado puede tener consecuencias nefastas como por ejemplo, un error en la programación del piloto automático de un avión puede generar muchas pérdidas de vidas, un defecto en el software que maneja la compra y venta de acciones en la bolsa, puede generar pérdidas millonarias en las empresas, un mal cálculo en el costo de producción de un software puede generar incumplimientos, demandas y hasta la quiebra de la empresa desarrolladora de software, ya que dicho producto informático es el soporte de cualquier proceso.

Por lo tanto, el riesgo en el software, deberá ser analizado, antes de iniciar cualquier proyecto, con el fin de minimizar impactos. El riesgo debe analizarse, con el fin de implementar acciones de forma proactiva (antes de que pase el suceso) para que no suceda, pero también de forma correctiva o reactiva (después de que pasa el suceso), en caso de que suceda, evitando a la máxima, pérdida de recursos, pero especialmente la vida.

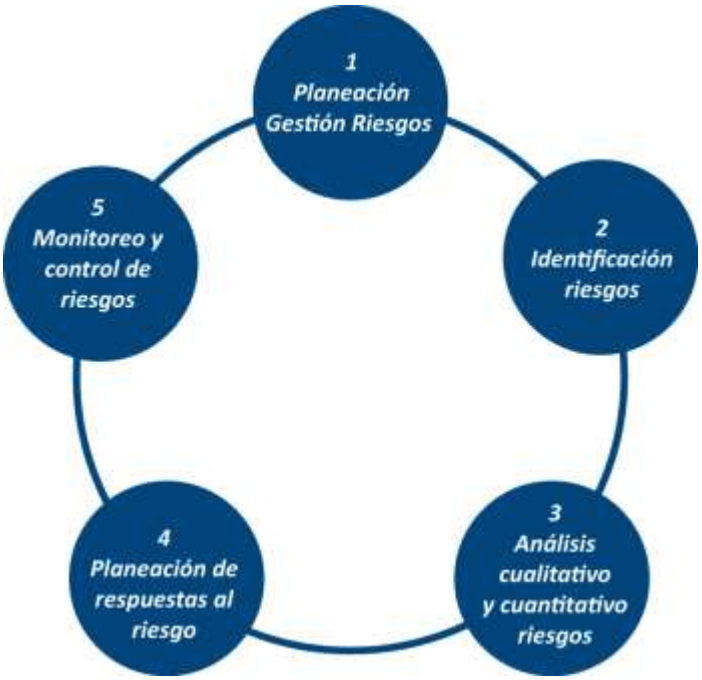
Ejemplo de riesgo proactivo (revisión de una máquina de confección cada dos meses con el fin de verificar su estado), en el caso del riesgo correctivo o reactivo, (éste se da cuando la máquina falla, donde es posible que muchas prendas quedaron defectuosas o se suspende todo el proceso por horas o días, ya que no se consideró con anticipación que dicho riesgo podía generar pérdidas económicas para la empresa). Por lo que se recomienda realizar un estudio completo sobre los posibles riesgos que pueden poner en peligro los procesos, los proyectos, la calidad de los productos de software, así como el planteamiento de estrategias que busquen minimizarlos. Ver video Riesgo al Generar Productos de Software

5Riesgos al generar productos de software - Entrevista a Liliana Buchtik Enlace

4.2.2 GESTIÓN DE RIESGOS EN LOS PROYECTOS

Se refiere a una serie de etapas, que se tienen en cuenta para que los proyectos cumplan con sus metas, consumiendo los recursos asignados y en los tiempos proyectados. Ver imagen

Imagen 54 Procesos para la gestión de riesgos en los proyectos



Fuente: elaboración propia.

A continuación, se hace alusión a cada uno de los procesos utilizados para gestionar los riesgos en los proyectos informáticos:

Planeación de la Gestión del Riesgo: Es el proceso por el cual se define como realizar las actividades de gestión de riesgos para un proyecto (en éste caso de software), asegurando el nivel, el tipo y la visibilidad del riesgo, teniendo en cuenta los recursos y el tiempo suficiente para las actividades de gestión de riesgos y para establecer una base adecuada para la evaluación. Éste proceso se debe iniciar al concebirse el proyecto, antes de ponerlo en ejecución.

Un plan de riesgo podría tener los siguientes elementos:

- Metodología: Se define como se realizará el proceso de gestión de riesgos para el proyecto.
- Roles y responsabilidades: Funciones que representa cada integrante del equipo de trabajo.
- Presupuesto: Cuanto cuesta la realización del proceso de gestión de riesgos.
- Tiempo: En cuantas horas se debe realizar el proceso de gestión de riesgos.
- Categorías: Lista posibles causas de riesgos que pueden afectar el proyecto.
- Definiciones de probabilidad e impacto: Escalas definidas para evaluar la posibilidad de ocurrencia y el efecto sobre los objetivos del proyecto.
- Niveles de tolerancia: Miden el nivel de aceptación de los riesgos por parte de los interesados.
- Formatos de reportes: Se define los elementos que se tendrán en cuenta para la elaboración de los reportes de riesgos.
- Seguimiento: Se definen los procesos de auditoría, monitoreo y retroalimentación.

Identificación del Riesgo: Se determinan los riesgos que pueden afectar positiva o negativamente al proyecto, sacando un listado de amenazas y oportunidades para el proyecto

El proceso para identificar los riesgos, podría tener los siguientes pasos:

- Determinar el equipo de trabajo para la identificación de los riesgos.
- Recolectar información histórica de proyectos similares.
- Sacar un listado de los riesgos.
- Documentar todos los riesgos identificados.
- Recolectar información adicional que ayuden completar la información sobre el riesgo.

Análisis cualitativo y cuantitativo de los riesgos

- Cualitativo: Es el proceso que consiste en priorizar los riesgos identificados, determinando la posibilidad de ocurrencia y el impacto sobre los objetivos del proyecto. Éste análisis es subjetivo y su certeza depende muchas veces de la experiencia del personal en asuntos de riesgos.
- Cuantitativo: Proceso que consiste en analizar numéricamente el efecto de los riesgos identificados sobre los objetivos del proyecto. El análisis cuantitativo, no siempre es requerido en el proceso de gestión de riesgos.
    - La forma como se analizan los riesgos, tanto de forma cuantitativa como cualitativa, se profundiza en la materia Calidad de Software.

Planeación de Respuestas: Consiste en determinar las acciones de respuesta efectiva que son apropiadas para minimizar o eliminar el riesgo.

Pasos recomendados:

- Seleccionar la(s) estrategia(s) de respuesta al riesgo.
- Por cada acción registrar: fecha de inicio, fecha final, responsable de cada acción del riesgo, estado del riesgo.
- Registrar el plan de contingencia y el responsable de ejecutarlo.
- Registrar el disparador de cada riesgo.
- Determinar las reservas de contingencia.
- Confirmar el responsable del riesgo.
- Comunicar los riesgos a los responsables de los riesgos.
- Comunicar las acciones a los responsables de las acciones de los riesgos.
- Comunicar a los interesados principales las actualizaciones al plan de dirección del proyecto.

Monitoreo y control de riesgos: se realiza seguimiento a los riesgos identificados, se monitorean los riesgos, se identifican nuevos riesgos, y se evalúa la efectividad del proceso de gestión de riesgos, informándolos y registrando las nuevas lecciones aprendidas.

4.2.3 TIPOS DE RIESGOS EN LOS PROYECTOS PARA LA CONSTRUCCIÓN DE SOFTWARE

Existe gran variedad de riesgos inmersos dentro de todo el proceso de Ingeniería de Software, en relación al producto a construir, al negocio para el cual se elabora el producto, con el cliente, relacionados con el proceso

de construcción del software, con los aspectos técnicos para el desarrollo, con las tecnologías a utilizar y requeridas, con el entorno de desarrollo, entre otros contexto y procesos dentro y fuera del ciclo de vida del software, para lo que Menéndez, Barzana (2005), realizan un aporte importante al ofrecer una serie de listas de chequeo que permiten, realizar validaciones orientadas a la detección de riesgos, con el fin de que sean clasificados para minimizar impactos en los proyectos de software. Las listas en mención se presentan a continuación, organizadas por temáticas específicas.

Riesgos del tamaño del producto:

- ¿Tamaño estimado del producto en LDC o FP?
- ¿Grado de seguridad en la estimación del tamaño?
- ¿Tamaño estimado del producto en número de programas, archivos y transacciones?
- ¿Porcentaje de desviación en el tamaño del producto respecto a la medida de productos anteriores?
- ¿Tamaño de la base de datos creada o empleada por el producto?
- ¿Número de usuarios del producto?
- ¿Número de cambios previstos a los requisitos del producto? ¿Antes de la entrega? ¿Después de la entrega?

Riesgos de impacto en el negocio:

- ¿Efecto del producto en los ingresos de la compañía?
- ¿Viabilidad del producto para los gestores expertos?
- ¿Es razonable la fecha límite de entrega?
- ¿Número de clientes que usarán el producto y la consistencia de sus necesidades relativas al producto?
- ¿Número de otros productos/sistemas con los que el producto debe tener interoperatividad?
- ¿Sofisticación del usuario final?
- ¿Cantidad y calidad de la documentación del producto que debe ser elaborada y entregada al cliente?
- ¿Limitaciones gubernamentales en la construcción del producto?
- ¿Costos asociados por un retraso en la entrega?
- ¿Costos asociados con un producto defectuoso?

Riesgos relacionados con el cliente:

- ¿Ha trabajado con el cliente anteriormente?
- ¿Tiene el cliente una idea formal de lo que se requiere?
- ¿Está dispuesto el cliente a establecer una comunicación fluida con el desarrollador?
- ¿Está dispuesto el cliente a participar en las revisiones?
- ¿Es sofisticado técnicamente el área del producto?
- ¿Está dispuesto el cliente a dejar a su personal hacer el trabajo?
- ¿Entiende el cliente el proceso del software?

Riesgos del proceso de software:

- ¿Posee la empresa de desarrollo procesos estándar para el desarrollo del software?
- ¿El equipo de trabajo está dispuesto a usar procesos estandarizados para el desarrollo de software?
- ¿El equipo de desarrollo cuenta con las suficientes competencias para realizar el trabajo?
- ¿Se ha proporcionado una copia de los estándares de Ingeniería de Software publicados a cada desarrollador y gestor de software?
- ¿Se han desarrollado diseños de documentos y ejemplos para todas las entregas definidas como parte del proceso del software?
- ¿Se llevan a cabo regularmente revisiones técnicas formales de las especificaciones de requisitos, diseño y código?
- ¿Se llevan a cabo regularmente: revisiones técnicas de los procedimientos de prueba y de los casos de prueba?
- ¿Se documentan todos los resultados de las revisiones técnicas, incluyendo los errores encontrados y recursos empleados?
- ¿Existe algún mecanismo para asegurarse de que el trabajo realizado en un proyecto se ajusta a los estándares de Ingeniería de Software?
- ¿Se emplea una gestión de configuración para mantener la consistencia entre los requisitos del sistema/software, diseño, código y casos de prueba?
- ¿Hay algún mecanismo de control de cambios de los requisitos del cliente que impacten en el software?
- ¿Está debidamente elaborado en contrato con el cliente en relación a funcionalidad del producto, costos y tiempos de entrega?

- ¿Se sigue algún procedimiento para hacer un seguimiento y revisar el proceso de construcción del software?

Aspectos técnicos:

- ¿Se emplean técnicas de especificación de aplicaciones para ayudar en la comunicación entre el cliente y el desarrollador?
- ¿Se emplean métodos específicos para el análisis del software?
- ¿Emplea un método específico para el diseño de datos y arquitectónico?
- ¿Está escrito su código en más de un 90 por ciento en lenguaje de alto nivel?
- ¿Se han definido y empleado reglas específicas para la documentación del código?
- ¿Emplea métodos específicos para el diseño de casos de prueba?
- ¿Se emplean herramientas de software para apoyar la planificación y el seguimiento de las actividades?
- ¿Se emplean herramientas de software de gestión de configuración para controlar y seguir los cambios a lo largo de todo el proceso del software?
- ¿Se emplean herramientas de software para apoyar los procesos de análisis y diseño del software?
- ¿Se emplean herramientas para crear prototipos software?
- ¿Se emplean herramientas de software para dar soporte a los procesos de prueba?
- ¿Se emplean herramientas de software para soportar la producción y gestión de la documentación?
- ¿Se han establecido métricas de calidad para todos los proyectos de software?
- ¿Se han establecido métricas de productividad para todos los proyectos de software?

Riesgos Tecnológicos:

- ¿Es nueva para su organización la tecnología a construir?
- ¿Demandan los requisitos del cliente la creación de nuevos algoritmos o tecnología de entrada o salida?
- ¿El software interactúa con hardware nuevo o no probado?
- ¿Interactúa el software a construir con productos software suministrados por el vendedor que no se hayan probado?

- ¿Interactúa el software a construir con un sistema de base de datos cuyo funcionamiento y rendimiento no se han comprobado en esta área de aplicación?
- ¿Demandan los requisitos del producto una interfaz de usuario especial?
- ¿Demandan los requisitos del producto la creación de componentes de programación distintos de; los que su organización haya desarrollado hasta ahora?
- ¿Demandan los requisitos el empleo de nuevos métodos de análisis, diseño o pruebas?
- ¿Demandan los requisitos el empleo de métodos de 'desarrollo del software no convencionales, tales como los métodos formales, enfoques basados en IA y redes neuronales?
- ¿Imponen excesivas restricciones de rendimiento los requisitos del producto?
- ¿No está seguro el cliente de que la funcionalidad pedida sea factible?

Riesgos en el entorno de desarrollo:

- ¿Se tiene disponible una herramienta de gestión del proceso del software?
- ¿Existen herramientas de análisis y diseño disponibles?
- ¿Proporcionan las herramientas de análisis y diseño, métodos apropiados para el producto a construir?
- ¿Hay disponibles compiladores o generadores de código apropiados para el producto a construir?
- ¿Hay disponibles herramientas de pruebas apropiadas para el producto a construir?
- ¿Se tiene disponible herramientas de gestión de configuración software?
- ¿Hace uso el entorno de bases de datos o información almacenada?
- ¿Están todas las herramientas de software integradas entre sí?
- ¿Se ha formado a los miembros del equipo del proyecto en todas las herramientas?
- ¿Existen expertos disponibles para responder todas las preguntas que surjan sobre las herramientas?
- ¿Es adecuada la ayuda en línea y la documentación de las herramientas?
- ¿Tiene el personal todos los conocimientos adecuados?
- ¿Se tiene suficiente personal?
- ¿Se ha asignado al personal para toda la duración del proyecto?
- ¿Dispone el personal de las expectativas correctas sobre el trabajo?
- ¿Ha recibido el personal la formación adecuada?

Peter Drucker dijo una vez: "Mientras que es inútil intentar eliminar el riesgo y cuestionable el poder minimizarlo, es esencial que los riesgos que se tomen sean los riesgos adecuados"

Con lo anterior se puede concluir, en relación a que el riesgo se puede minimizar, pero no se puede eliminar completamente, ya que todo es cambiante y existen tanto variables internas como externas que permean todos los procesos con la probabilidad de alterar los resultados de los proyectos, ya sea de forma positiva o desviando metas puntuales.

Observar la película sobre Steve Jobs (creador de Apple) en el link (https://www.youtube.com/watch?v=-gxMnH2qzgU#t=6392.013577). Redactar un ensayo sobre la gestión de proyectos, en una hoja tamaño carta a mano.

4.3 TEMA 3: MEDICIÓN EN LOS PROCESOS DE INGENIERÍA DE SOFTWARE

Durante el proceso de construcción de cualquier producto, se debe establecer previamente la forma como se debe medir, procesos, tiempos, utilización de recursos, entre otros aspectos que permiten otros procesos de retroalimentación que en definitiva buscan la optimización de todos los recursos tangibles e intangibles que se tienen en cuenta a la hora de construir un producto de calidad.

En relación al software, al igual que en otros tipos de productos, es de suma importancia la aplicación de procesos de medición que buscan como lo define el wordreference “Comparar una cantidad con su respectiva unidad, con el fin de averiguar cuántas veces la primera contiene la segunda”, “Igualar y comparar una cosa no material con otra”; en relación a la medida, el Thefreedictionary, la define como “cantidad que resulta de medir una longitud”, “Hecho a propósito con unas dimensiones determinadas”; Y de acuerdo al glosario de la IEEE, la define como “una medida del grado en el que un sistema, componente o proceso posee un atributo determinado”. Para ésta temática tan importante como es la medición, se tendrá en cuenta los lineamientos que se dan desde la ISO 25000 (2014), donde se aborda claramente el tema de métricas en el software.

Es así como las métricas son herramientas que se aplican para medir y valorar la calidad de los productos de ingeniería o los sistemas que se construyen, las cuales proporcionan una manera sistemática de valorar la

calidad basándose en un conjunto de reglas claramente definidas permitiendo descubrir y corregir problemas potenciales. Ver video Métricas de Software (https://www.youtube.com/watch?v=y68Tx1ogdu0)

4.3.1 ALGUNAS DE LAS RAZONES QUE SE TIENEN PARA MEDIR

Las razones para medir un proceso o un producto de software pueden variar acorde a las diversas necesidades, tanto de las empresas que construyen el producto como a la misma necesidad del cliente que normalmente se orienta al cubrimiento de unos requisitos funcionales y no funcionales del software. Por lo tanto se mencionan algunas razones que se tienen para medir el software como son:

- Sirve para indicar la calidad del producto
- Permite medir la productividad de cada una de las personas involucradas en la construcción del producto.
- Para evaluar los beneficios en términos de productividad y de calidad, derivados del uso de nuevos métodos y herramientas de ingeniería de software.
- Permite establecer una línea de base para la estimación, ya que se basa en la experiencia.
- Para ayudar a justificar el uso de nuevas herramientas o de formación adicional, entre otros.

4.3.2 CATEGORÍAS DE LAS MEDIDAS

Las medidas que se utilizan para el software, se agrupan en dos grandes categorías como son las medidas directas y las medidas indirectas.

Medidas Directas: Son aquellas medidas que se encuentran de forma directa, sin necesidad de ser procesadas como por ejemplo el valor de una hora de trabajo, las líneas de código producidas en determinado tiempo, la velocidad de ejecución del programa, el tamaño de memoria, el número de defectos observados en un determinado periodo de tiempo, entre otras.

Medidas Indirectas: Son medidas que se basan en las directas como por ejemplo el esfuerzo utilizado para desarrollar una actividad (número de horas por número de personas), a la funcionalidad, calidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento, etc. Dichas medidas se sacan de la combinación entre medidas directas e indirectas.

4.3.3 ALGUNOS DE LOS MÉTODOS MÁS UTILIZADOS PARA LA APLICACIÓN DE MÉTRICAS DE SOFTWARE

Algunos de los métodos más utilizados para la aplicación de métricas de software, se apoyan en las métricas orientadas en las líneas de código (LDC), Las métricas orientados a los puntos de función ajustados (PFA), métricas orientadas en los casos de uso, métricas COCOMO.

4.3.4 MÉTRICAS ORIENTADAS A LAS LÍNEAS DE CÓDIGO (LOC)

Estas métricas, permiten la creación de análisis sencillos como por ejemplo en que tiempo se va a terminar el software, cuantas personas se van a necesitar. Son medidas directas al software y el proceso por el cual se desarrolla, si una organización de software mantiene registros sencillos, se puede crear una tabla de datos orientados al tamaño como se muestra en la siguiente tabla, donde se pueden observar tres proyectos, orientados al manejo de procesos contables, cuyo promedio de datos históricos permite tener una base para la aplicación de algunas métricas como por ejemplo la productividad, calidad orientadas al tamaño del software.

Imagen 55 Ejemplo sobre métricas LOC

Nombre del proyecto	Esfuerzo	Valor Proyecto	KLDC*mil	Pág. De documentación	Errores	Nro de personas	Nro de horas	Valor hora
Conta_U	4.500	55.000.000	200	500	50	5	900	61.111
ABC_Costos	4.800	60.000.000	300	600	80	6	800	75.000
Cartera_Une	9.600	80.000.000	500	800	70	8	1.200	66.667
Promedio Para datos históricos	**6.300**	**65.000.000**	**333**	**633**	**67**	**6**	**967**	**67.593**

Fuente: elaboración propia

Se puede observar que algunas medidas son directas y otras indirectas

Mediadas directas:

- Nro de horas: equivale a un dato que es negociado con el cliente, el cual puede contratar la construcción del software por un número x de horas, siendo una medida directa.
- Número de personas: Equivale a una medida directa, ya que depende de la asignación de personas por parte del líder del proyecto.

- Kilolíneas de Código: corresponde al tamaño del producto de software, data que ofrecen los lenguajes de programación, siendo un valor que está dato por el sistema y que sirve de referencia para calcular el tamaño de proyectos similares.
- Páginas de documentación: equivale a medidas directas suministradas por el sistema.
- Errores: son resultados que arroja el sistema.

Medidas indirectas:

- Valor del proyecto: Es un dato que se puede calcular del número de horas del proyecto por el valor de la hora, siendo ambos datos medidas directas, que al hacer el cálculo arrojan como resultado un dato procesado.
- Esfuerzo: Se calcula, teniendo en cuenta el número de horas por el número de personas que realiza el proceso, donde en el anterior ejemplo no se separa por fases, sino que se toma como si las personas estuviesen encargadas de todo el proceso de construcción del software, entre otras.

Algunas fórmulas pueden ser:

- Productividad = KLDC/persona-hora (donde si tomamos el proyecto Conta_U, se puede decir que por cada persona hora, se tiene una productividad de 45 líneas de código).
- Costo por persona por hora=Valor proyecto/esfuerzo (donde si tomamos el proyecto Conta_U, se puede decir que a cada persona por hora se le paga 12.222 pesos)
- Calidad = errores/KLDC (donde si tomamos el proyecto Conta_U, por cada mil líneas de código, se encontró 1 error).
- Documentación = Pags. Doc/ KLDC (donde si tomamos el proyecto Conta_U, por cada mil líneas de código se tienen 1 páginas de documentación)
- Costo por kilo línea de código = Valor proyecto/KLDC (donde si tomamos el proyecto Conta_U, cada kilolínea de código, tiene un valor de 275 pesos aproximadamente).

En éste ejemplo se muestran algunas métricas, pero cada empresa puede establecer otras, lo importante es que se deben construir indicadores estándares de tal forma que se pueda medir de la misma manera, conservando las líneas base establecidas para ello.

Es importante hacer claridad que para aplicar métricas de software, se deben generar medidas que se alimentan de datos históricos, en relación a proyectos que posean las mismas característica o características similares, es así como se puede observar en la tabla anterior que existen tres proyectos relacionados con el área de contabilidad, donde en el caso de requerirse un proyecto de características totalmente diferentes como por ejemplo el seguimiento de los rayos solares sobre un grupo de plantas, se deberían generar otro tipo de métricas.

Ejercicio sobre métricas orientadas a las líneas de código (LDC)

- Analizar la siguiente información, teniendo en cuenta las fórmulas presentadas (LCF, CHP, HPT, CTP, LCFH, CLCF), en el taller que se visualiza más adelante.
- Pasar la información a la hoja electrónica Excel y aplicar las fórmulas
- Aplicar fórmulas para calcular los valores de las celdas que se encuentran en rojo
- Analizar la función (SI), teniendo en cuenta los rangos de productividad exigidos por la empresa XYZ.

Imagen 56 Rangos Empresa XYZ

RANGOS EMPRESA XYZ		
LÍMITE INFERIOR	**LÍMITE SUPERIOR**	**PRODUCTIVIDAD**
0	39%	Muy baja
40%	59%	Baja
60%	79%	Media
80%	99%	Alta
100%	…	Muy alta

Fuente: elaboración propia.

- Qué análisis merece el costo por línea que arroja el proyecto E
- Realizar el análisis en caso de HPT(Horas total del proyecto sea 10 y 60

Imagen 57 Taller Métricas Orientadas al Tamaño

	A	B	C	D	E	F	G
1	**Taller Métricas orientadas al tamaño**						
2	La empresa XYZ productora de software, posee los siguientes datos históricos, de proyectos						
3	realizados con características similares al actual.						
4							
5	LCF: Cantidad de línea código Fuente del proyecto				Directas		
6	CHP: Valor de la hora pagada al programador				Directas		
7	HPT: Horas del Proyecto Totales				Indirectas	Suma de HPD	
8	CTP: Costo Total del Proyecto				Indirectas	(CHP*HPT)	
9	LCFH: Líneas de Código Fuente por hora				Indirectas	(LCF/HPT)	
10	CLCF: Costo por línea de código Fuente.				Indirectas	(CTP/LCF)	
11	**Nom. Proy**	**LCF**	**CHP**	**HPT**	**CTP**	**LCFH**	**CLCF**
12	A	3.000,00	10.000,00	30	300.000,00	100,00	100,00
13	B	5.000,00	10.000,00	35	350.000,00	142,86	70,00
14	C	2.500,00	11.000,00	32	352.000,00	78,13	140,80
15	D	4.000,00	12.000,00	40	480.000,00	100,00	120,00
16	**Valor medio**	**3.625,00**	**10.750,00**	**34**	**370.500,00**	**105,25**	**102,21**
17							
18	Hallar la productividad del empleado Juan, sabiendo que el proyecto que tiene a cargo						
19	posee LCF(4000), CHP(12500), HPT(30), calcular CTP, LCFH, CLCF)						
20	Ubicarlo en el rango que le corresponda entre muy bajo, bajo, normal, alto, muy alto.						
21	Nota: el proyecto posee características similares al anterior						
22							
23	**Nom. Proy**	**LCF**	**CHP**	**HPT**	**CTP**	**LCFH**	**CLCF**
24	E	4.000,00	12.500,00	30	375.000,00	133,33	93,75
25	**Productividad de Juan**	**Muy alta**					
26	SI(F24/F16<0,4;"Muy baja";SI(F24/F16<0,6;"Baja";SI(F24/F16<0,8;"Normal";SI(F24/F16<1;"Alta";"Muy alta"))))						
27							
28	Análisis: El rendimiento de Juan con respecto a la cantidad de línea de código es Muy alto, ya que						
29	la cantidad de líneas del proyecto es superior al promedio y la cantidad de horas utilizadas						
30	es inferior al promedio establecido.						
31	Por lo tanto el costo por línea de código es inferior al promedio de los proyectos de referencia						

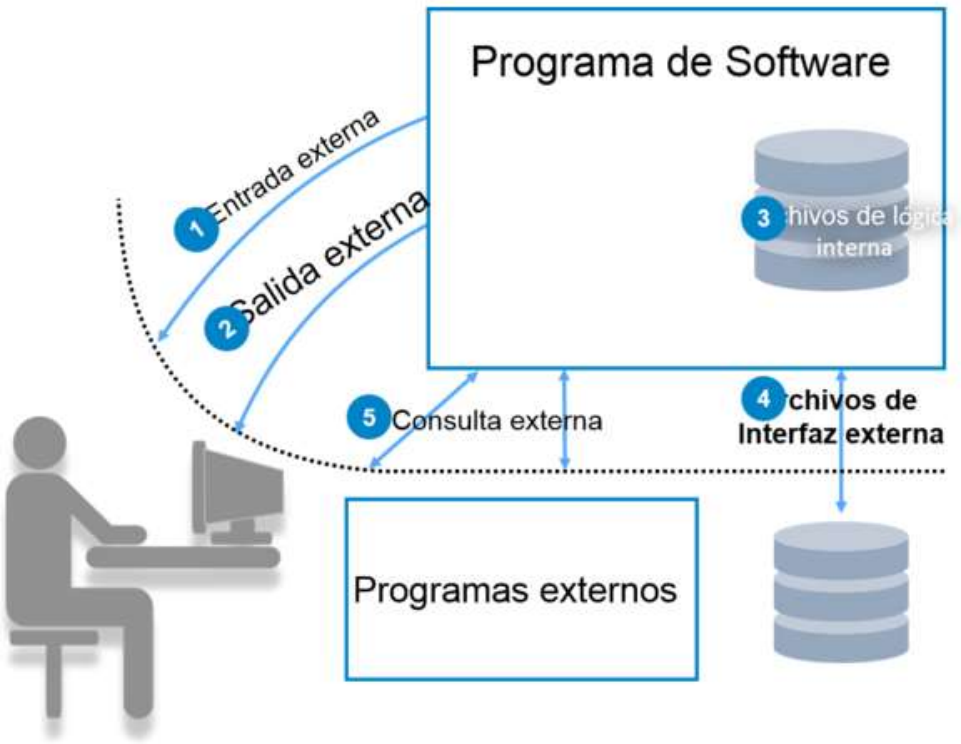
Fuente: elaboración propia.

4.3.5 MÉTRICAS ORIENTADAS A LOS PUNTOS DE FUNCIÓN

Es un método propuesto por Allan Albrecht (1979) utilizado para realizar medición, dejando de lado lo relacionado con la codificación, se centra especialmente en el tamaño del proyecto, el costo de producción, tiempo de desarrollo, productividad de las personas involucradas con cada uno de los procesos de construcción, entre otros. Dicho método es utilizado para medir el tamaño del software, centrándose en la funcionalidad del sistema, donde las características y funcionalidades representadas en Entradas externas, Salidas externas, consultas externas, archivos de datos, interfaz externa, permiten medir la complejidad del sistema.

Por lo tanto, un punto de función se puede definir como una función comercial que puede ejecutar un usuario final, donde un software que tenga "X" puntos de función, entrega "X" funciones al usuario final, convirtiéndose los puntos de función en uno de los métodos más usados en la medición del software. Ver esquema general del modelo puntos de función en la imagen.

Imagen 58 Esquema del Modelo de métricas Puntos de Función



Fuente: Tutorialspoint.com

Para aplicar el método por puntos de función, se deben tener en cuenta dos pasos como son las funciones disponibles para el usuario acorde a su nivel de complejidad que puede ser simple, media o compleja y el ajuste de acuerdo a catorce características del entorno, las que se explican a continuación.

- Funciones disponibles para el **usuario** acorde a su nivel de complejidad

Se identifican cinco funciones disponibles para el usuario final, los cuales se organizan en cinco grupos, que posteriormente son clasificados de acuerdo a su complejidad (simple, media, compleja) así:

- **Entradas:** Corresponde a cada dato que el usuario introduce a la aplicación y que permite actualizar los datos de los archivos lógicos internos, conjunto de datos, tablas o datos independientes, transacciones recibidas de otras aplicaciones, con ingresos a través de:
  - ✓ Mouse
  - ✓ Teclado
  - ✓ Documentos MICR (Magnetic ink character recognition)
  - ✓ Transacciones de cintas

- ✓ Pantallas sensitivas
- ✓ Lectores de código de barras, etc.

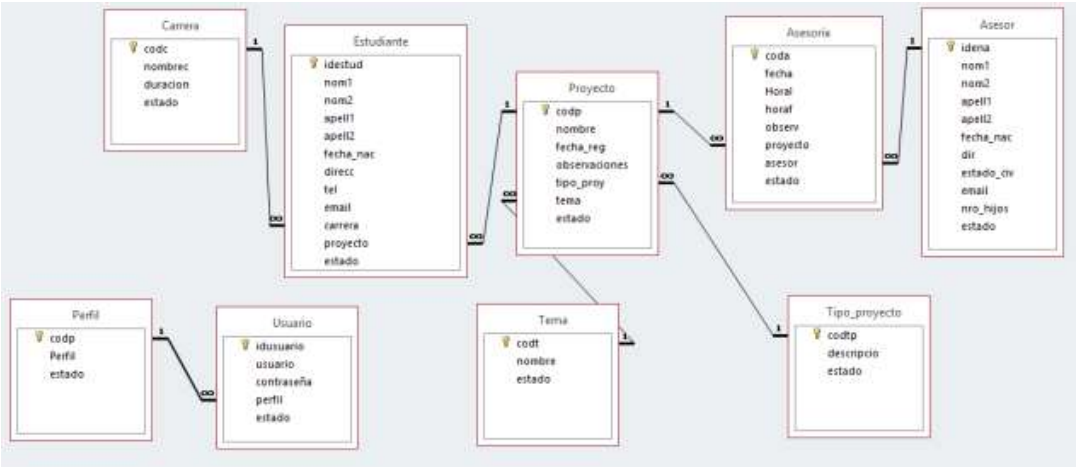
Imagen 59 Complejidad de las entradas al sistema

Entradas	Items de datos Referenciados		
Tablas Referenciadas	1 a 4	5 a 15	16 o más
0 - 1	Simple (3)	Simple (3)	Medio (4)
2 - 3	Simple (3)	Medio (4)	Complejo (6)
4 o más	Medio (4)	Complejo (6)	Complejo (6)

Fuente: elaboración propia.

Es así que, para determinar la complejidad de la entrada estudiante, de debe analizar las tablas que se requieren para la creación de la vista, así como el número de ítems (campos) que se tendrán en cuenta para el diseño, lo cual da como resultado un valor de ponderación (un número, según tabla anterior) que permitirá, según el modelo de puntos de función, determinar la complejidad de dicha entrada al sistema. Si observamos el modelo relacional (imagen 62) del proyecto de grado (caso de estudio), nos podemos dar cuenta que intervienen tres tablas y el diseño de la vista de usuario toma 12 items (ver imagen 63 vista Estudiante), por lo tanto 3 tablas que intervienen y 12 items, da como resultado que la vista estudiante tiene una complejidad media, que según la tabla propuesta por el modelo puntos de función para las entradas equivale a un valor de ponderación de cuatro (4).

Imagen 60 Modelo relacional (Proyecto de Grado)



Fuente: elaboración propia.

Imagen 61 Vista Gestión Estudiante

Fuente: elaboración propia.

Éste proceso se debe hacer con todas las entradas al sistema, donde se debe multiplicar cada entrada por el valor de ponderación y al final sumar todas las entradas. En éste caso en el subtotal aparece (1) que se relaciona con una vista o sea "Gestión estudiante" y cuatro es el valor de ponderación.

Imagen 62 Entradas al sistema, acorde a su nivel de complejidad para Gestión Estudiante

Entradas							
Nombre	**Tablas**	**Items**	**Complejidad**			**Subtotal**	**Total**
			Simple	**Media**	**Compleja**		
Gestión Estudiante	Estudiante, Carrera, Proyecto	12		4		1*4	4
Otra entrada							
Otra entrada							
…						Total entradas	Xxx

Fuente: elaboración propia.

- **Salidas:** Se debe contar cada dato único de usuario o salida de control generado que sale del límite de la aplicación. Esto incluye Informes y mensajes a otras aplicaciones como por ejemplo informes por pantalla, en papel, enviado a otros dispositivos, entre otros.

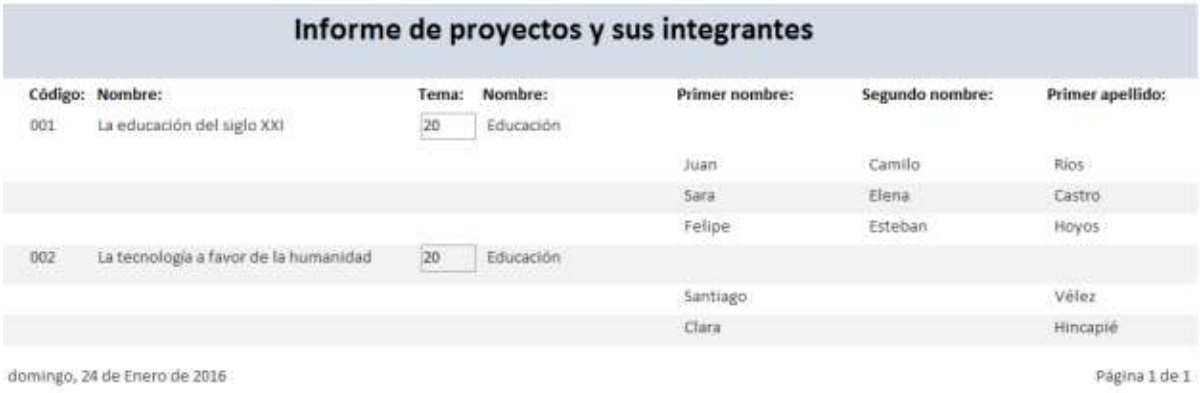
Imagen 63 Complejidad de los informes, acorde al modelo Puntos de Función.

Salidas	Items de datos Referenciados		
Tablas Referenciadas	1 a 5	6 a 19	20 o más
0 - 1	Simple (4)	Simple (4)	Medio (5)
2 - 3	Simple (4)	Medio (5)	Complejo (7)
4 o más	Medio (5)	Complejo (7)	Complejo (7)

Fuente: elaboración propia.

Es así que si se desea diseñar el siguiente informe sobre Proyecto de Grado y sus integrantes, observando el modelo de datos, nos podemos dar cuenta que se toma información de tres tablas (estudiante, proyecto, tema) y que además la cantidad de ítems tomados para el informe equivale a siete ítems, por lo tanto la complejidad del informe es media y equivale a un valor de ponderación, según el modelo de puntos de función de (5).

Imagen 64 Informe de Proyectos y sus integrantes



Informe de proyectos y sus integrantes

Código:	Nombre:	Tema:	Nombre:	Primer nombre:	Segundo nombre:	Primer apellido:
001	La educación del siglo XXI	20	Educación			
				Juan	Camilo	Rios
				Sara	Elena	Castro
				Felipe	Esteban	Hoyos
002	La tecnología a favor de la humanidad	20	Educación			
				Santiago		Vélez
				Clara		Hincapié

domingo, 24 de Enero de 2016 Página 1 de 1

Fuente: elaboración propia.

Éste proceso se debe hacer con todas las salidas del sistema, donde se debe multiplicar cada salida por el valor de ponderación y al final sumar todas las salidas. En éste caso en el subtotal aparece (1) que se relaciona con un informe o sea “Informe de proyectos y sus integrantes” y cinco es el valor de ponderación.

Imagen 65 Complejidad para el informe de proyectos y sus estudiantes

Salidas							
Nombre	**Tablas**	**Items**	**Complejidad**			**Subtotal**	**Total**
			Simple	**Media**	**Compleja**		
Informe de proyectos y sus estudiantes	Estudiante, Proyecto, Tema	7		5		1*5	5
Otra Salida							
Otra Salida							
…						Total salidas	xxx

Fuente: elaboración propia.

- **Consultas:** se debe contar cada combinación única de entrada/salida en la que la entrada on-line definida por el usuario genera una salida inmediata on-line. Las consultas se pueden proporcionar a/desde otra aplicación; por ejemplo, responder a otra aplicación que pregunta por el precio de un producto se contaría como una consulta.

Imagen 66 complejidad de las consultas, acorde al modelo Puntos de Función.

Consultas Entrada	Items de datos Referenciados		
Tablas Referenciadas	**1 a 4**	**5 a 15**	**16 o más**
0 - 1	Simple (3)	Simple (3)	Medio (4)
2	Simple (3)	Medio (4)	Complejo (6)
3 o más	Medio (4)	Complejo (6)	Complejo (6)

Fuente: elaboración propia.

De igual forma que los informes, se tiene en cuenta la cantidad de tablas que intervienen en la consulta, así como los ítems que se mostrarán.

- **Tablas o Ficheros:** Se debe contar cada grupo lógico de datos de usuario o de información de control mantenidos dentro de los límites de la aplicación. Se pueden encontrar tablas maestras o de referencia, tablas mantenidas por los usuarios como estados, tarifas, mensajes, entre otros; tablas de procesamiento de datos o transaccionales, índices de referencias cruzadas.

La siguiente tabla permite clasificar la complejidad de las tablas de la base de datos, acorde al modelo Puntos de Función.

Tablas	Items de datos Referenciados		
Formato/Relación Registro Logico	1 a 19	20 a 50	51 o más
1	Simple (7)	Simple (7)	Medio (10)
2 a 5	Simple (7)	Medio (10)	Complejo (15)
6 o más	Medio (10)	Complejo (15)	Complejo (15)

Fuente: elaboración propia.

Éste proceso se debe hacer con todas las tablas de la base de datos, incluyendo las tablas intermedias, teniendo en cuenta las relaciones que alimentan la tabla que se está creando, en el caso de ejemplo, al crear la tabla proyecto, nos damos cuenta que recibe información de la tabla tipo de proyecto y de la tabla tema, por lo tanto se cuentan las dos anteriores más ella misma, o sea que en éste caso la tabla proyecto posee siete ítems, por lo tanto se ubica de acuerdo a la tabla propuesta por el modelo puntos de función como una tabla de complejidad simple con un valor de ponderación de siete (7). Ver tabla xxx

Imagen 67 Almacenamientos de la base de datos (tablas), acorde a su nivel de complejidad para la tabla proyecto.

Tablas							
Nombre	Tablas	Items	Complejidad			Subtotal	Total
			Simple	Media	Compleja		
Tabla Proyecto	Proyecto recibe de Tema y Tipo Proyecto	7	7			1*7	7
Otra Tabla							
Otra Tabla							
…						Total Tablas	xxx

Fuente: elaboración propia.

- **Interfaces:** Se debe contar como uno cada archivo lógico de otro grupo de datos (o información de control) que se envía fuera de los límites de la aplicación, o se comparte o es recibido desde otra

aplicación. Las interfaces se pueden encontrar en archivos lógicos internos accesibles desde otra aplicación, en archivos lógicos internos accedidos en otra aplicación, base de datos compartida, lista de parámetros compartida, archivos de impresión exportado, archivo de transacción compartido que requiere conversión, entre otros datos externos al software y que requieran de una interfaz para compartir del software a otros software o viceversa.

Imagen 68 Complejidad de las interfaces, acorde al modelo Puntos de Función.

Interfaces	Items de datos Referenciados		
Formato/Relación Registro Logico	1 a 19	20 a 50	51 o más
1	Simple (5)	Simple (5)	Medio (7)
2 a 5	Simple (5)	Medio (7)	Complejo (10)
6 o más	Medio (7)	Complejo (10)	Complejo (10)

Fuente: elaboración propia.

En conclusión, a éste primer aspecto relacionado las cinco funciones disponibles para el usuario acorde a su nivel de complejidad, según IFPUG (International Function Point User Group) se pueden resumir en la siguiente tabla.

Imagen 69 Resumen sobre Funciones disponibles para el usuario (Nivel de Complejidad)

Resumen sobre Funciones disponibles para el usuario (Nivel de Complejidad)			
Funciones	**Simple**	**Media**	**Compleja**
Entradas Externas	3	4	6
Salidas Externas	4	5	7
Consultas Externas	3	4	6
Tablas (Ficheros) internos	7	10	15
Interfaces Externas	5	7	10

Fuente: elaboración propia.

Suponiendo que tenemos las siguientes funciones disponibles para el usuario.

Imagen 70 Resumen sobre Funciones disponibles para el usuario (Nivel de Complejidad)

Resumen sobre Funciones disponibles para el usuario (Nivel de Complejidad)						
Funciones	**Cantidad**	**Simple**	**Media**	**Compleja**	**Fórmula**	**Subtotal**
Entradas Externas	10	6*3	3*4	1*6	((6*3)+(3*4)+(1*6))	36

Salidas Externas	5	4*4	1*5	0*7	((4*4)+(1*5)+(0*7))	21
Consultas Externas	3	3*3	0*4	0*6	((3*3)+(0*4)+(0*6))	9
Tablas (Ficheros) internos	12	8*7	2*10	0*15	((8*7)+(2*10)+(0*15))	76
Interfaces Externas	1	0*5	1*7	0*10	((0*5)+(1*7)+(0*10))	7
					Cuenta Total (Puntos de Función sin ajustar)	149

Fuente: elaboración propia

Características generales del entorno, acorde al tipo de software

Según este método, la cuenta de puntos de función no ajustada debe calibrarse con otros 14 elementos que dependen del entorno, los que se deben calificar de cero (0) a cinco (5), donde cero equivale a menos importancia, acorde al tipo de software y cinco como de mayor importancia. El valor resultante, no se debe dividir, se aplica directamente a la fórmula para hallar los Puntos de Función.

Estos son:

1. Comunicaciones de datos
2. Datos o procesamiento distribuidos
3. Objetivos de rendimiento
4. Configuración utilizada masivamente
5. Tasa de transacción
6. Entrada de datos on-line
7. Eficiencia para el usuario
8. Actualización on-line
9. Procesamiento complejo
10. Reutilización
11. Facilidad de instalación y conversión
12. Facilidad de operación
13. Puestos múltiples
14. Facilidad de cambio.

En la siguiente tabla se amplía cada uno de los puntos expuestos

Imagen 71 Características generales del entorno (14 puntos calificables de 0 a 5)

Característica	Descripción
Comunicación de datos	Cuántas facilidades de comunicación hay disponibles para ayudar en el intercambio de información"con la aplicación o el sistema?
Procesamiento distribuido de datos	Cómo se manejan los datos y las funciones de procesamiento distribuido?
Rendimiento	Existen requerimientos de velocidad o tiempo de respuesta?
Configuraciones fuertemente utilizadas	Qué tan intensivamente se utiliza la plataforma de hardware donde se ejecutará la aplicación o el sistema?
Frecuencia de transacciones	Que tan frecuentemente se ejecutan las transacciones? Diariamente, semanalmente, mensualmente?
Entrada de datos on-line	Qué porcentaje de la información se ingresa on-line?
Eficiencia del usuario final	Se designa la aplicación para maximizar la eficiencia del usuario final?
Actualizaciones on-line	Cuántos Archivos Lógicos Internos se actualizan por una transacción on-line?
Procesamiento complejo	Hay procesamientos lógicos o matemáticos intensivos en la aplicación?
Reusabilidad	La aplicación se desarrolla para suplir una o muchas de las necesidades de los usuarios?
Facilidad de instalación	Qué tan difícil es la instalación y la conversión al nuevo sistema?
Facilidad de operación	Que tan efectivos o automatizados son los procedimientos de arranque, parada, backup y restore del sistema?
Instalación en distintos lugares	La aplicación fue concebida para su instalación en múltiples sitios y organizaciones?
Facilidad de cambio	La aplicación fue concebida para facilitar los cambios sobre la misma?

Al calificar, se puede tener la siguiente información

Imagen 72 Características Generales del entorno calificadas

Característica	Descripción	Peso
Comunicación de datos	Aplicación web	3
Procesamiento distribuido de datos	No hay procesamiento distribuido, pero hay datos distribuídos	2
Rendimiento	No hay requerimientos especiales de rendimiento	0
Configuraciones fuertemente utilizadas	No hay restricciones con respecto al hardware	0
Frecuencia de transacciones	Hay un pico diario de transacciones	3
Entrada de datos on-line	Todos los datos se ingresan on-line	5
Eficiencia del usuario final	Media	3
Actualizaciones on-line	La mayoría de los archivos se actualizan on-line	3
Procesamiento complejo	No hay procesamiento lógico ni matemático complejo	0
Reusabilidad	Se pretende algún grado de reutilización	2
Facilidad de instalación	No hay restricciones	0
Facilidad de operación	Operación desatendida	5
Instalación en distintos lugares	No se requiere más de una instalación	0
Facilidad de cambio	Media	3

Al sumar los 14 puntos de ajuste, da como resultado 29, valor que se utiliza para despejar la fórmula y con los puntos de función sin ajustar, acorde a las cinco funciones disponibles para el usuario, donde se tiene una cuenta total de 149.

Imagen 73 Fórmula para hallar los Puntos de función

Fórmula para hallar los Puntos de función
PF= Cuenta total * [0.65 + 0.01 x Σ(Fi)]

Cuenta total=149
Fi (equivale a los 14 puntos de ajuste)= 29
PF= 149*[0.65 + (0.01*29)]
PF= 149*[0.65+0.29]
PF= 149*0.94
PF=141
Por lo tanto, el software analizado, acorde a los datos presentados, tiene 141 funcionalidades para el usuario final.

Se pueden crear variedad de fórmulas apoyados en Puntos de Función, como, por ejemplo:

-Errores pon puntos de función= (E/PF)

-Defectos por puntos de función= (D/PF)

Suponiendo que se encontraron 300 errores y 20 defectos, se puede calcular errores por PF y defectos por PF así:

Errores por PF = 300/141 =2.12 Lo que equivale aproximadamente a 3 error por cada punto de función.

Defectos por PF = 20/141=0.14 Lo que equivale aproximadamente a 1 defecto por cada 8 puntos de función.

Es importante tener en cuenta que existen y se pueden hacer métricas de muchos tipos, no sólo basadas en el tamaño (LDC) y la funcionalidad (PF) sino también basadas en otras mediciones como, por ejemplo, proporción de pruebas ejecutadas exitosamente Vs. El número de defectos identificados; también se puede hacer métricas primitivas que se utilizan para monitorear el desempeño de una aplicación como, por ejemplo, utilización de memoria, tiempo de finalización de consultas y complejidad de módulos.

Según el modelo de Puntos de Función, estima que un software elaborado con lenguajes de cuarta generación que corresponden en nuestro caso a lenguajes como Java, .Net, PHP, entre otros. Cada Punto de Función se debe desarrollar en un promedio de 8 horas. Ver tabla.

Imagen 74 Tabla presentada para el modelo Puntos de Función, basada en estudio para varios lenguajes de programación

Tabla presentada para el modelo Puntos de Función, basada en estudio para varios lenguajes de programación		
Lenguaje	**Horas por PF promedio**	**Líneas de código por PF**
Ensamblador	25	300
Cobol	15	100
Lenguajes de cuarta generación	**8**	**20**

Lo que según nuestro caso indica que la cantidad de horas que se requiere para construir un software con 141 funcionalidades es de mil ciento veintiocho horas (1128).

Tiempo total en horas=PFA*8

Donde= 141*8=1128 horas requeridas (Siendo lo anterior medidas estimadas)

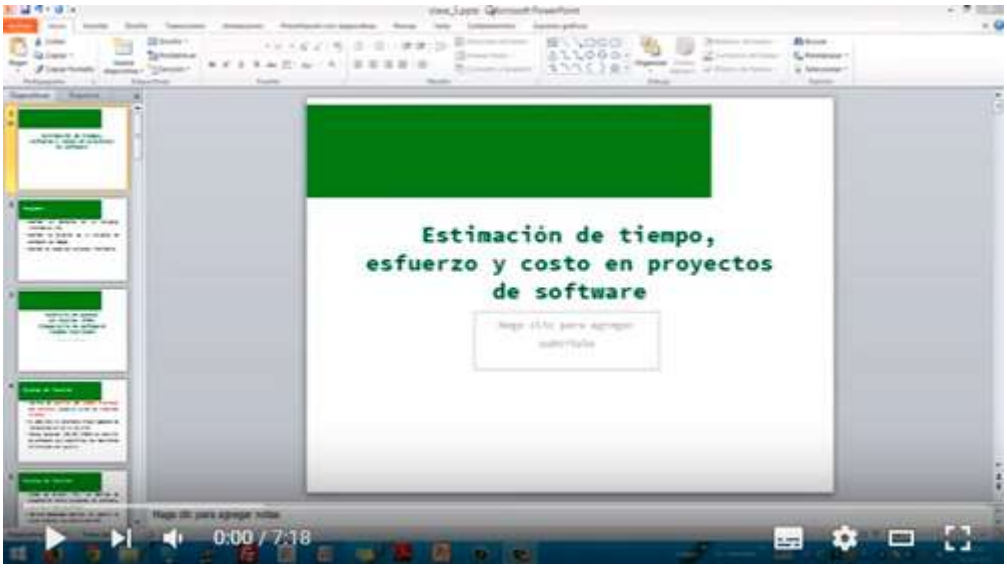
De igual forma se podría fácilmente, hallar el valor total del proyecto.

Costo del proyecto=Tiempo total en horas*Valor hora

Suponiendo que la empresa paga por el valor de la hora diez mil peso (10.000), quedaría de la siguiente forma:

Costo total del proyecto: 1128*10000=11´280.000

Para complementar la temática. Ver videos sugeridos sobre Puntos de Función:



Análisis de puntos de función + EJEMPLO [1/3] Enlace

Análisis de puntos de función + EJEMPLO [2/3] Enlace

Análisis de puntos de función + EJEMPLO [3/3] Enlace

- Taller de aplicación al proyecto

En éste caso se analizarán algunas métricas basadas en las líneas de código y puntos de función que ayuden a realizar una estimación sobre el costo de un producto de software, aplicados al proyecto de software que se desarrolla desde la Ingeniería de Software I, dándole continuidad en la Ingeniería de software II y terminación completamente funcional, en la Ingeniería de Software III.

4.3.6 MÉTRICAS DE COMOMO

Las métricas de COCOMO, no se abordará en éste módulo, sin embargo, se recomienda link que permite realizar una inducción sobre sus principales funcionalidades. ver video Métricas de COCOMO

Cocomo II Enlace

4.3.7 TALLER DE ENTRENAMIENTO UNIDAD 3

Nombre del taller: Taller 3	Modalidad de trabajo: Aprendizaje basado en problemas
Actividad previa: Realización por parte del estudiante del taller 1, y revisión, retroalimentación por parte del docente, donde se debe contar con todo el diseño de la base de datos, diseño de interfaz gráfica (Menú, formularios, Informes, consultas).	
Describa la actividad: Evolucionando la solución del taller 1 hacia la aplicación de métricas orientada a los Puntos de Función. Se pide: **Calcular el valor de puntos de función:** • Basarse en las cinco funciones disponibles para el usuario final (Entradas, Salidas, Consultas, Ficheros (tablas), Interfaces), clasificados de acuerdo a su complejidad (simple, media, compleja).	

- Evaluar las 14 características generales del entorno (evaluar de 0 a 5, de acuerdo a la importancia para éste tipo de software ((Fi(i=1 a 14)).

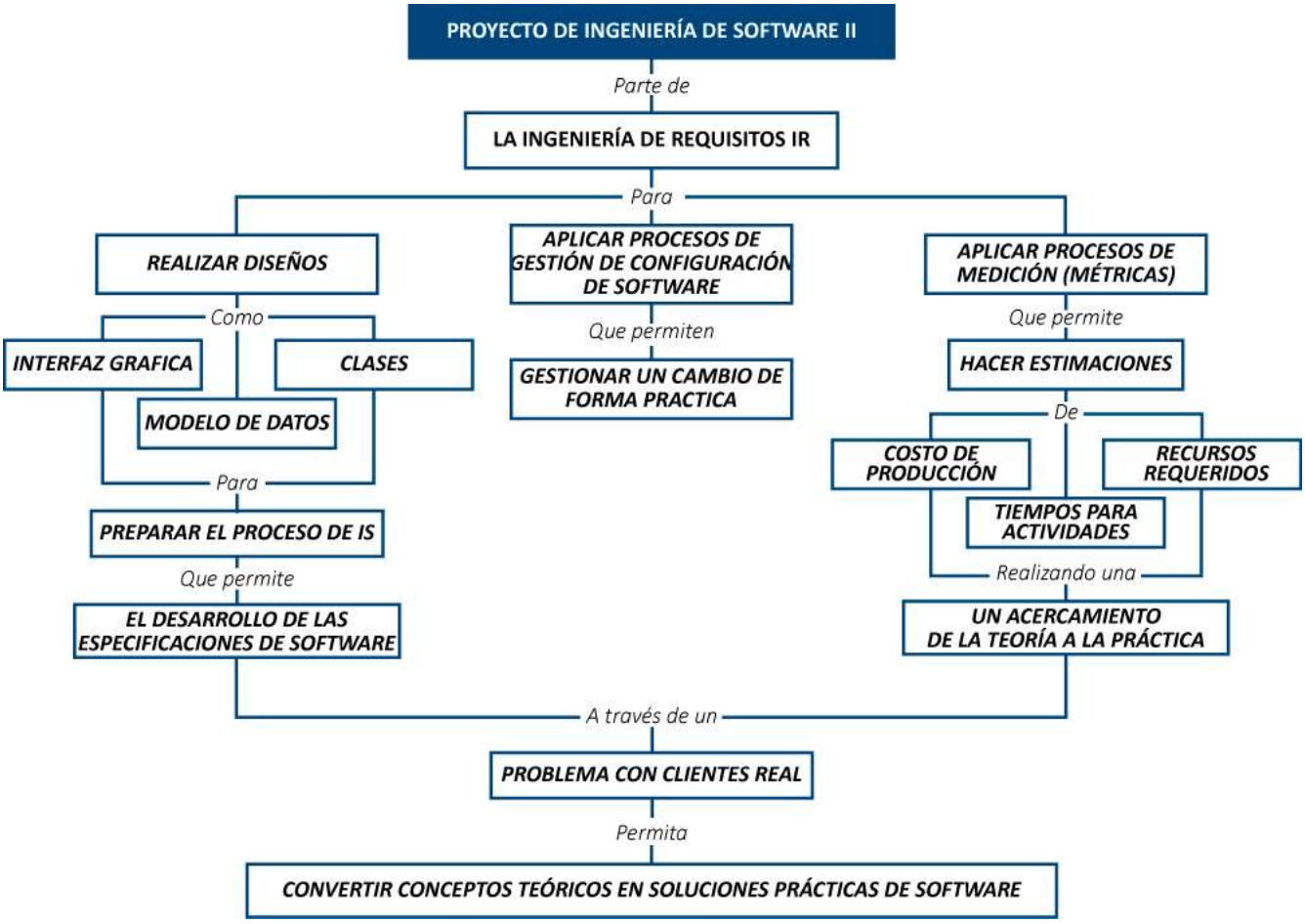
- Calcular el punto de función (PF=(CUENTA TOTAL*[0.65+0.01 x E(Fi)]

Análisis:

- Analizar la utilidad de los puntos de función.

- Teniendo en cuenta el nivel de compejidad de las diferentes funciones disponibles para el usuario final representado en (Entradas, Salidas, Consultas, Ficheros (tablas), Interfaces) y sabiendo que la hora de construcción del software en cualquier actividad dentro del ciclo de vida del software, tiene un valor de ($10.000), diez mil pesos. ¿Cuánto costaría la producción de todo el software? (Se debe estimar el tiempo de acuerdo a la complejidad de cada componente).

5 UNIDAD 4: PROYECTO DE INGENIERÍA DE SOFTWARE (APRENDIZAJE BASADO EN PROBLEMAS E INVESTIGACIÓN FORMATIVA)



5.1 TEMA 1: GUÍA INGENIERÍA DE SOFTWARE II (DISEÑOS DE SOFTWARE, ADMINISTRACIÓN DE RIESGOS, GESTIÓN DE CONFIGURACIÓN Y MEDICIÓN)

Dentro de la orientación del proceso pedagógico en relación a la asignatura Ingeniería de Software II, se trabajará con aprendizaje basado en problemas (ABP) e investigación formativa, donde se dará continuidad al proyecto de la Ingeniería de Software I, se adicionará herramientas, métodos, técnicas, buenas prácticas que permitan la

evolución de las especificaciones hacia el diseño y desarrollo de software, como estructura fundamental en la construcción de un producto informático de calidad.

5.2 TEMA 2: DESARROLLO DE PROYECTO DE SOFTWARE (GUÍA INGENIERÍA DE SOFTWARE II)

El desarrollo de un proyecto con cliente real, permite al estudiante aplicar los conceptos teóricos a una situación real, con el fin de dar solución a una necesidad informacional a través de herramientas de Ingeniería de Software, hasta el diseño e iniciación de la codificación (apoyo de lenguaje de programación) relacionado con el acceso al sistema, además de la aplicación de gestión de cambios, medición y administración del proyecto.

Dicho proyecto deberá ajustarse desde el inicio (Ingeniería de Software I), ya que el estudiante va adquiriendo nuevas competencias que le permiten mejorar cada una de sus partes en la medida que se fortalecen sus conocimientos en la Ingeniería de Software II.

5.2.1 COMPONENTE DE LA GUÍA DE INGENIERÍA DE SOFTWARE II

1. Portada
2. Tabla de contenido generada por el sistema
3. Glosario
4. Introducción
5. Justificación del proyecto
6. Ubicación de la empresa donde se realizará el proyecto (Razón social, Reseña histórica, Misión, Visión, Objeto social, Sector al que pertenece, Ubicación geográfica, Datos del contacto, Tamaño de la empresa, Organigrama, Carta de aceptación para la realización del proyecto, firmada)
7. Elicitación de Requerimientos (Extracción, análisis, modelado, validación)
8. Planteamiento del problema (antecedentes, formulación de pregunta problematizadora, justificación del problema)
9. Necesidades y características
10. Objetivo General

11. Objetivos específicos
12. Alcance del sistema propuesto
13. Nombre del sistema propuesto
14. Cronograma de actividades (actividad, tiempo, recurso)
15. Análisis de riesgo (proyecto, proceso, producto)
16. Análisis de requisitos (RU, RF, RNF, Tabla general casos de uso)
18. Especificación de requisitos (Diagramas de casos de uso, diagramas de secuencia, escenarios, etc.)
19. Modelado (Clases, modelo de datos, documentación)
20. Mapa de navegación
21. Diseños (entradas, procesos, salidas)
23. Prototipo funcional (acceso al sistema, Menú principal)
24. Proceso de Gestión de Configuración aplicado al proyecto
24. Métricas de software para funcionalidad y valoración de proyecto
20. Estimación de Recursos
21. Conclusiones
22. Referentes

El proyecto, deberá estar fortalecido por cada una de las temáticas relacionadas con la Ingeniería de Software II, con la debida asesoría del docente.

5.2.2 TALLER DE ENTRENAMIENTO UNIDAD 4

Nombre del taller: Proyecto	Modalidad de trabajo: Aprendizaje basado en problemas e investigación formativa.
Actividad previa: Estudio de unidades, acorde a tiempos para entregables	
Describa la actividad: La actividad, se desarrollará a lo largo de módulo, donde se realizarán asesorías físicas y/o virtuales en relación a los diferentes entregables, los cuales deberán ser suficientemente retroalimentados, tanto por parte del	

docente como del estudiante quien debe realizar los ajustes pertinentes, hasta que el proyecto cumpla con los objetivos del módulo.

El proyecto debe ser individual, evitando dificultades en caso de cambio de jornada, cancelación de materia, entre otras, para evitar que deba iniciar otro proyecto, para las materias ingeniería de software II y III.

6 PISTAS DE APRENDIZAJE

Sabía usted que todas las entidades que se relacionan con el diseño son objetos como por ejemplo persona, automóvil, universidad, la cuales contienen atributos y métodos que se pasan mensajes unos con otros.

Tenga presente que una clase, es la descripción general de un objeto, donde los atributos son las variables y los métodos son las funcionalidades del sistema

Sabía que el concepto de encapsulación es unir en un solo lugar objetos y sirve para restringir el acceso a datos y métodos desde el exterior, recibiendo también el nombre de ocultación de la información.

Sabía usted que el concepto de herencia en la POO, está relacionada con la capacidad que tienen los objetos de importar, implementar, reutilizar variables y métodos de otras clases y que funciona como la herencia del nieto al padre y del padre al abuelo.

Tenga presente que el polimorfismo, se relaciona con la capacidad que tiene una interfaz de realizar varias tareas, dependiendo de cómo se aplique determinada función.

Sabía usted, que las métricas de software ayudan a detectan errores y defectos a tiempo, minimizando reproceso y pérdida de recursos.

Tenga presente que los riesgos, se presentan de improviso, por lo que se aconseja preverlos.

Sabía que la GCS, es una actividad que buscan controlar los cambios para que el sistema conserve su calidad.

Sabía que las métricas, no son utilizadas actualmente, en todos los procesos del ciclo de vida del software, sólo se utilizan en su mayoría en las etapas de codificación y pruebas.

Sabía que la Ingeniería de Software es una disciplina joven, que viene aprendiendo de las buenas prácticas.

Sabía que el modelo relacional fue creado por Codd, a inicio de los años 70 y que aún sigue vigente para la estructuración de las bases de datos relacionales.

7 GLOSARIO

- **Acoplamiento:** Dos elementos están acoplados en la medida en el que los cambios en uno tienden a necesitar cambios en el otro. Se relaciona con el grado de relación de un módulo con los demás, donde a menor acoplamiento, será más conveniente, ya que será más sencillo de diseñar, programar, probar y mantener.
- **Arquitectura de software**: Conjunto de patrones que proporcionan un marco de referencia para la elaboración de un producto informático. La arquitectura, permite establecer la estructura, funcionamiento e interacción entre las partes del software.
- **Array:** tiene relación con el almacenamiento continuo de elementos, que desde el punto de vista lógico se encuentran debidamente ordenados, dentro de almacenamientos temporales, entre ellos se tienen los vectores y las matrices.
- **Atomicidad:** característica que deben cumplir los campos dentro de la base de datos, consistente en que cada unidad de datos posee un significad indivisible, por ejemplo, primer nombre.
- **Base de datos:** dos o más tablas relacionadas a través de claves primarias o foráneas, sobre la cual se puede crear, consultar, modificar, actualizar datos e información, siendo especiales para almacenamiento fijo.
- **Calidad:** Se relaciona con el grado de satisfacción del cliente en relación al producto de software y las funcionalidades de éste en cuanto a requisitos de usuario, requisitos funcionales y requisitos no funcionales.
- **Clase.** Estructura estática relacionada con la programación orientada a objetivos, que se utiliza para definir las características de un objeto y las acciones que se pueden ejecutar sobre dichos objetos, como por ejemplo la clase estudiante y la acción consultar estudiante.
- **Cliente servidor:** Es una estructura para la distribución de tareas entre un computador principal llamado servidor y unos computadores secundarios, llamado clientes, donde el servidor posee recursos o servicios y el cliente solicita recursos y servicios.
- **Cohesión:** Se relaciona con la independencia que tienen los módulos del sistema para realizar un proceso o entidad, donde a mayor cohesión, el módulo en mención será más sencillo de diseñar, programar, probar y mantener.

- **Componentes:** son los elementos que posee un objeto como los atributos, identidad, relaciones y métodos.
- **Concurrencia:** es una propiedad de los sistemas en la cual los procesos de un cómputo se hacen simultáneamente, y pueden interactuar entre ellos.
- **Configuración de software:** conjunto de procesos que se realizan para asegurar la calidad del software, en caso de generarse cambios que requieran la adaptación del sistema a nuevas necesidades.
- **Dependencia funcional:** relación que existe entre atributos, donde el uno depende totalmente del otro, como por ejemplo la cédula de una persona tiene asociada un nombre y un apellido.
- **Dependencia transitiva:** relación que existe entre atributos, donde un atributo no se identifica totalmente del principal, como por ejemplo la cédula de una persona, se encuentra asociada con su nombre, pero no se asocia directamente con el nombre del acudiente, ya que acudiente posee sus propias características como su cédula, nombre, apellido.
- **Diseño:** proceso del ciclo de vida del software que consiste en definir la arquitectura, los componentes, interfaces, los datos, convirtiendo una especificación en un producto funcional, ayudado por el lenguaje de programación.
- **Diseño Arquitectónico:** consiste en describir las estructuras y organización del software de alto nivel e identificar los componentes que lo forman.
- **Escenario:** consiste en la descripción de un sistema, donde se muestra su comportamiento ante determinados estímulos externos y su respuesta a ellos. Los escenarios sirven para definir, que hace y que no hace un sistema.
- **Estructura de datos**: medios para manejar grandes cantidades de datos de manera eficiente, entre las que podemos contar vectores, matrices, archivos, bases de datos.
- **Fiabilidad:** probabilidad de que un sistema funcione de acuerdo a sus requisitos funcionales, durante un período determinado.
- **Foreign Key:** se refiere a un campo clave llamado clave secundaria o extranjera, que se conecta con una clave primaria, a través de una relación.
- **Formas Normales**: se refiere a un conjunto de reglas, que buscan estandarizar el modelo relacional, con el fin de evitar errores en los datos e información.
- **Interfaz**: componente que permite transformar los datos o señales generadas por un sistema en datos, señales e información comprensibles por otro.

- **Lenguaje de Programación:** software que permite crear otro software, a través de ciertos códigos que facilitan la conversión de los requerimientos del cliente en requisitos funcionales.
- **Métricas:** es una metodología de planificación, desarrollo y mantenimiento de los sistemas de software. Herramienta que busca realizar controles, a cada una de las etapas del ciclo de vida del software, a través de fórmulas, estándares, indicadores, buscando la optimización del tiempo, recursos, procesos, buscando la calidad del software.
- **Métricas COCOMO**: (COnstructive COst MOdel), modelo matemático, que se basa en la experiencia en la construcción del software, utilizado para la estimación de costos en el proceso de ingeniería de software.
- **Métricas de LOC** (Lines Of Code): considerada como una de las primeras métricas para predecir la fiabilidad y la complejidad de las aplicaciones, basada en la cantidad de líneas de código.
- **Métricas orientadas a los Puntos de Función**: métrica que permite estimar el tamaño de un Sistema Software teniendo en cuenta las diferentes funciones que el software deberá de realizar, y posteriormente asignando un valor, en puntos, a cada una de esas funciones.
- **Modelo relacional:** modelo para estructurar bases de datos, propuesto por Codd, a inicio de los años 70, actualmente es el modelo más utilizado en la industria del software.
- **Modularidad:** Capacidad que tiene un sistema de ser estudiado, visto o entendido como la unión de varias partes que interactúan entre sí y que trabajan para alcanzar un objetivo común.
- **Objetos**: Componentes de un modelo como unidades de código para resolver situaciones específicas.
- **Patrones:** base para la búsqueda de soluciones a problemas comunes en el desarrollo de software, como por ejemplo las interfaces.
- **Primary key:** Campo único que identifica los demás campos de un registro y no se puede repetir en la tabla.
- Proyecto: Esquema, programa o plan que realiza para coordinar, actividades, recursos, riesgos, buscando cumplir con la meta del proyecto.
- **Registro**: Conjunto de campos interrelacionado que se identifican con un campo único.
- **Reglas del negocio**: Condiciones que debe cumplir un software, acorde a las necesidades del cliente.
- **Relaciones**: Asociación entre entidades, a través de los campos clave.
- **Restricción**: Se refiere a las condiciones que debe cumplir un software, con excepciones claras.

- **Reutilización**: Condición que debe cumplir la programación orientada a objetos, en cuanto al aprovechamiento de código en otros procesos, evitando programarlo nuevamente.
- **Riesgo**: Se refiere a todo aquello que atenta contra el no desarrollo de un proyecto.
- **Seguridad:** Una de las condiciones del software, en cuanto a calidad, en relación a la protección contra accesos no autorizados.
- **SGBD**: Sistema de Gestión de Base de Datos, que a través de un motor de base de datos, está en capacidad de administrar los datos e información.
- **Tabla**: objeto compuesto por filas y columnas, el cual es utilizado para almacenar datos e información.

8 BIBLIOGRAFÍA

8.1 FUENTES BIBLIOGRÁFICAS

- Charette, R. N. (1989), Software Engineering Risk Analysis and Management, McGraw-HillDntertext.
- G. Kotonya and I.(2000) Sommerville, Requirements Engineering: Processes and Techniques, John Wiley & Sons
- IEEE Computer Society, (2014), SWEBOK (Guide to the Software Engineering Body of Knowledge), Version 3.0. ISBN-10: 0-7695-5166-1
- Jacobson, Ivar; Booch, Grady; Rumbaugh, James (2000) (en Español). El Proceso Unificado de Desarrollo de Software. Pearson Addisson-Wesley.
- Piattini, Mario G. (1996), Análisis y diseño detallado de aplicaciones informáticas de gestión. 1ª ed. RA-MA Editorial, Madrid, 1996
- Pressman, R. (2010). Ingeniería de Software un Enfoque práctico, Séptima Edición. ISBN 978-607-15-0314-5.
- Sommerville, Lan. (2011) Ingeniería de software, novena edición. Pearson, México. ISBN 0137035152 | 9780137035151
- Sommerville, Lan (2005), Ingeniería de software, séptima edición, Pearson Educación, Madrid (España) ISBN: 84-7829-074-5
- [Wasserman, 1996] Wasserman, A. “Toward a Discipline of Software Engineering”. IEEE Software, 13(6):23-31. November/December 1996

8.2 FUENTES DIGITALES O ELECTRÓNICAS

- Barzana, A & Menéndez, R. (2005), Gestión de Riesgos en Ingeniería de Software. Consultado el 29 de febrero de 2016 de http://www.wikilearning.com/curso_gratis/gestion_de_riesgos_en_ingenieria_del_software-introduccion/3620-1

- IEEE Computer Society, (2004), Software Engineering Body of Knowledge, Consultado el 09 de noviembre de 2015 de: http://www.swebok.org

- IEEE Computer Society, (2014), SWEBOK (Guide to the Software Engineering Body of Knowledge), Version 3.0. ISBN-10: 0-7695-5166-1, consultado el 09 de noviembre de 2015 de: http://www.computer.org/web/swebok/v3-guide
- ISO/IEC 25000, SQuaRE, (2014). Consultado el 09 de noviembre de 2015 de: System and Software Quality Requirements and Evaluation.
- Real Academia Española, Asociación de Academias de la Lengua Española. Diccionario de la lengua española, 23.ª ed., Edición del Tricentenario, [en línea]. Madrid: Espasa, 2014.
- Diccionario WordReference Copyright / derecho de autor © 2016 WordReference.com. http://www.wordreference.com/definicion/dise%C3%B1o
- Hernández, ITH, (2013), Arquitectura de software. http://angehernandezith.blogspot.com.co/2013/05/unidad-3-ingenieria-de-software.html
- J.B. Dreger, "Function Point Analysis", Prentice-Hall, Inc. Upper Saddle River, NJ, USA ©1989. ISBN:0-13-332321-8
- ISO/IEC 2502n. (2014), División de Medición de Calidad.
- Wordreference Diccionario. (2005). Medir.
- Thefreedictionary. (2013). Medida.
- Software -Diseño Complejidad http://www.tutorialspoint.com/es/software_engineering/software_design_strategies.htm
- Juristo N, Moreno A. Vegas Sira. (2006), Técnicas de evaluación de Software, consultado el 29 de febrero de 2016 de web:http://www.grise.upm.es/htdocs/docencia/erdsi_old/Documentacion_Evaluacion_7.pdf

8.3 FUENTES BASES DE DATOS ESPECIALIZADAS

- http://scholar.google.es/
- http://dialnet.unirioja.es/
- http://www2.ebsco.com/es-es/Pages/index.aspx
- http://biblioteca.remington.edu.co/es/recursos-electronicos/bibliotecas-virtuales
- http://www.redalyc.org/
- http://biblioteca.remington.edu.co/es/recursos-electronicos/bases-de-datos-libres

8.4 VIDEOS

- historia del diseño industrial: https://www.youtube.com/watch?v=pLuJU_-fyXw.
- Ver video Normalización de Base de Datos (https://www.youtube.com/watch?v=bO18omSzeR4)
- Diccionario. Definicion ABC http://www.definicionabc.com/general/riesgo.php
- Para complementar la temática. Ver videos sugeridos sobre Puntos de Función:
  - Video 1: https://www.youtube.com/watch?v=BeP6dXdLLo8
  - Video 2: youtube.com/watch?v=GWuYmAbdycA#t=115.635547
  - Video 3: https://www.youtube.com/watch?v=0wbALQ9lz7o
- Riesgo al Generar Productos de Software https://www.youtube.com/watch?v=vqDjFdGYRFo
- Gestión de proyectos y riesgos https://www.youtube.com/watch?v=2WLZFsntabY
- Gestión de Proyectos Informáticos en https://www.youtube.com/watch?v=6yC8ZmbZDrw
- Película sobre Steve Jobs (creador de Apple), https://www.youtube.com/watch?v=-gxMnH2qzgU#t=6392.013577
- Métricas de Software (https://www.youtube.com/watch?v=y68Tx1ogdu0)
- Métricas de COCOMO_https://www.youtube.com/watch?v=SRwwTLfcgsE

LISTA DE IMÁGENES