
VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN**

# "DEPÓSITO DE BUENAS PRÁCTICAS DE INGENIERÍA DE SOFTWARE Y BASES DE DATOS"

# T E S I S

QUE PARA OBTENER EL GRADO DE:

## MAESTRA EN INGENIERÍA
**(COMPUTACIÓN)**

**P R E S E N T A:**


## GLORIA ANGÉLICA NOGUEZ BARAJAS

**DIRECTORA DE TESIS:**

**M. EN C. MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ**

**CODIRECTORA DE TESIS:**

**DRA. AMPARO LÓPEZ GAONA**


**MÉXICO, D.F.** **2010.**

*Agradecimientos*

*A ese SER maravillo que siempre está conmigo y guía mi camino.*

*A mi familia*

*Por su cariño, comprensión y apoyo incondicional en todo momento.*

*A mis profesores*

*Por su disposición y por transmitir abiertamente sus conocimientos.*

*A mis amigos*

*Por compartir conmigo momentos tan especiales.*

# Índice

# Índice de Figuras

# Introducción

## Antecedentes

El estudio de este trabajo surge a raíz de que en el seminario de Ingeniería de Software y Bases de Datos, tienen como objetivo promover el uso de las buenas prácticas. Haciendo una investigación se encontró que existen algunos sitios donde se encuentran buenas prácticas pero de un solo tema por ejemplo JSP o en EJB, pero no están organizadas, la información está presentada como un documento, sin especificar para qué son, cuales son los beneficios que se adquieran al utilizar, etc.
Lo mismo en libros, únicamente se enfocan a un cierto tema y no se cuenta con diversidad de temas.

## Objetivos

- Impulsar la difusión de buenas prácticas para que trasciendan positivamente en las personas dedicadas al desarrollo de software.
- Hacer la recopilación de buenas prácticas.
- Generar la plantilla descriptiva de las buenas prácticas.
- Clasificar las buenas prácticas.
- Desarrollar una aplicación web a través de la cual se puedan consultar las buenas prácticas.

## Beneficios Esperados

Que los proyectos no fracasen por falta de información, y que sean proyectos de calidad.
Que la búsqueda de las buenas prácticas sea sencilla.
Tener buenas prácticas de diferentes temas almacenadas en un mismo depósito.

## Alcances y Límites

Se clasificarán las buenas prácticas. Se podrán consultar por ciertos criterios de búsqueda y se podrán proponer nuevas prácticas mediante la generación de un archivo XML.
El administrador será quien defina si es una buena práctica para ser almacenada en un depósito.

## Justificación del depósito

No se cuenta con un lugar donde se puedan consultar las buenas prácticas clasificadas, ordenadas y con diversos temas.

Organización de la Tesis

Esta tesis consiste de los siguientes capítulos:

- Introducción. Se describe el problema a resolver y su solución.
- Capítulo 1. Las Buenas Prácticas. En este capítulo se habla sobre la definición y clasificación de las buenas prácticas, la generación de la plantilla, que contiene los puntos que serán tomados en cuenta para poder proponer una buena práctica. Y la diferencia que existe entre las buenas prácticas y los patrones.
- Capítulo 2. Herramientas de desarrollo. En este capítulo se realiza un análisis de diferentes herramientas para poder desarrollar el sistema en el cual se podrá almacenar y consultar las buenas prácticas.
- Capítulo 3. Planteamiento del problema y propuesta de solución. En este capítulo se realiza el levantamiento de requerimientos, se realiza el diagrama general de casos de uso y su detalle, así como un prototipo para saber cómo quedará el sistema.
- Capítulo 4. Análisis. En este capítulo se identifican las clases, las cuáles se utilizan para generar los diagramas de secuencia.
- Capítulo 5. Diseño. En este capítulo se especifica la arquitectura para desarrollar el sistema, se detallan las clases que obtuvimos en el análisis, y por último se diseña la base de datos.
- Capítulo 6. Implementación y Pruebas. En este capítulo se muestra una parte del código de clases y el reporte del plan de prueba del sistema.
- Capítulo 7. Navegación del Sistema. En este capítulo se proporciona una guía para entender el funcionamiento del sistema y uso del depósito. La navegación se divide en tres: Navegación del Ingeniero de Software, Navegación del Ponente, y Navegación del Administrador.
- Conclusiones
- Referencias Bibliográficas
- Anexos

# Capítulo 1

# Las Buenas prácticas de Ingeniería de Software y Bases de Datos

## 1.1. Introducción

Aunque el término “buenas prácticas” es utilizado prácticamente a diario en el ámbito de sistemas, no se tiene una definición clara sobre este término. Es por ello que a lo largo de este capítulo se darán algunas definiciones que organismos de desarrollo de software reconocidos dan para esta expresión.

Las buenas prácticas son importantes ya que se busca trabajar de la mejor forma y a partir de la experiencia generada por otros; para evitar perder tiempo en intentar descubrir el hilo negro, si ya tenemos la manera de resolver un problema pues se utiliza y se continúa con el desarrollo. En muchos casos hasta se puede ahorrar dinero y esfuerzo humano. Es por esta razón que las buenas prácticas van ocupando un lugar importante para todos los desarrolladores de software, además de que sirven para tener un punto de referencia para comparar esfuerzos.

Lo que se intenta en este trabajo, es fomentar la difusión de buenas prácticas que repercutan positivamente, además de facilitar el aprendizaje mediante el intercambio continuo de experiencias.

El área de ingeniería de software y bases de datos es muy extensa, por lo que es necesario clasificar las buenas prácticas.

Para realizar este trabajo es necesario hacer una recopilación de buenas prácticas y para esto se pedirán entrevistas dentro del ámbito empresarial, dedicadas a la ingeniería de software, además se hará una investigación en libros y se asistirá a seminarios y conferencias para que el depósito cuente con una vasta colección de buenas prácticas. De esta manera se tendrán las buenas prácticas administradas y centralizadas para una mejor consulta y utilización.

## 1.2. Definición de buenas prácticas

Empecemos por definir lo que es una práctica. Proviene del latín “practicus”, que significa lo que se realiza o se lleva a cabo conforme reglas o a la costumbre [1].

Para Rational Software Corporation las mejores prácticas son “un conjunto de enfoques probados y aplicados al desarrollo de software que cuando son usados en

combinación, impactan directamente en las causas raíz de los problemas del desarrollo de software, su uso común es observable en la industria dentro de las organizaciones más exitosas" [2].

"Es una técnica, método, proceso, actividad, incentivo o reconocimiento, que es más efectivo que cualquier otro conocido, para entregar cierto resultado" [3].

Las buenas prácticas deben ser interpretadas como recomendaciones, nunca como guías rígidas o planes de trabajo.

"Son buenas prácticas porque en la correcta aplicación de estas habilidades, herramientas y técnicas se puede aumentar la posibilidad de éxito de muchos proyectos, no por que deban aplicarse siempre de forma uniforme en todos" [4].

"Acción o conjunto de acciones que, fruto de la identificación de una necesidad, son sistemáticas, eficaces, eficientes, sostenibles, flexibles y están pensadas y realizadas por miembros de una organización con el apoyo de sus órganos de dirección, y que, además de satisfacer las necesidades y expectativas de sus clientes, suponen una mejora evidente de sus procesos" [5].

Se pueden distinguir dos tipos de buenas prácticas:

- Innovadoras. Se introducen por primera vez en un servicio, centro u organización y son innovadoras porque no se han hecho antes.
- Excelentes. Retoman buenas prácticas anteriores desarrolladas en cualquier organización aprovechando los conocimientos que contienen para mejorar su eficiencia y eficacia.

También son conocidas como consejos (tips), técnicas probadas [6].

Es importante conocer la definición de "Buenas Prácticas" ya que de esta manera podemos entender mejor su razón de ser, evitando errores en sus aplicaciones, pues debemos de entender que son aplicables en ciertas ocasiones y no en todos los casos.

Con base en lo anterior, se pueden definir las buenas prácticas como: Acción o conjunto de acciones que han sido probadas y aplicadas al desarrollo de software, que al ser utilizadas adecuadamente, satisfacen las necesidades de la organización y dan mejora en sus procesos.

## 1.3. Modelos base para clasificar las áreas

Para poder clasificar las buenas prácticas se requiere analizarlas y buscando en la literatura se tomarán algunos estándares.

### 1.3.1 **COMPETISOFT** (Modelo de Referencia de Procesos) [7]

Este Modelo de Procesos está conformado por tres categorías que agrupan procesos de acuerdo a la estructura típica de una organización:

- Categoría de Alta Dirección. Establece la razón de ser de la organización, lo que se desea alcanzar y las estrategias para lograrlo. Contiene al proceso de Gestión de Negocio.
- Categoría de Gerencia. Establece planes de acción para instrumentar las estrategias en cuanto a proyectos, procesos y recursos necesarios, realiza monitoreo de la categoría de Operación y retroalimenta a la categoría de Alta Dirección. Contiene tres procesos:
    - Gestión de Procesos.
    - Gestión de Cartera de Proyectos.
    - Gestión de Recursos (Humanos, Bienes, Servicios e Infraestructura, Conocimiento).
- Categoría de Operación. Realiza proyectos de desarrollo o mantenimiento de software que cubran las necesidades del cliente en el tiempo y costo esperados y reporta los resultados a la categoría de Gerencia. Contiene:
    - Administración del Proyecto
    - Desarrollo de Software
    - Mantenimiento de Software

### 1.3.2 **RUP** (Rational Unified Process) [8]

Este modelo trata sobre el desarrollo exitoso de software orientado a objetos asegurando alta calidad de manera repetible y predecible.
Se conforma por nueve disciplinas, a continuación se explicarán brevemente:

- Modelado de negocios: El propósito del modelado de negocios es entender problemas actuales en la organización e identificar potenciales de mejora, determinar el impacto del cambio así como asegurarse de que clientes, usuarios finales, desarrolladores, y otros involucrados tengan un entendimiento común de la organización.

- Requerimientos: El propósito de la disciplina de requerimientos es establecer y mantener un acuerdo con el cliente y otros involucrados en lo que el sistema debe hacer, proveer a los desarrolladores del sistema un entendimiento de los

requerimientos, definir los límites del sistema, proveer las bases para planear las iteraciones de contenido técnico y para estimar tiempo y costo del desarrollo del sistema y definir la interfaz de usuario del sistema, enfocándose a las necesidades y metas de los usuarios.

- Análisis y diseño: El propósito de la disciplina de análisis y diseño es transformar los requerimientos en un diseño de lo que el sistema debe hacer, desarrollar una arquitectura robusta para el sistema y adaptar el diseño para combinar con el ambiente de implementación.

- Implementación: El propósito de la disciplina de implementación es construir los elementos de diseño en términos de elementos de código (archivos de fuente, binarios, ejecutables, y otros), probar los componentes desarrollados como unidades, integrar los resultados producidos en un ejecutable del sistema.

- Pruebas: El propósito de la disciplina de pruebas es encontrar y documentar defectos en la calidad del software, validar que el producto de software trabaja según lo diseñado y que los requerimientos están implementados apropiadamente.

- Despliegue: El propósito de la disciplina de despliegue es describir las actividades asociadas con el aseguramiento de que el producto de software está disponible para su fin.

- Administración de la configuración y cambios: El propósito de la disciplina de administración de la configuración y cambios es identificar elementos de configuración, restricción de cambios de esos elementos, auditar los cambios realizados a esos elementos, y definir y administrar las configuraciones de esos elementos.

- Administración de proyectos: El propósito de la disciplina de administración de proyectos es proporcionar un marco de trabajo para manejar proyectos orientados al software, las guías prácticas para la planeación, proveyendo de personal, ejecutando y monitoreando los proyectos y finalmente proporcionar un marco de trabajo para la administración de riesgos.

- Ambiente: La disciplina del ambiente describe las actividades requeridas para desarrollar las directrices en apoyo del proyecto.

### 1.3.3 **PMBOK** (Project Management Body of Knowledge). [9]

Es un estándar reconocido internacionalmente (IEEE, ANSI) trabaja con el uso del conocimiento, de las habilidades, de las herramientas y de las técnicas para resolver requisitos del proyecto.
Esta guía define un ciclo vital del proyecto, cinco grupos de proceso y nueve áreas de conocimiento de la tarea de administración de proyectos.

Los cinco grupos de proceso son:

1. El inicio.
2. Planificación.
3. Ejecución
4. Supervisión y control.
5. Cierre.

Las nueve áreas de conocimiento de los proyectos son:

1. Integración.
2. Alcance.
3. Tiempo.
4. Costo.
5. Calidad.
6. Recurso humano.
7. Comunicaciones.
8. Riesgos.
9. Adquisiciones.

Muchos de los proyectos de Ingeniería de Software fracasan en gran medida por su informalidad y muchos otros por su excesiva e inflexible formalidad. Es importante emplear una visión de proyectos para construir estrategias de mejora en el desarrollo de sistemas, ya que problemas básicos de los proyectos como requerimientos mal planificados, retraso y presupuesto irreal pueden ser disminuidos con la aplicación de la Gestión de Proyectos.

La disciplina de la Gestión de Proyecto proporciona el marco por el cual se crea y maneja un proyecto. Al hacer eso, el resto de las disciplinas se utilizan como parte del trabajo de proyecto.

## 1.4. Clasificación de Buenas Prácticas

Con base a los estándares existentes antes mencionados, se pueden clasificar las buenas prácticas en:

1. Ingeniería de Software,
2. Bases de Datos, y
3. Administración de Proyectos

Se decidió que la Ingeniería de Software sea una de las áreas ya que abarca al grupo de métodos, técnicas y herramientas que se utilizan en la producción del software.

Otra área es el área de Bases de Datos. Es un área importante ya que es donde se tiene almacenada la información y es importante que funcione bien ya que de ella depende en gran parte el sistema, y mientras mejor desempeño tenga el desarrollo de la base de datos también lo será el del sistema. Se atiende como un ejemplo específico de los elementos principales de desarrollo

Por último, se concluyó que la Administración de Proyectos sea considerada como una de las áreas para las buenas prácticas, ya que es un área bastante amplia, además de que requiere la gestión de proyectos para que se pueda desarrollar una aplicación en el plazo previsto y con el presupuesto establecido y de esta manera tanto el cliente como los desarrolladores queden satisfechos con la aplicación.

Algunos temas de interés relativo a estas áreas por su gran auge en la actualidad son los siguientes:

**Ingeniería de Software**

- Ingeniería de Requisitos
- Arquitecturas Software y Patrones de Diseño
- Calidad, Medición y Estimación de Productos y Procesos Software
- Diseño de páginas web
- Ingeniería del Software basada en Componentes
- Desarrollo de Software Orientado a Aspectos
- Mantenimiento, Prueba, Verificación y Validación
- Ingeniería Web
- Procesos de Negocio e Ingeniería de Servicios
- Seguridad en Ingeniería del Software

**Bases de Datos**

- Tecnología SGBD (optimización de consultas, transacciones, seguridad, vistas, etc.)
- Modelado de Datos (conceptual y lógico)
- BD deductivas, BD activas y BD orientadas a objetos
- BD columnares
- BD móviles, BD distribuidas
- Operación Continua de las Bases de Datos
- Minería de Datos
- Data Streaming y Almacenes de Datos
- Sistemas de Información Federados
- Web Semántica
- Recuperación de Información, Indexación y BD en Web
- XML y Datos Semiestructurados

**Administración de Proyectos**

- Administración estratégica y cuantitativa
- Administración por objetivos
- Administración del portafolio
- Dirección de proyectos
- Gestión de Requerimientos
- Ejecución del Proyecto
- Generación de Software
- Oficina de proyectos

## 1.5. Generación de la plantilla

En el libro PL/SQL "Best Practices" [6] se intenta clasificar las "Buenas Prácticas" por temas y clasificarlas como si fueran patrones, así mismo, en el seminario de Ingeniería de Software y Bases de Datos [3] también se da una clasificación de las buena prácticas, y después de platicar con especialistas mencionaron que observaron que existiendo un problema que se solucionó y después de que ese problema se presentara varias veces y lo resolvían de la misma manera, se dieron cuenta de que ese problema atacándolo de esa forma se solucionaba. Entonces como la buena práctica es para solucionar un problema pues es necesaria la descripción, que es saber lo que se hizo para resolver el problema y después es necesario saber que se obtuvo y para esto se requiere tener un beneficio. El área es para saber en qué área de especialización y aún más acotado se tiene la subárea ya que el área puede ser extensa. También se tenía que nombrar de alguna forma a esa práctica y eso por eso que se pide un título. El rol se pidió ya que resultó interesante el saber quién podía consultar la práctica. El autor y la referencia se piden como información para saber qué especialistas generaron la buena práctica y el ejemplo es por si no basta con la descripción para darnos una mejor idea de cómo utilizar la práctica.

A continuación, se presenta la plantilla con los puntos que se observaron eran necesarios para especificar una buena práctica:

| | |
|---|---|
| Área | En que parte del desarrollo de sistemas se utiliza |
| Subárea | En que parte del área se utiliza |
| Rol | Quién puede utilizar la buena práctica |
| Título | Nombre que se le da a la buena práctica para identificarla y describirla |
| Problema | Qué es lo que se tiene que mejorar |
| Descripción | Explicación de lo que tiene que hacer la buena práctica |
| Beneficio | ¿Por qué deberíamos molestarnos con esta buena práctica? ¿Qué tan decisivo es para uno seguir esta recomendación |

| | |
|---|---|
| | en particular? Es un resumen de los principales beneficios que obtendremos siguiendo esta buena práctica. |
| Autor | Quién creó la buena práctica |
| Referencia | De dónde se obtuvo la buena práctica |
| Ejemplo | Donde se utilizó la buena práctica, ya que puede ser más ilustrativo el valor de la buena práctica. |

## 1.6. Mapeo de Roles

Para realizar el mapeo de los roles se analizarán los roles propuestos en MoProSoft y el Proceso Unificado, a continuación se presentan las tablas de cada uno:

Tabla de roles MoProSoft

| Rol | Abreviatura | Descripción |
|---|---|---|
| Cliente | CL | Es quien solicita el producto de software y financia el proyecto para su desarrollo o mantenimiento. |
| Usuario | US | Es el que utilizará el producto de software. |
| Grupo Directivo | GD | Es el encargado de dirigir la organización y son los responsables de su funcionamiento exitoso. |
| Responsable de Proceso | RP | Es el encargado de que se lleven a cabo las prácticas de un proceso y del cumplimiento de sus objetivos. |
| Involucrado | | Se refiere a otros roles que pueden ser requeridos por sus habilidades para realizar actividades o tareas específicas. Por ejemplo: Analista, Programador, entre otros. |
| Responsable de Gestión de Negocio | RGN | Es quien define e implanta exitosamente el proceso de Gestión de Negocio y que se realice con éxito. |
| Responsable de Gestión de Proceso | RGP | Es el encargado de definir e implantar exitosamente el proceso de Gestión de Procesos. |
| Responsable de Gestión de Proyectos | RGPY | Es el encargado de realizar Gestión de Proyectos. |
| Responsable de la Administración del Proyecto Específico | RAPE | Es el que realizará la administración de los proyectos específicos por lo que deberá contar con: capacidad de liderazgo con experiencia en la toma de decisiones, planificación estratégica, manejo de personal, delegación y supervisión, |

| | | |
|---|---|---|
| | | finanzas y desarrollo de software. |
| Grupo de Gestión | GG | Es el encargado de realizar las propuestas de mejora al Plan Estratégico e implantar los procesos definidos. |
| Responsable de Gestión de Recursos | RGR | Es el encargado de definir e implantar exitosamente el proceso de Gestión de Recursos y sus subprocesos. |
| Responsable de Recursos Humanos y Ambiente de Trabajo | RRHAT | Debe tener el conocimiento de las actividades necesarias para implantar exitosamente el subproceso de Recursos Humanos y Ambiente de Trabajo. |
| Responsable de Bienes, Servicios e infraestructura | RBSI | Debe tener el conocimiento de las actividades necesarias para implantar exitosamente el subproceso de Bienes, Servicios e infraestructura. |
| Responsable de Conocimiento de la Organización | RCO | Es el encargado de garantizar la integridad, seguridad y eficiencia de la Base de Conocimiento. |
| Responsable de Capacitación | RC | Es el encargado de implantar la capacitación solicitada. |
| Equipo de Trabajo | ET | El Equipo de Trabajo debe contar con conocimiento y experiencia de acuerdo a su rol. |
| Grupo de Responsables de Procesos | GRP | Debe conocer las necesidades del proceso con respecto a la Base de Conocimiento. |
| Responsable del Subcontrato | RSC | Es el encargado de administrar los subcontratos por lo que debe tener conocimiento de administración de proyectos. |
| Responsable de Desarrollo y Mantenimiento de Software | RDM | Es el encargado del proceso de desarrollo y mantenimiento de software por lo tanto debe contar con conocimiento y experiencia en este proceso. |
| Analista | AN | Es el encargado de la obtención, especificación y análisis de los requerimientos, debe tener conocimiento y experiencia. |
| Diseñador de Interfaz de Usuario | DU | Es el encargado de realizar el diseño de las interfaces de usuario, debe tener conocimientos de diseño y criterios ergonómicos para las interfaces de usuario. |
| Diseñador | DI | Es el encargado del diseño de la estructura de los componentes de software. |

| | | |
|---|---|---|
| Programador | PR | Es el encargado de la programación, integración y de realizar las pruebas unitarias. |
| Responsable de Pruebas | RPU | Es el encargado de la planificación y realización de las pruebas de integración y de sistema. |
| Revisor | RE | Es el encargado de aplicar las técnicas de revisión y debe tener experiencia en el desarrollo y mantenimiento de software. |
| Responsable de Manuales | RM | Es el encargado de generar los manuales por lo tanto debe tener experiencia en redacción en el desarrollo y mantenimiento de software. |
| Evaluador | EV | Es el encargado de aplicar las evaluaciones, debe contar con conocimiento en metodologías de evaluación. |

Tabla de roles de la tesis Componentes de bases de datos para MoProSoft [10]

| Rol | Abreviatura | Descripción |
|---|---|---|
| Administrador de la Base de Datos | ABD | Conocimiento y experiencia en administración de base de datos. |
| Diseñador de la Base de Datos | DBD | Conocimiento y experiencia en diseño de base de datos. |
| Programador | PR | Es el encargado de la programación, integración y de realizar las pruebas unitarias a nivel aplicación y base de datos. |
| Responsable de Pruebas | RPU | Es el encargado de la planificación y realización de las pruebas de integración y de sistema. Deberán contemplarse conocimiento y experiencia de pruebas de integración a nivel base de datos. |
| Revisor | RE | Es el encargado de aplicar las técnicas de revisión y debe tener experiencia en el desarrollo y mantenimiento de software y de base de datos. |
| Responsable de Manuales | RM | Es el encargado de generar los manuales por lo tanto debe tener experiencia en redacción en el desarrollo y mantenimiento de software y de base de datos. |

Tabla de roles del Proceso Unificado

| Rol | Mapeo con MoProSoft | Descripción |
|---|---|---|

| | | |
|---|---|---|
| Analista | AN | Es el responsable de definir todos los requerimientos funcionales y no funcionales. |
| Especificador de casos de uso | DI | Es el responsable de uno o varios casos de uso. |
| Diseñador de interfaz de usuario | DU | Dan forma visual a las interfaces de usuario. Esto puede implicar el desarrollo de prototipos de interfaces de usuario para algunos casos de uso. |
| Arquitecto | DI, PR | Da prioridad a los casos de uso para planear la arquitectura del análisis y es el responsable por la integridad del modelo del análisis, del diseño y de instalación. Crea el modelo de implementación y describe la arquitectura. |
| Ingeniero de casos de uso | DI | Responsable de la integridad de la realización de uno o varios casos de uso. |
| Ingeniero de componentes | DI | Responsable de una o varias clases del análisis y paquetes del análisis. Implementa componentes, la interfaz, hace pruebas unitarias y prueba componentes según el plan de pruebas. |
| Integrador de sistemas | DI, RPU | Planifica la secuencia de construcciones necesarias en cada interacción y la integración de cada construcción cuando sus partes han sido implementadas. |
| Diseñador de pruebas | RPU | Planea las pruebas, crea los casos de prueba y registra los resultados. |
| Ingeniero de pruebas de integración | RPU | Efectúa las pruebas de integración |
| Ingeniero de pruebas del sistemas | RPU | Efectúa la prueba del sistema y se encarga de documentar los defectos en los resultados de las pruebas de sistema |
| Administrador del proyecto | CL, US, GD, RP, RGN, RGP, RGPY, RAPE, GG, RGR, RRHAT, RBSI, RCO, RC, ET, GRP, RSC | Se encarga de terminar el Proyecto a tiempo, dentro del presupuesto y de acuerdo a las condiciones de satisfacción del Cliente. |

De acuerdo a sus funciones se incorporaron los roles de MoProSoft y los propuestos en la tesis Componentes de bases de datos para MoProSoft [10] a los del Proceso Unificado, para que de esta manera todos los roles sean tomados en cuenta.

El nombre de los roles que se utilizará para la plantilla serán los utilizados por MoProSoft y la tesis Componentes de bases de datos para MoProSoft [10] ya que están más especificados y separados por sus funciones proporcionando de esta manera facilidad para definir el rol de la persona que puede utilizar la buena práctica.

Dentro de la tabla de los roles de MoProSoft se incorporará el área en el que el rol se desenvolverá.

Ingeniería de Software: IS

Bases de Datos: BD

Administración de Proyectos: AP

| Rol | Área | Abreviatura | Descripción |
|---|---|---|---|
| Responsable de Proceso | AP | RP | Es el encargado de que se lleven a cabo las prácticas de un proceso y del cumplimiento de sus objetivos. |
| Involucrado | | | Se refiere a otros roles que pueden ser requeridos por sus habilidades para realizar actividades o tareas específicas. Por ejemplo: Analista, Programador, entre otros. |
| Responsable de Gestión de Negocio | AP | RGN | Es quien define e implanta exitosamente el proceso de Gestión de Negocio y que se realice con éxito. |
| Responsable de Gestión de Proceso | AP | RGP | Es el encargado de definir e implantar exitosamente el proceso de Gestión de Procesos. |
| Responsable de Gestión de Proyectos | AP | RGPY | Es el encargado de realizar Gestión de Proyectos. |
| Responsable de la Administración del Proyecto Específico | AP | RAPE | Es el que realizará la administración de los proyectos específicos por lo que deberá contar con: capacidad de liderazgo con experiencia en la toma de decisiones, planificación estratégica, manejo de personal, delegación y supervisión, finanzas y desarrollo de software. |
| Grupo de Gestión | AP | GG | Es el encargado de realizar las propuestas de mejora al Plan Estratégico e implantar los |

| | | | |
|---|---|---|---|
| | | | procesos definidos. |
| Responsable de Gestión de Recursos | AP | RGR | Es el encargado de definir e implantar exitosamente el proceso de Gestión de Recursos y sus subprocesos. |
| Responsable de Recursos Humanos y Ambiente de Trabajo | AP | RRHAT | Debe tener el conocimiento de las actividades necesarias para implantar exitosamente el subproceso de Recursos Humanos y Ambiente de Trabajo. |
| Responsable de Bienes, Servicios e infraestructura | AP | RBSI | Debe tener el conocimiento de las actividades necesarias para implantar exitosamente el subproceso de Bienes, Servicios e infraestructura. |
| Responsable de Conocimiento de la Organización | AP | RCO | Es el encargado de garantizar la integridad, seguridad y eficiencia de la Base de Conocimiento. |
| Responsable de Capacitación | AP | RC | Es el encargado de implantar la capacitación solicitada. |
| Equipo de Trabajo | AP | ET | El Equipo de Trabajo debe contar con conocimiento y experiencia de acuerdo a su rol. |
| Grupo de Responsables de Procesos | AP | GRP | Debe conocer las necesidades del proceso con respecto a la Base de Conocimiento. |
| Responsable del Subcontrato | AP | RSC | Es el encargado de administrar los subcontratos por lo que debe tener conocimiento de administración de proyectos. |
| Responsable de Desarrollo y Mantenimiento de Software | IS | RDM | Es el encargado del proceso de desarrollo y mantenimiento de software por lo tanto debe contar con conocimiento y experiencia en este proceso. |
| Analista | IS | AN | Es el encargado de la obtención, especificación y análisis de los requerimientos, debe tener conocimiento y experiencia. |
| Diseñador de Interfaz de Usuario | IS | DU | Es el encargado de realizar el diseño de las interfaces de usuario, debe tener conocimientos de diseño y criterios ergonómicos para las interfaces de usuario. |
| Diseñador | IS | DI | Es el encargado del diseño de la estructura de los componentes de software. |

| | | | |
|---|---|---|---|
| Programador | IS, BD | PR | Es el encargado de la programación, integración y de realizar las pruebas unitarias a nivel aplicación y base de datos. |
| Responsable de Pruebas | IS, BD | RPU | Es el encargado de la planificación y realización de las pruebas de integración y de sistema. Deberán contemplarse conocimiento y experiencia de pruebas de integración a nivel base de datos. |
| Revisor | IS, BD | RE | Es el encargado de aplicar las técnicas de revisión y debe tener experiencia en el desarrollo y mantenimiento de software y de base de datos. |
| Responsable de Manuales | IS, BD | RM | Es el encargado de generar los manuales por lo tanto debe tener experiencia en redacción en el desarrollo y mantenimiento de software y de base de datos. |
| Evaluador | IS | EV | Es el encargado de aplicar las evaluaciones, debe contar con conocimiento en metodologías de evaluación. |
| Administrador de la Base de Datos | BD | ABD | Conocimiento y experiencia en administración de base de datos. |
| Diseñador de la Base de Datos | BD | DBD | Conocimiento y experiencia en diseño de base de datos. |

¿Quién es el Ingeniero de Software? Es la persona que consultará las buenas prácticas, y es cualquier persona que se encuentra dentro del área de sistemas y tecnologías de información. El ponente es similar al Ingeniero sólo que él será quién podrá proponer las buenas prácticas.

El administrador es el que decidirá si se aceptan o no las buenas prácticas, por lo que deberá contar con una amplia experiencia dentro de este campo.

## 1.7. Ejemplos de Buenas Prácticas

Las buenas prácticas se van a obtener de:

- Libros
- Seminarios
- Web

A continuación se muestran algunos ejemplos de buenas prácticas para empezar a introducirnos en las ventajas que nos proporcionan éstas buenas prácticas y el porqué de la necesidad de contar con un depósito para éstas.

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Proceso de desarrollo y mantenimiento de software |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Hacer desarrollo de ciclos |
| Problema | Porque el cliente:<br>• Lo quiere para ayer<br>• Hasta no ver no creer |
| Descripción | Hacer desarrollo de ciclos, para hacer entregables previos a la entrega final. |
| Beneficio | Cuando el cliente lo ve:<br>• Se da cuenta de lo que no quiere ver<br>• Se da cuenta de lo que si quiere ver<br>• Se da cuenta de lo que hace falta |
| Autor | Hanna Oktaba |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Modelado y diseño |
| Rol | Diseñador de la Base de Datos |
| Título | Estandarización de términos |
| Problema | Confusión y pérdida de semántica de los objetos a crear |
| Descripción | Establecer convenciones en el nombrado de la base de datos, y sus objetos, estandarizarlo y apegarse a él. |
| Beneficio | Mantenimiento e interoperabilidad de sistemas más rápido y efectivo. |
| Autor | María del Pilar Ángeles |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Modelado y diseño |
| Rol | Administrador de la Base de Datos |
| Título | Análisis de requerimientos y utilización de índices |
| Problema | Acceso secuencial en lugar de acceso por índices |
| Descripción | Es muy importante que mediante el análisis de requerimientos se obtengan las consultas más populares, de tal forma que la información pueda indexarse y aprovechar el acceso prácticamente directo que nos permiten los árboles al generar los |

| | |
|---|---|
| | índices correspondientes y actualizar el plan de acceso. Por tanto, se recomienda que al diseñar las entidades se identifiquen sus consultas, llaves primarias, foráneas e índices |
| Beneficio | Mejora en el tiempo de respuesta al obtener la información |
| Autor | María del Pilar Ángeles |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Programación SQL |
| Rol | Programador |
| Título | Documentación de objetos SQL |
| Problema | Dada la ilegibilidad de programas, se presenta lentitud y confusión en el mantenimiento de programas |
| Descripción | Documentar los procedimientos almacenados, triggers, y demás programas |
| Beneficio | Tiempo de mantenimiento reducido |
| Autor | María del Pilar Ángeles |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos. |
| Subárea | Diseño y modelado |
| Rol | Diseñador de la Base de Datos |
| Título | Creación de la BD |
| Problema | Identificación de BD y aplicaciones |
| Descripción | Usualmente las BD tienen nombre en “clave”. Asignen nombres que identifiquen la BD, procedimientos y aplicaciones. |
| Beneficio | Mantenimiento ágil de la BD |
| Autor | Gustavo Adolfo López Manjanez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos. |
| Subárea | Modelado de la BD |
| Rol | Diseñador de la Base de Datos |
| Título | Diseño de la BD |
| Problema | Falta de coherencia entre el diseño lógico y físico de la BD |
| Descripción | Implementar efectivamente las necesidades de negocio, Conocimiento de las demandas de negocio. |
| Beneficio | Garantizar la correcta implementación de las soluciones de back-end basadas en el diseño de la BD |
| Autor | Gustavo Adolfo López Manjanez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos. |
| Subárea | Protección de la información de las bases de datos. |
| Rol | Administrador de la Base de Datos |
| Título | Auditoría en las bases de datos. |
| Problema | Violación y uso indebido de la información que se encuentra en las bases de datos |
| Descripción | Implementación de esquemas de auditoría en las bases de datos. |
| Beneficio | Dar seguimiento y detectar de manera oportuna violaciones y/o transacciones que afecten la consistencia y seguridad de la información en las bases de datos |
| Autor | Héctor Bautista Vázquez |

Obtenidas de [3]. En el Anexo A se puede observar un depósito inicial de buenas prácticas, donde se muestra su uso en un mayor número de subáreas y roles.

## 1.8. Diferencias entre Buenas Prácticas y Patrones

Un patrón describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de la solución del problema, de forma que puede utilizarse un millón de veces sin tener que hacer dos veces lo mismo [11].

Los patrones son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.

Para que una solución sea considerada un patrón debe poseer varias características, una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores; otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias [12].

Lenguajes de Patrones

Para describir un patrón se usan lenguajes de patrones que expresan un medio de comunicación uniforme entre los diseñadores.
El más utilizado es el dado por GoF y consta de los siguientes apartados:

| | |
|---|---|
| Nombre del patrón | Nombre estándar del patrón por el cual será reconocido en la comunidad (normalmente se expresan en inglés). |

| | |
|---|---|
| Clasificación del patrón | Creacional, estructural o de comportamiento. |
| Intención | ¿Qué problema pretende resolver el patrón? |
| También conocido como | Otros nombres de uso común para el patrón. |
| Motivación | Escenario de ejemplo para la aplicación del patrón. |
| Aplicabilidad | Usos comunes y criterios de aplicabilidad del patrón. |
| Estructura | Diagramas de clases oportunos para describir las clases que intervienen en el patrón. |
| Participantes | Enumeración y descripción de las entidades abstractas (y sus roles) que participan en el patrón. |
| Colaboraciones | Explicación de las interrelaciones que se dan entre los participantes. |
| Consecuencias | Consecuencias positivas y negativas en el diseño derivadas de la aplicación del patrón. |
| Implementación | Técnicas o comentarios oportunos de cara a la implementación del patrón. |
| Código de ejemplo | Código fuente ejemplo de implementación del patrón. |
| Usos conocidos | Ejemplos de sistemas reales que usan el patrón. |
| Patrones relacionados | Referencias cruzadas con otros patrones. |

Haciendo una comparación con la plantilla de las buenas prácticas, la de los patrones tiene más campos, por lo que se puede decir que las buenas prácticas son más informales que los patrones. Los patrones son más formales en el sentido de que tienen bien definido exactamente lo que se tiene que hacer y en las buenas prácticas es más comentada.

En los patrones hay niveles de granularidad muy pequeños de código y las buenas prácticas son a niveles más abstractos del proceso de desarrollo.

“Se puede decir que las buenas prácticas son algo más generales que los patrones, dado qué éstos sirven para dar soluciones a problemas particulares, bien documentados. Por otro lado, las buenas prácticas se pueden aplicar a cualquier faceta del desarrollo de aplicaciones, desde la escritura de código hasta qué soluciones o técnicas utilizar en determinadas soluciones” [6].

# Capítulo 2

# Herramientas de Desarrollo

Las herramientas que se pueden utilizar para el desarrollo del depósito de las buenas prácticas, son variadas y ofrecen demasiadas alternativas. Escoger las herramientas más adecuadas constituye un aspecto fundamental en el momento de desarrollar una implementación. Por estas razones se debe profundizar e investigar las diferentes alternativas que se tiene al alcance, evitando de esta manera algunos inconvenientes como son: pérdida de tiempo o de dinero, o aún más grave comprometer la credibilidad profesional al avalar un concepto técnico sin el suficiente soporte y conocimiento.

Para proponer una solución que realmente satisfaga las necesidades del depósito, se necesita analizar diversas opciones técnicas. La decisión principal en este momento se enfoca al lenguaje de programación que se empleará para implementar la solución, así como la base de datos a utilizar.

A continuación se evaluarán algunas herramientas que se podrán utilizar para desarrollar el depósito de las buenas prácticas.

## 2.1. Lenguajes de programación

**Java**

En 1991, la empresa Sun Microsystems crea el lenguaje Oak (de la mano de del llamado proyecto Green). Mediante este lenguaje se pretendía crear un sistema de televisión interactiva. Este lenguaje sólo se llegó a utilizar de forma interna. Su propósito era crear un lenguaje independiente de la plataforma y para uso en dispositivos electrónicos.

Sun deseaba un lenguaje para programar pequeños dispositivos electrónicos. La dificultad de estos dispositivos es que cambian continuamente y para que un programa funcione en el siguiente dispositivo aparecido, hay que rescribir el código. Por eso Sun quería crear un lenguaje independiente del dispositivo.

En 1995 pasa a llamarse Java y se da conocer al público. Adquiere notoriedad rápidamente. Java pasa a ser un lenguaje totalmente independiente de la plataforma y a la vez orientado a objetos. Esa filosofía y su facilidad para crear aplicaciones para redes TCP/IP ha hecho que sea uno de los lenguajes más utilizados en la actualidad. La versión actual de Java es el llamado Java 2.

Ventajas:

- No hay punteros (lo que lo hace más seguro)
- Totalmente orientado a objetos
- Preparado para aplicaciones TCP/IP
- Implementa excepciones de forma nativa
- Es un lenguaje intérprete (lo que acelera su ejecución remota, aunque provoca que las aplicaciones Java se ejecuten más lentamente que por ejemplo C++ en un ordenador local).
- Permite multihilos
- Admite firmas digitales
- Tipos de datos y control de sintaxis más rigurosa
- Es independiente de la plataforma, tanto en código fuente como en binario. Esto quiere decir que el código producido por el compilador Java puede transportarse a cualquier plataforma que tenga instalada una máquina virtual Java y ejecutarse. Pensando en Internet esta característica es crucial ya que esta red conecta ordenadores muy distintos.
- La última ventaja (quizá la más importante) se consigue ya que el código Java no se compila, sino que se precompila, de tal forma que se crea un código intermedio (llamado bytecode) que no es ejecutable. Para ejecutarle hace falta pasarle por un intérprete que va ejecutando cada línea. Ese intérprete suele ser la máquina virtual de Java.

En java la unidad fundamental del código es la clase. Son las clases las que se distribuyen en el formato bytecode de Java. Estas clases se cargan dinámicamente durante la ejecución del programa Java.

No se permite el acceso directo a memoria. La primera línea de seguridad de Java es un verificador del bytecode que permite comprobar que el comportamiento del código es correcto y que sigue las reglas de Java. Normalmente los compiladores de Java no pueden generar código que se salte las reglas de seguridad de Java. Pero un programador malévolo podría generar artificialmente código bytecode que se salte las reglas. El verificador intenta eliminar esta posibilidad.

Hay un segundo paso que verifica la seguridad del código que es el verificador de clase que es el programa que proporciona las clases necesarias al código. Lo que hace es asegurarse que las clases que se cargan son realmente las del sistema original de Java y no clases creadas reemplazadas artificialmente.

Finalmente hay un administrador de seguridad que es un programa configurable que permite al usuario indicar niveles de seguridad a su sistema para todos los programas de Java.
Hay también una forma de seguridad relacionada con la confianza y no de una fuente no identificada. En java se pemite añadir firmas difitales para veificar al autor del mismo.

Java es un lenguaje fácil de aprender. Tiene un tamaño pequeño que favorece el desarrollo y reduce las posibilidades de cometer errores; a la vez es flexible.

Para poder crear los bytecodes de un programa Java, hace falta un Java Developer Kit (JDK). Sin embargo, Sun va renovando este kit actualizando el lenguaje.De ahí que se hable de Java 1.1, Java 1.2, etc.}Actualmente se habla de Java 2 para indicar las mejoras en la versión. Desde la versión 1.2 del JDK, el Kit de desarrollo se llama Java 2 Developer Kit en lugar de Java Develoer Kit. La última versión es Java SE 7. [13]

**C#**

A continuación se enumeran las principales características que definen al lenguaje de programación C#. Algunas de estas características no son propias del lenguaje, sino de la plataforma .NET, aunque se mencionan ya que tienen una implicación directa en el lenguaje. [14]

- Sencillez de uso: Es autocontenido ya que un programa en C# no necesita de muchos elementos añadidos por otros lenguajes, como son archivos adicionales al propio código fuente, como son los de cabecera (.h) de C, lo que simplifica la arquitectura de los proyectos.
- Orientado a objetos: C# como lenguaje de última generación, y de propósito general, es orientado a objetos. C# no permite la inclusión de funciones ni variables globales que no estén incluidos en una definición de tipos, por lo que la orientación a objetos es más pura y clara que en otros lenguajes como C++. Además, C# soporta todas las características del paradigma de la programación orientada a objetos, como son la *encapsulación, la herencia* y el *polimorfismo*.
- Orientado a componentes: La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular. La sintaxis de C# incluye por ejemplo formas de definir *propiedades*, *eventos* o *atributos*.
- Recolección de basura: Todo lenguaje incluido en la plataforma .NET tiene a su disposición el recolector de basura del CLR(Common Language Runtime – Lenguaje común en tiempo de ejecución). Esto implica que no es necesario incluir instrucciones de destrucción de objetos en el lenguaje.
- Eficiente: En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a

diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad de procesamiento muy grandes.

- Compatible: Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados Platform Invocation Services (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32.

**PHP**

PHP es un acrónimo recursivo que significa Hipertext Pre-processor. Fue creado originalmente por Ramus Lerdorf en 1994; Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

Es un lenguaje de programación intérprete de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. [15]

Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta.

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, Microsoft SQL Server.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX (y de ese tipo, como Linux o Mac OS X) y Windows, y puede interactuar con los servidores de web más populares como Apache, e ISAPI.

Características de PHP:

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones (desde PHP5).

| Lenguaje de programación | Utilidad | Rendimiento | Portabilidad | Flexibilidad | Documentación | Orientado a objetos | Licencia |
|---|---|---|---|---|---|---|---|
| Java | Muy buena | Muy buena | Muy buena | Muy buena | Muy buena | Si | Libre |
| C# | Muy buena | Muy buena | Muy buena | Muy buena | Buena | Si | Comercial |
| PHP | Buena | Buena | Buena | Buena | Buena | Si | Libre |

Para el desarrollo de nuestra aplicación se utilizará el lenguaje Java en versión 6.0 ya que es un lenguaje multiplataforma potente, soporta el desarrollo rápido de aplicaciones, es sencillo de usar y posee una amplia conexión a bases de datos.

## 2.2. Bases de datos

Para poder implementar el depósito se estudiarán distintas tecnologías de gestión de datos para el almacenamiento de los documentos, cómo recuperarlos y cómo consultarlos. Se utilizarán dos tipos de base de datos y dependiendo del tipo de datos que se quieran gestionar se utilizará una base de datos u otra.

Dato es una colección de hechos considerados de forma aislada. Los datos describen la organización. Estos hechos aislados portan un significado, pero en general no son de utilidad por si solos.

Una base de datos es un conjunto de datos que pertenecen al mismo contexto y son almacenados sistemáticamente para su uso posterior.

Son datos interrelacionados que modelan una realidad. La base de datos es el componente estructural clave en el diseño de sistemas de información. Es la principal fuerza del sistema de integración del sistema de información de una organización. [16]

### 2.2.1 Bases de Datos Relacionales

Una base de datos relacional consiste de una colección de “tablas”.
Cada renglón de una tabla representa una relación entre el conjunto de valores. Esto es conocido formalmente como tupla.

Dado que una tabla es un conjunto de estas relaciones, hay una fuerte correspondencia entre el concepto de tabla y el concepto de relación.

Un atributo representa las propiedades de la relación, y se representan mediante columnas en las tablas. Cada atributo de una relación se caracteriza por un nombre y por un dominio.

Un dominio indica qué valores pueden ser asumidos por una columna de la relación, o sea, es el conjunto de valores sobre los que se define el tipo de un atributo.

Normalmente todas las tablas deben tener una clave principal definida. Una clave principal es una columna (o combinación de columnas) que permite identificar de forma inequívoca cada fila de tabla, por lo que no pueden haber dos filas con el mismo valor en la columna definida como clave principal.

Una clave foránea es una columna (o combinación de columnas) que contiene un valor que hace referencia a una fila de otra tabla (en algunos casos puede ser la misma tabla).

Una tabla tiene una única clave primaria. Una tabla puede contener cero o más claves foráneas.

Cuando se define una columna como clave principal, ninguna fila de la tabla puede contener un valor nulo en esa columna ni tampoco se pueden repetir valores en la columna.

Cuando se define una columna como clave foránea, las filas de la tabla pueden contener en esa columna o bien el valor nulo, o bien un valor que existe en la otra tabla. Eso es lo que se denomina integridad referencial que consiste en que los datos que referencian otros (clave foránea) deben ser correctos. [17]

### 2.2.2 Bases de Datos XML.

De acuerdo al tipo de almacenamiento y gestión de documentos XML, se encuentran las BD XML nativas y las Extensiones de Bases de Datos relacionales (BDR) para XML.

#### Nativas

Una base de datos nativa XML es aquella diseñada para almacenar documentos XML. Define un modelo lógico para un documento XML, almacena y recupera documentos de acuerdo al modelo.

Tiene características de otro tipo de base de datos, como son: soportar transacciones, seguridad, acceso multi-usuario, programación de APIs, lenguajes de consulta, etc. Los lenguajes de consulta que soportan son los mismos que soporta XML, como por ejemplo, XPath y XQuery.

Tiene una variedad de estrategias para actualizar y eliminar documentos, desde reemplazar o borrar los documentos ya existentes hasta modificaciones a través de un árbol DOM a lenguajes que especifican como modificar fragmentos de un documento.

Algunas de estas bases pueden incluir datos remotos en los documentos almacenados.

Dependiendo de la arquitectura que utilicen las bases de datos XML, pueden dividirse en base de datos: basadas en texto y basadas en el modelo.

Las bases de datos basadas en texto almacenan el documento XML como texto, almacenando así una copia idéntica del dato. Son indexadas, lo cual permite la ejecución de las consultas ir a un punto del documento XML.

Las bases de datos basadas en el modelo construyen un modelo de objetos interno del documento XML y lo almacenan. El almacenamiento del modelo depende de la base de datos. Algunas bases de datos lo almacenan en bases de datos relacionales, otras en bases de datos orientados a objetos. [18]

### Extensiones de Bases de Datos relacionales para XML

Cuentan con diferentes aproximaciones:

- Almacenamiento no estructurado

Almacenamiento del documento XML directamente en formato de texto en un CLOB (Character Large OBject).

Lo soportan la mayoría de los Sistemas de Gestión de Bases de Datos relacionales. Incluyen además funciones para acceder el contenido de los documentos del SQL.

- Almacenamiento estructurado

Un metamodelo detallado de documentos XML capaz de representar árboles de nodos de documentos XML arbitrarios se construye utilizando primitivas de modelado del SGBD convencional que hay por debajo.

Los contenidos de los documentos XML se pueden consultar utilizando las facilidades proporcionadas por el SGBD.

- Mapeo

El contenido de documentos XML se mapea en esquemas BD específicamente diseñado para este contenido.

Permite utilizar las capacidades de modelado de los Sistemas de Gestión de Bases de Datos convencionales para la representación eficiente y adecuada del contenido de los documentos.

Existe gran cantidad de herramientas y formalismos para la especificación del mapeo entre un formato XML y un esquema de BD.

Existe mucha investigación respecto a la generación automática de esquemas de bases de datos relacionales a partir de documentos XML y el mapeo automático entre los mismos. [19]

## Ejemplos de Bases de Datos con extensiones para XML

### PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD. [20]

- Se inicia en la Universidad de Berkeley en 1977 bajo el nombre Ingres como un ejercicio de aplicación de las teorías de las RDBMS.
- 1986, cambia su nombre a Postgres con el objetivo de aplicar los conceptos de Objetos Relacionales.
- 1995, cambia su nombre a Postgres95 que luego derivaría a PostgreSQL
- 1996, el proyecto se integra al mundo del Open Source inicia en la versión 6.0
- 2000, se comienza a implementar el soporte de Ipv6
- 2004, PostgreSQL 8.0, adopción en el mundo comercial, se le califico como la 5ta DBMS mas popular en USA.
- 2005 Julio, PostgreSQL pasó el test de Coverty Inspected encontrando sólo 20 errores en 775,000 líneas de código.
- 2006 Versión 8.1.4

Características de PostgreSQL:

- PostgreSQL está bajo licencia BSD (Berkeley Software Distribution). Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público.
  La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.
- PostgresSQL es Full ACID compliant (Atomicity, Consistency, Isolation and Durability)

  - Atomicidad (Indivisible) es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
  - Consistencia es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
  - Aislamiento es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error.
  - Durabilidad es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

- Corre en casi todos los principales sistemas operativos: Linux, Unix, Mac OS, Windows, etc.
- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.
- Comunidades muy activas, varias comunidades en castellano.
- Bajo “Costo de Propiedad Total” (TCO) y rápido “Retorno de la Inversión Inicial” (ROI)
- Altamente adaptable a las necesidades del cliente.
- Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, etc.
- Drivers: ODBC, JDBC, .Net, etc.
- Soporte de triggers, procedimientos almacenados – funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.
- Soporte de tipos de datos de SQL92 y SQL99.
- Soporte de protocolo de comunicación encriptado por SSL
- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.
- Utilidades para limpieza de la base de datos (Vacuum)
- Utilidades para análisis y optimización de Querys.
- Almacenaje especial para tipos de datos grandes (TOAST)
- Varios tipos de índices
- El mejor Sistema Operativo para correr PostgreSQL es *BSD y Unix, por su sistema dinámico de I/O (más eficiente que en otros Sistemas Operativos).

Una de las funcionalidades a destacar es el soporte al estándar SQL/XML de ANSI SQL: 2003, incluyendo verificaciones de sintaxis, nuevos operadores, el tipo de datos XML y consultas XPath. Con esto se puede operar con documentos XML como son: documentos de texto estructurado. No tener que tratar un XML como un simple texto nos permitirá aprovecharnos de las ventajas propias de este tipo de documentos, como poder comprobar fácilmente si un valor de tipo XML está bien formado. [21]

**SQL Server**

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado el modelo relacional. Sus lenguajes de consultas son T-SQL y ANSI SQL. [22]

Características de SQL Server:

- Soporte de transacciones y procedimientos almacenados.
- Escalabilidad, estabilidad y seguridad.

- Incluye entorno gráfico de administración, que permite el uso de comandos DDL (Lenguaje de definición de datos) y DML (Lenguaje de Manipulación de Datos) gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.

Las aplicaciones trabajan tanto con datos de formato relacional como un formato XML.

XML se adapta especialmente bien al intercambio de datos entre aplicaciones, a la gestión de documentos, al envío de mensajes (SOAP), etc.

Se pueden crear tablas relacionales que, además de las columnas de los tipos habituales, contengan una o más de tipo XML.

Las columnas de tipo XML utilizan formato binario (blob) para almacenar la información en la base de datos relacional, de forma que el documento XML se conserva en el estado correcto. De hecho, el espacio para cada dato XML está limitado a 2 GB. Además, el documento XML no debe estar estructurado con una jerarquía de más de 128 niveles.

Para almacenar datos directamente en formato XML, SQL Server 2005 evita tener que hacer un trabajo largo y pesado para definir las correspondencias entre el formato XML y la estructura relacional con la que están organizada la información en las bases de datos. Este tipo dispone de los métodos especiales para trabajar con los datos. Estos métodos se basan en XQuery, un subconjunto de XML específico para consultas.

Para satisfacer las exigencias del consorcio W3C, se puede asociar una colección de esquemas a una columna de tipo XML. Los datos almacenados en la columna deberán respetar las restricciones del esquema. Esta columna se denominará entonces XML con tipo; si no, se tratará de una columna XML sin tipo.

XML sin tipo

El tipo XML tal y como está definido en SQL Server 2005 respeta la definición normalizada por el estándar ISO SQL-2003, es decir que trabaja con documentos XML 1.0 bien formados.

Sin embargo, la columna que utilice un tipo XML si tipo, es decir no basado en un esquema XML, sólo contendrá datos conformes con un documento XML 1.0 bien formado o un fragmento XML.

Este método de funcionamiento es el más flexible, pero cuando se dispone de un esquema XML es preferible recurrir a XML con tipo.

Debe utilizarse cuando no se disponga de esquema XML, o cuando haya varios esquemas XML asociados a orígenes externos de datos.

XML con tipo

Si los datos descritos en una colección de esquemas XML irán en una columna, se puede asociar esta colección de esquemas XML a la columna. Entablando esta asociación en el nivel de la definición de la columna todos los datos escritos en ella deben respetar un esquema XML asociado. Entonces se dice que la columna está definida en XML con tipo.

Los esquemas XML actúan como una restricción de integridad que garantiza una estructura claramente identificada para todos los datos presentes en esta columna. Las actualizaciones de datos XML se controlan mejor y más rápido en el proceso de ejecución de las consultas.

La colección de esquemas XML debe estar creada antes de que se pueda hacer referencia a ella en una columna XML. [23]

Características de la funcionalidad XML en SQL Server:

- Posibilidad de acceder a SQL Server vía HTTP.
- Soporte para esquemas XRD(XML-Data Reduced).
- Recuperación y escritura de datos XML, usando la cláusula FOR XML de la sentencia SELECT de T-SQL.
- Uso de Updategram, los cuales permiten la manipulación de datos basada en transacciones utilizando XML.
- XML Bulk load dirigido a trasladar cantidades enormes de datos basados en XML a SQL Server.
- Posibilidad de definir mediante un parámetro el retorno de datos binarios desde un origen de datos SQL Server.
- Funcionalidad totalmente adaptada al estándar XSD, es decir, en conformidad con la definición de esquema W3C conocido como XML Schema Definition (XSD).
- El formato del lado del cliente permite que se formatee el XML de los conjuntos de filas SQL Server en el servidor IIS en lugar de en el servidor de base de datos.

### Oracle

Oracle es la primera Base de Datos Diseñada para Grid Computing, es un sistema de gestión de base de datos relacional fabricado por Oracle Corporation.

- Oracle es básicamente una herramienta cliente/servidor para la gestión de bases de datos la gran potencia que tiene y su elevado precio hace que sólo se vea en empresas muy grandes y multinacionales, por norma general.
- Ha sido diseñada para que pueda controlar y gestionar grandes volúmenes de contenidos no estructurados en un único repositorio con el objetivo de reducir costos y riesgos asociados a la pérdida de información.
- Oracle corre en computadoras personales, microcomputadoras, mainframes y computadoras con procesamiento paralelo masivo.
- Corre automáticamente en más de 80 arquitecturas de hardware y software distinto sin tener la necesidad de cambiar una sola línea de código.
- Se puede almacenar, en forma independiente, funciones y procedimientos sin tener que escribirlo repetidamente para cada forma, y pudiendo compilarlos independientemente de las formas que lo usen. Pero, además, las funciones y procedimientos se pueden agrupar en un paquete para compartir definiciones, variables globales, constantes, cursores y excepciones, así como garantizar y revocar los permisos a nivel de paquete.
- Oracle posee igual interacción en todas las plataformas (Windows, Unix Macintosh y Mainframes). Esto porque más del 80% de los códigos internos de Oracle son iguales a los establecidos en todas las plataformas de Sistemas Operativos. [24]

Una Base de Datos Oracle tiene una estructura física y una estructura lógica; la estructura física se corresponde a los ficheros del sistema operativo y la estructura lpogica estpa formada por los espacios de tablas (tablespace) y los objetos de un esquema de Bases de Datos. [25]

Oracle Database, con Oracle XML DB, es un híbrido de bases de datos para XML y datos relacionales. Oracle XML DB es una característica de la base de datos Oracle, la cual proporciona almacenamiento nativo XML. Oracle Database 11g introduce un nuevo modelo de almacenamiento – incluye el soporte a formatos binarios de XML y nuevas capacidades tales como índices, mayor soporte para XQuery. [18]

## Ejemplos de Bases de Datos Nativas XML

### eXist

Se trata de una versión ‘Open Source’ de una Base de Datos XML nativa.

Cubre muchas de las características básicas de las bases de datos nativas XML, así como un número de técnicas avanzadas de búsqueda de palabras claves en texto, consultas en la proximidad de términos y patrones de búsqueda basadas en expresiones regulares. La base de datos está completamente escrita en Java.

La aplicación de base de datos puede ejecutarse como servidor estandalone, embebido dentro de una aplicación o en una conexión con un contenedor de servlets. Esta tres alternativas se ejecutan en “hilos” independientes y soportan operaciones concurrentes por múltiples usuarios.

Entre sus características más destacables son:

- Soporta Xquery 1.0 como lenguaje de consulta, indexación y procesado de la información. También soporta Xupdate pero de una forma limitada.
- Proporciona almacenamiento de documentos XML sin la necesidad de la definición de un esquema o DTD. En otras palabras, se permite que los documentos sólo estén bien formados.
- Dentro de la base de datos, los documentos son administrados en colecciones jerárquicas. Los documentos pueden ser almacenados arbitrariamente dentro de la misma colección.
- Implementa un API para la invocación desde aplicaciones Java.
- Implementa extensiones para el soporte de HTTP, manipulación y transformaciones XSL.
- Incorpora proyectos “extra” para la explotación de los datos que almacene.
- Incorpora un framework del lado del servidor para la creación de aplicaciones web, basadas totalmente en XML.
- Desde un punto de vista económico, resulta un producto muy ventajoso al sustentarse sobre licencias de tipo: GNU LGPL. [26]

**XIndice**

Xindice es la continuación de un proyecto anterior llamado dbXML Core. El código fue donado por el grupo dbXML (http://www.dbxml.com) a Apache Software Foundation en Diciembre de 2001.

Es un servidor de base de datos diseñada para administrar un gran número de de documentos pequeños. Los documentos son almacenados en colecciones y el servidor proporciona la capacidad para consultar colecciones. El servidor es ligero modular y adecuado para estar embebido en aplicaciones personalizadas o ejecutarse como una base de datos independiente. Xindice está escrito en Java.

El servidor actualmente proporciona soporte para almacenar documentos XML bien formados, o sea, no tiene un esquema que obligue a un documento a estar dentro de una colección. Esto hace a Xindice una base de datos semiestructurada y proporciona gran flexibilidad para almacenar los datos.

Entre sus características más destacables son:

- Utiliza Xpath como lenguaje de consulta y XML:DB XUpdate como lenguaje de actualización en la BDD.
- Proporciona una implementación de XML:DB API para la invocación desde desarrollos Java.
- Para mejorar el rendimiento de las consultas sobre un gran número de documentos se pueden definir índices sobre elementos y valores de atributos. Esto puede acelerar el tiempo de la respuesta de las consultas.
- Mediante XML-RPC plugin es posible acceder a Xindice desde otros entornos o lenguajes.
- Los estándares de XML que soporta Xindice son los recomendados por W3C (World Wide Web Consortium).
- Es una base de datos muy ventajosa desde el punto de vista económico ya que se sustenta sobre el tipo de licencia: Apache Software License

Xindice permite el manejo de un lenguaje de consultas para recuperar documentos o datos de documentos. Para ello se usa el lenguaje de consulta XPATH (hay otros como XQuery, XQL, XML-QL, QUILT, etc.), añadiéndose nuevas funcionalidades como el lenguaje de actualización / modificación de datos. [27]

**Xperanto**

Es un proyecto comandado por IBM Almaden Research Center su meta es ser un servidor como capa intermedia que mantiene la publicación de datos XML para esta clase de usuarios [28].

Internamente Xperanto traduce las consultas XML en SQL, propone sistemas de bases de datos, recibe las respuestas de las consultas y después traduce sus resultados en términos de XML.

Arquitectura de XPEERANTO

Está organizado principalmente en cuatro componentes los cuales son: Traducir consultas, servicio de vistas XML, generador de esquemas XML, y las etiquetas XML. El núcleo de XPERANTO es el componente traducción de consultas.
La figura muestra los principales roles que juegan cada uno de los subcomponentes.

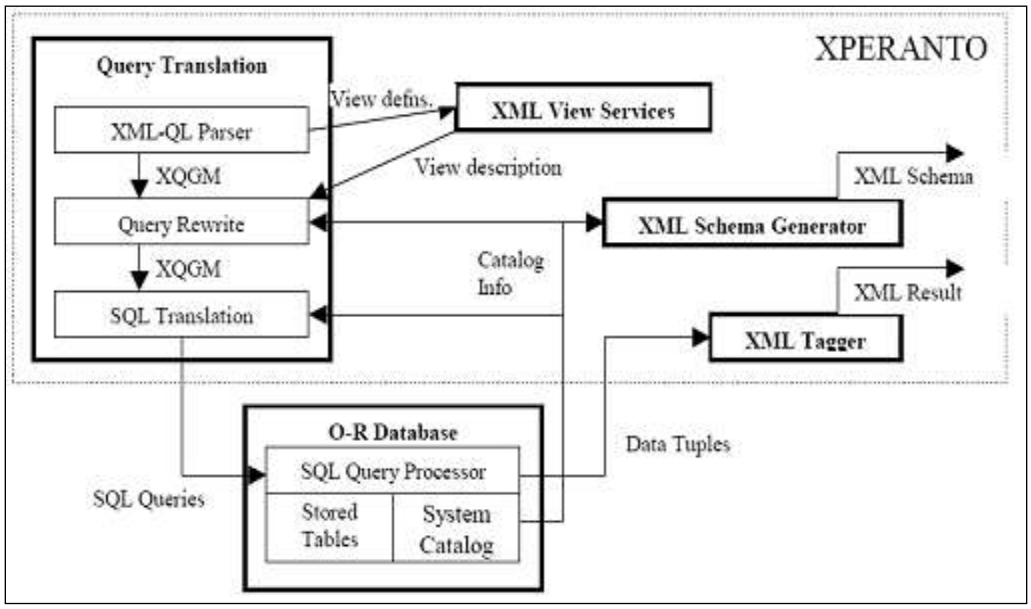



Figura 2.2-1 Arquitectura de Xperanto

XML-QL Parser: toma una consulta XML-QL y genera XQGM (XML Query Graph Model) – un lenguaje neutral intermedio para representar consultas XML. XQGM XPERANTO se escuda de los detalles de un particular lenguaje de consulta XML. De esta manera XPERANTO puede adaptarse fácilmente al lenguaje de consulta estándar XML cuando sea disponible.

Query Rewrite. Toma el XQGM de una consulta resuelve las referencias de vista, ejecuta la composición de las vistas XML y produce una representación semántica de SQGM equivalente de la consulta. También consulta los catálogos del sistema de base de datos en caso de que la consulta del usuario este sobre los modelos de datos relacional y el modelo meta-datos

SQL Translation: Traduce XQGM en sentencias SQL. Este subcomponente hace uso del catalogo de información del sistema de bases de datos para ejecutar la comprobación de tipos.

XML View Services: Servidor como almacenamiento y recuperación de interfaces para definir vistas XML-QL. Cuando las vistas son definidas, ellas se están almacenando en una tabla dedicada. Después pueden ser recuperadas.

XML Schema Generator: Toma información del catálogo de la base de datos y produce esquemas de información de resultados de consultas y vistas XML.

XML Tagger: Convierte el resultado de la consulta SQL en documentos XML estructurados.

| SGBD | Usabilidad | Sólo acepta documentos válidos | Documentación | Licencia |
|---|---|---|---|---|
| PostgreSQL | Buena | No | Muy Buena | Libre |
| SQL Server | Buena | Opcional | Muy Buena | Comercial |
| Oracle | Muy Buena | Si | Muy Buena | Comercial |
| eXist | Muy buena | No | Buena | Código abierto |
| Xindice | Mala | No | Regular | Código abierto |
| Xperanto | - | No | Mala | Comercial |

Para el desarrollo de la parte relacional se consideró a PostgreSQL, ya que es de licencia libre, además de que esta parte de la aplicación no es compleja, y no se requiere de un manejador de base de datos tan robusto.

Para el desarrollo de la aplicación en la parte de los documentos XML se seleccionó eXist como manejador de la base de datos, ya que es una tecnología de código abierto, además es recomendable pues al ser una base de datos nativa, el acceso y almacenamiento de información es en formato XML, sin tener la necesidad de incorporar código adicional ni de traducirlos a una estructura relacional o de objeto. Además, eXist ofrece una interfaz de usuario fácil de utilizar.

## 2.3. Proceso Unificado

El Proceso Unificado [8] es un proceso de desarrollo de software que es el conjunto de actividades necesarias para transformar los requerimientos de un usuario en un sistema de software. El Proceso Unificado es un marco de trabajo de procesos genérico que puede especializase para cada clase de sistemas de software, para diferentes áreas de aplicación, tipos de organizaciones, niveles de competencia y tamaños de proyectos.

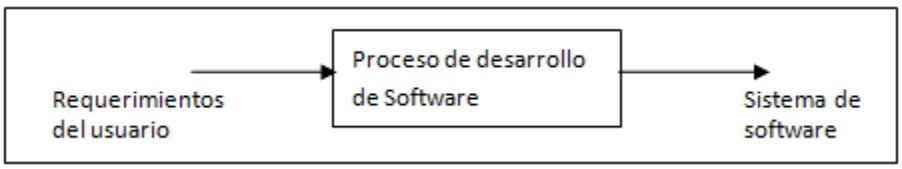



Figura 2.3-1 Proceso de desarrollo de software, de [8]

Los aspectos que definen al Proceso Unificado son: dirigido por casos de uso, centrado en la arquitectura, y es iterativo e incremental.

- El Proceso Unificado está dirigido por casos de uso.

El término usuario representa alguien o algo (como otro sistema fuera del sistema en consideración) que interactúa con el sistema que se está desarrollando. En respuesta, el sistema lleva a cabo una secuencia de acciones que proporcionan al usuario un resultado importante. Una interacción de este tipo es un caso de uso; que es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requerimientos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso, el cual describe la funcionalidad total del sistema. Aunque los casos de uso guían el proceso de desarrollo (diseño, implementación y prueba), no se desarrollan aisladamente. Se desarrollan a la vez que la arquitectura del sistema. Es decir, los casos de uso guían la arquitectura del sistema y la arquitectura del sistema influye en la selección de los casos de uso. Por lo tanto, la arquitectura del sistema y los casos de uso maduran según avanza el ciclo de desarrollo.

- El Proceso Unificado está centrado en la arquitectura.

La arquitectura en un sistema de software se describe mediante diferentes vistas del sistema en construcción. Incluye los aspectos estáticos y dinámicos más significativos del sistema.

La arquitectura se ve influenciada por distintos factores, como la plataforma en la que tiene que funcionar el software, los bloques de construcción reutilizables de que se dispone, consideraciones de implementación, y requerimiento no funcionales.

La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado. El proceso ayuda al arquitecto a centrarse en los objetivos adecuados, como la comprensibilidad, la capacidad de adaptación al cambio, y la reutilización.

Cada producto tiene tanto una función como una forma. La función corresponde a los casos de uso y la forma a la arquitectura. Debe haber interacción entre los casos de uso y la arquitectura. Por un lado, los casos de uso deben entrar en la arquitectura cuando se llevan a cabo, y por el otro, la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, ahora y en el futuro. Finalmente, el arquitecto:

- Crea un esquema en borrador de la arquitectura, comenzando por la que no es específica de los casos de uso.
- Después, el arquitecto trabaja con un subconjunto de los casos de uso especificados. Cada caso de uso seleccionado se especifica en detalle y se realiza en términos de subsistemas y componentes.
- A medida que los casos de uso se especifican y maduran, se descubre más de la arquitectura. Esto a su vez, lleva a la maduración de más casos de uso.

Este proceso continúa hasta que se considere que la arquitectura es estable.

- El Proceso Unificado es iterativo e incremental

El desarrollo de un producto de software lleva mucho esfuerzo que puede tardar entre varios meses hasta años. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencias a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Las iteraciones deben estar controladas, es decir, deben seleccionarse y ejecutarse de una forma planificada.

La selección de lo que se implementará en una iteración se basa en dos factores:

- La iteración trata un grupo de de casos de uso que juntos amplían la utilidad del producto desarrollado hasta ahora.
- La iteración trata los riesgos más importantes.

Las iteraciones sucesivas se construyen sobre los artefactos de desarrollo tal como quedaron al final de la última iteración. Los miniproyectos comienzan con los casos de uso y continúan a través del análisis, diseño, implementación y prueba, que termina convirtiendo en código ejecutable los casos de uso que se desarrollaban en la iteración.

En cada iteración, los desarrolladores identifican y especifican los casos de uso relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, implementan el diseño mediante componentes, y verifican que los componentes satisfacen los casos de uso. Si una iteración cumple con sus objetivos el desarrollo continúa con la siguiente iteración; si no, se deben revisar las decisiones previas e intentar una nueva propuesta.

La vida del Proceso Unificado.

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes.
Cada ciclo consta de cuatro fases: inicio, elaboración, construcción y transición. Cada fase se subdivide a su vez en iteraciones.

Cada ciclo produce una nueva versión del sistema, y cada versión es un producto preparado para su entrega. Consta de un cuerpo de código fuente incluido en componentes que puede compilarse y ejecutarse, además de manuales y productos asociados.

El producto terminado incluye los requerimientos, casos de uso, especificaciones no funcionales y casos de prueba, así como, el modelo de la arquitectura y el modelo visual (artefactos modelados con UML)

El Proceso Unificado consta de cuatro fases como se muestra en la figura 2.3-2, en la columna de la izquierda se muestran los flujos de trabajo –requisitos, análisis, diseño, implementación y prueba. Las curvas son una aproximación de hasta dónde se llevan a cabo los flujos de trabajo en cada fase.

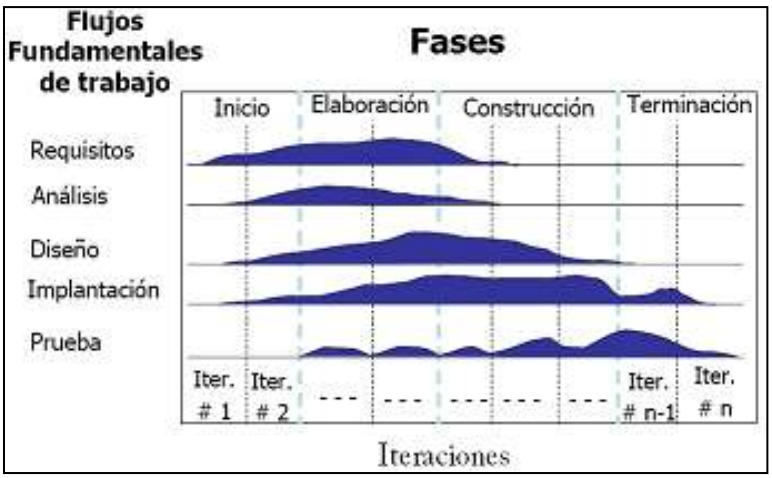



Figura 2.3-2 Fases del Proceso Unificado, de [8]

La fase de Inicio.

● Se desarrolla una descripción del producto a partir de una buena idea y se presenta el análisis de negocio para el producto.
● Se delimita el alcance del sistema propuesto
● Se describe la arquitectura candidata del sistema
● Se identifican los riesgos críticos, aquellos que afectan la capacidad de construir el sistema, y determinar si hay manera de mitigarlos
● Se demuestra que el sistema es capaz de resolver el problema o soportar el objetivo de negocio construyendo una prueba de concepto.

La fase de Elaboración.

● El producto principal es una arquitectura estable para guiar el sistema por su vida futura, y planeación de la fase de construcción con fidelidad

● Se crea una arquitectura que cubre la funcionalidad y características importantes
● Se identifican los riesgos significativos que puedan afectar planes, costos, y programas de fases
● Se especifican los niveles a lograr atributos de calidad y tiempos de respuesta
● Se captura el 80% de los casos de uso suficientes para planear la construcción

La fase de Construcción.

● El objetivo general es la capacidad operacional inicial, significa producto listo para prueba beta.
● Se extiende la identificación, descripción y realización de caso de uso
● Se termina el análisis, diseño, implementación y prueba
● Se mantiene la integridad de la arquitectura
● Se monitorean los riesgos críticos y significativos de fases anteriores y si se presentan, mitigarlos

La fase de Transición.

● Se empieza con la liberación beta
● Se empiezan actividades de preparación, como preparar instalación
● Se advierte al cliente sobre actualizaciones (Sistema operativo, hardware, protocolos de comunicación, etc.)
● Se preparan los manuales y documentación
● Se ajusta el software para operar bajo parámetros actuales del ambiente del usuario
● Se corrigen defectos después de pruebas beta
● Se modifica el software en presencia de problemas no previstos

# Capítulo 3

# Planteamiento del problema y propuesta de solución

Se requiere diseñar un depósito de buenas prácticas para las tecnologías de información, el cual tendrá un conjunto ordenado y organizado de archivos XML. Se requiere que el depósito cuente con diferentes tipos de búsqueda para tener acceso a las buenas prácticas, y además, que se puedan ir almacenando las nuevas prácticas al depósito.

## 3.1 Definición del problema

Necesidades del Negocio:

- La comunidad dedicada a las tecnologías de información muchas veces no cuenta con información acerca de las buenas prácticas, y generalmente trabajamos de más, empleando más recursos, tiempo y esfuerzo, cuando se puede facilitar utilizando las buenas prácticas.
- Además, si no se tienen las buenas prácticas disponibles, pues nadie se enterará de que existen y por lo tanto nadie las consultará.
- Tener buenas prácticas mejora la calidad.

Propósitos del proyecto:

Proveer un sistema web que facilite:

- Incluir las buenas prácticas a través de la generación de un depósito.
- La búsqueda de distintas formas de las buenas prácticas

Objetivos del proyecto:

- Permitir que los ingenieros de software puedan consultar las buenas prácticas, sin poder modificarlas
- Desarrollar un depósito de buenas prácticas, en el cual se puedan incluir y consultar las buenas prácticas.

Alcance:

El sistema sólo tendrá archivos XML y sólo el administrador podrá incluir las nuevas buenas prácticas a la base de datos, pues siendo él el experto primero las revisará y dependiendo de la información de la buena práctica decidirá si realmente es, o no adecuada para publicarla.

El ingeniero de software únicamente podrá consultar la buenas prácticas sin poderlas modificar, ya que primero deben pasar por una previa revisión, esto porque posiblemente alguien querrá subir algo que realmente no es una buena práctica. Las buenas prácticas únicamente serán propuestas mediante un formulario web.

Restricciones:

- Sólo el administrador podrá ingresar nuevas prácticas.
- Se seguirá el Proceso Unificado para el desarrollo del sistema de captura y consulta de buenas prácticas.

## 3.2 Flujo de Requerimientos

Flujo de trabajo fundamental cuyo propósito esencial es orientar el desarrollo hacia el sistema correcto. Esto se lleva a cabo mediante la descripción de los requisitos del sistema de forma tal que se pueda llegar a un acuerdo entre el cliente (incluyendo los usuarios) y los desarrolladores del sistema, acerca de lo que el sistema debe hacer y lo que no. [10]

### 3.2.1 Diagrama general de casos de uso

Para el desarrollo del depósito se utilizará el Proceso Unificado y utiliza al Lenguaje Unificado de Modelado, el cual se sostiene de tres representaciones -casos de uso, arquitectura y desarrollo iterativo e incremental.

Empezaremos con los casos de uso.
Los casos de uso son piezas de funcionalidad que el sistema ofrece para dar un resultado de valor para los actores [10]; los cuáles asumen roles cuando interactúan con el sistema, la comunicación con el sistema es mediante el envío y recepción de mensajes hacia y desde el sistema según éste lleva a cabo los casos de uso.

Especifican un comportamiento deseado, no imponen cómo se llevará a cabo ese comportamiento.

De acuerdo a los roles que se marcan en el punto 1.6 Mapeo de Roles se pueden especificar a nuestros tres actores dentro de estos roles.

¿Quién es el Ingeniero de Software y el ponente? Es la persona que consultará las buenas prácticas, y es cualquier persona que se encuentra dentro del área de sistemas y tecnologías de información. Por lo que caen dentro de cualquier rol, ya que todos los roles fueron especificados para el desarrollo, la mejora y mantenimiento de software y sistemas.

El administrador es igual pero con una amplia experiencia dentro de este campo.

La figura 3.2.1 -1 muestra el diagrama de casos de uso general del depósito:

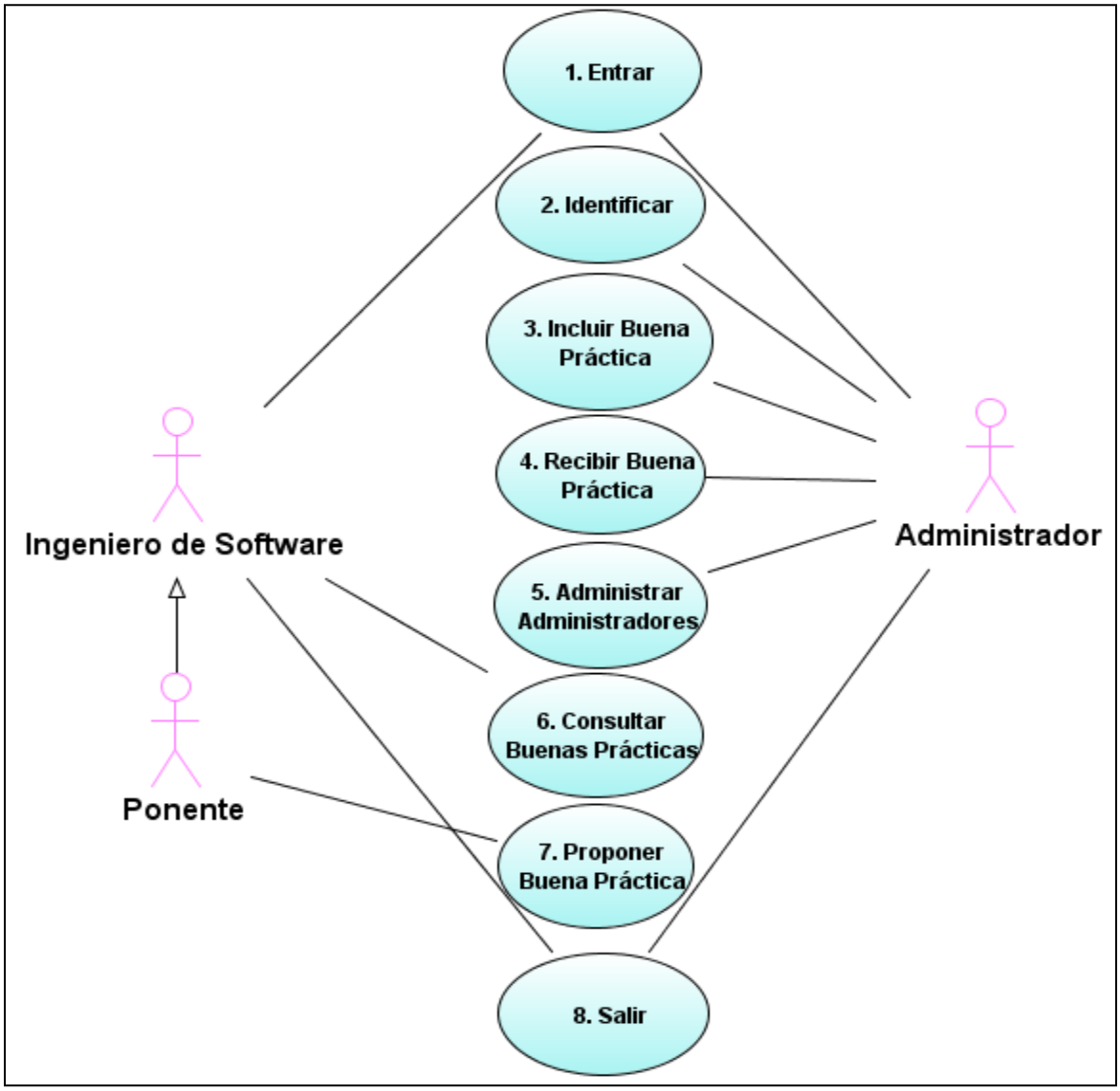



Figura 3.2.1-1 Diagrama de casos de uso general

Este diagrama se compone de ocho casos de uso, los cuales están repartidos en tres diferentes actores; el administrador, ponente y el ingeniero de software. Los casos de uso 1 y 8 pueden ser realizados por los tres actores. Además, el administrador se identificará para acceder a ciertas funcionalidades destinadas para él, como son administrar las buenas prácticas y a los administradores del depósito, y finalmente podrá recibir las buenas prácticas propuestas.

Existe una generalización entre el ponente y el ingeniero de software ya que ambos podrán consultar las buenas prácticas, pero además, el ponente podrá proponer buenas prácticas.

## 3.3 Detalle de los casos de uso, Prototipo y Plan de Pruebas

El detallado es importante para visualizar, especificar y documentar el comportamiento de un elemento, facilita que el sistema o clases sean factibles y entendibles, al presentar una vista externa de cómo pueden utilizarse estos elementos en un contexto dado.
Para no excederse en este punto, sólo se mostrarán los detalles de los casos de uso, prototipo y plan de pruebas de aquellas operaciones que se ha creído tienen mayor relevancia. Los demás se podrán visualizar en el Anexo B1.

### 3.3.1 Ingeniero de Software

**Caso de uso 6:** Consultar Buenas Prácticas

**Actor:** Ingeniero de Software


**Descripción:** El ingeniero de software puede consultar las buenas prácticas

**Precondiciones:**
- El ingeniero de software está dentro del sistema
- El ingeniero de software desea consultar las buenas prácticas

**Flujo:**

| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Elige la opción “Consultar Buenas Prácticas” de la página principal | 2 | Muestra el formulario para la búsqueda de la buena práctica a consultar | - |
| 3 | Elige el criterio búsqueda, introduce los datos requeridos en el formulario y presiona el botón “Buscar”. | 4 | Realiza la búsqueda en la base de datos. | E1, E2 |

| | | | | |
|---|---|---|---|---|
| - | - | 5 | Muestra el listado con los resultados de la consulta. | - |
| 6 | Selecciona el registro de la buena práctica que desea consultar y presiona “Consultar”. | 7 | Muestra en la pantalla la buena práctica a consultar. | E2 |
| 8 | Presiona el botón “Aceptar”. | 9 | Muestra la pantalla inicial. | - |

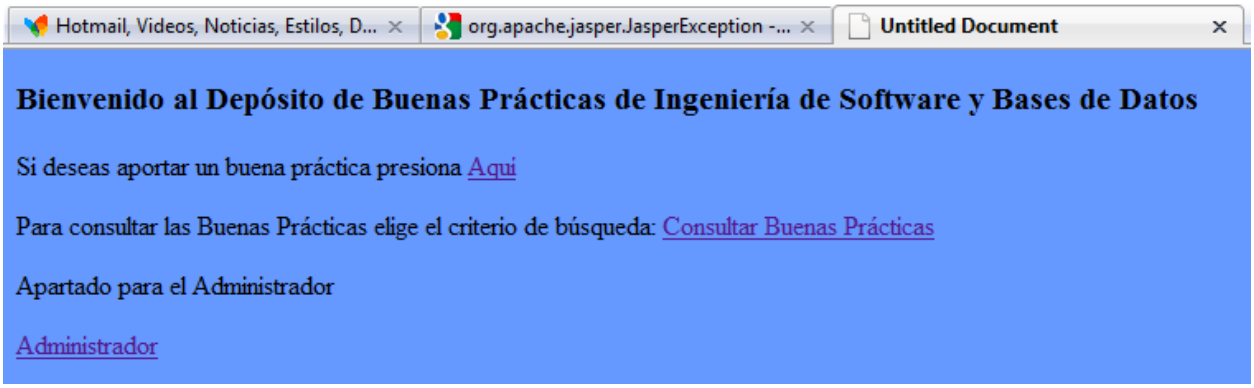
**Excepciones:**

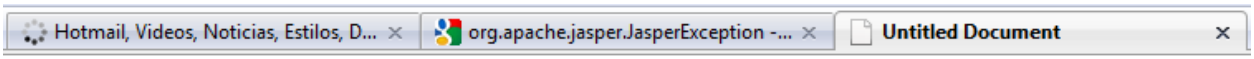
| ID | NOMBRE | ACCIÓN |
|---|---|---|
| E1 | Datos faltantes o incorrectos | El sistema presenta un mensaje de error solicitando que se vuelva a introducir los datos. |
| E2 | Fallo en la conexión con la BD | El sistema presenta un mensaje de error solicitando se intente nuevamente más tarde. |

**Poscondiciones**:

- El ingeniero de software ha consultado las buenas prácticas
- El sistema despliega las buenas prácticas

**Prototipo**






**Bienvenido al Depósito de Buenas Prácticas de Ingeniería de Software y Bases de Datos**

| Elige el criterio de búsqueda | |
|---|---|
| Área: | Ingeniería de Software |
| Título: | |
| Problema: | |
| Autor: | |
| Buscar Limpiar | |

Cancelar

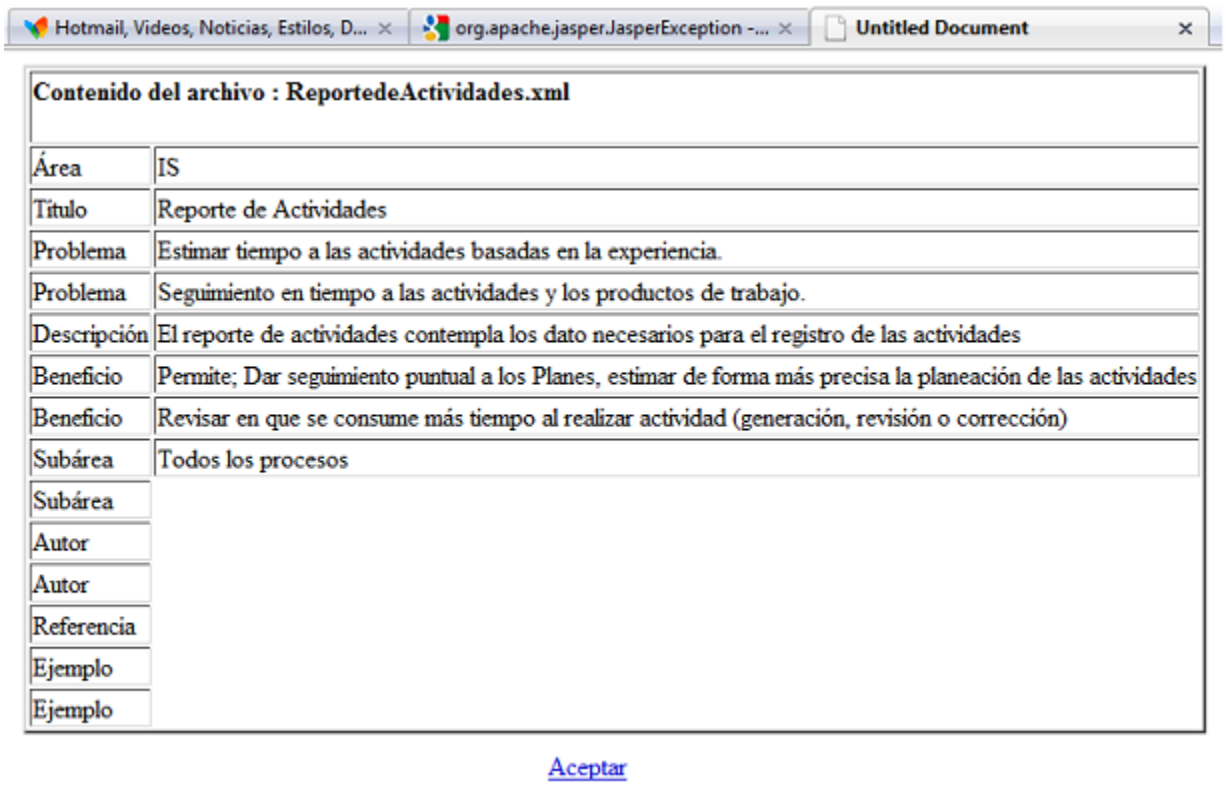

## Ingeniero de Software Buenas Prácticas

Area de busqueda IS

| No. | Título |
|---|---|
| 1 | Reporte de Actividades |
| 2 | casa |
| 3 | mio |

Se encontraron 3 registros

Cancelar



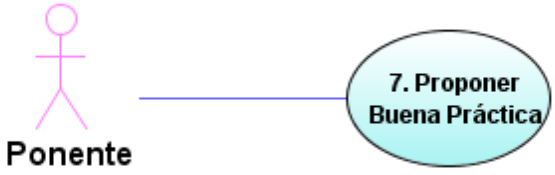

**Plan de Prueba**

| Caso de Uso | Entradas | Resultados Esperados |
|---|---|---|
| 6 | Se introducen criterios de búsqueda adecuados y se presiona el botón “Buscar”. | El sistema muestra el listado de resultados. |
| 6 | Se introducen criterios de búsqueda incompletos o incorrectos y se presiona el botón “Buscar”. | El sistema muestra una pantalla solicitando nuevamente los datos. |
| 6 | Selecciona un registro y presiona el botón “Consultar” | El sistema muestra la pantalla del registro. |
| 6 | Selecciona un registro y no presiona el botón “Consultar” | El sistema no cambia el estado. |

### 3.3.2 Ponente

**Caso de uso 7:** Proponer Buena Práctica

**Actor:** Ponente



**Descripción:** El ponente desea proponer una buena práctica

**Precondiciones:**

- El ponente está dentro del sistema
- El ponente tiene una buena práctica y quiera que sea expuesta a todos.

**Flujo:**

| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Elige la opción “Aquí” de la página principal | 2 | Muestra el formulario para ingresar la buena práctica. | - |
| 3 | Registra los datos pedidos y presiona el botón “Enviar”. | 4 | Guarda el archivo en el directorio correspondiente a Buenas Prácticas, que se encuentra en el servidor. Y manda un mensaje de confirmación de que el archivo fue enviado. | E3 |
| 5 | Presiona “Salir” | 6 | Muestra la pantalla inicial. | |

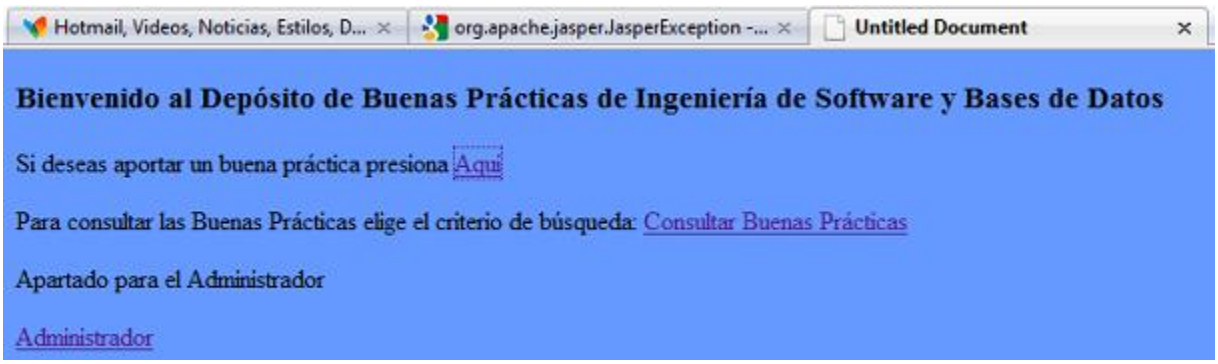
**Excepciones:**

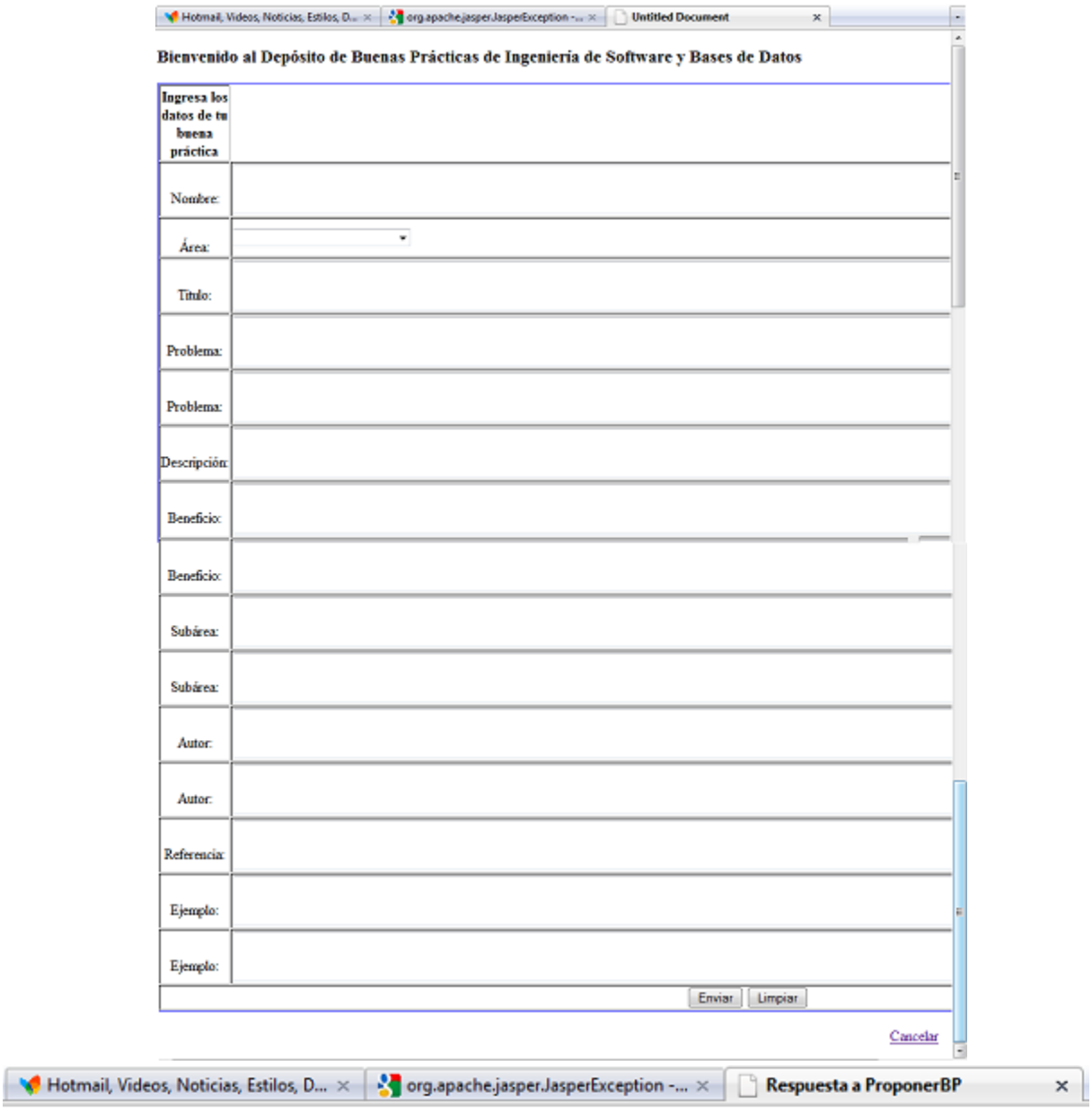
| ID | NOMBRE | ACCIÓN |
|---|---|---|
| E3 | Fallo envío de archivo | El sistema presenta un mensaje de error solicitando se intente nuevamente más tarde. |

**Poscondiciones**:

- El ponente propuso la Buena Práctica al administrador.
- El sistema guardó en el servidor la propuesta.

**Prototipo**





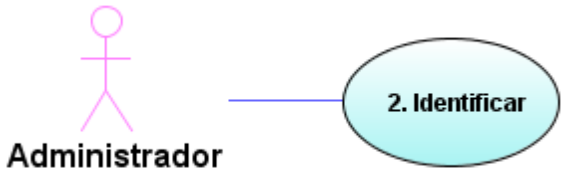

**Plan de Prueba**

| Caso de Uso | Entradas | Resultados Esperados |
|---|---|---|
| 7 | Se introducen los datos requeridos | El sistema muestra pantalla de confirmación. |
| 7 | No se introducen los datos requeridos | Se muestra mensaje, con la petición de llenar los datos faltantes. |

### 3.3.3 Administrador

**Caso de uso 2:** Identificar

**Actor:** Administrador




**Descripción:** El administrador entra a su parte del sistema

**Precondición:** El administrador cuenta con un nombre de usuario y contraseña válidos, se encuentra en la página principal del sistema

**Flujo:**

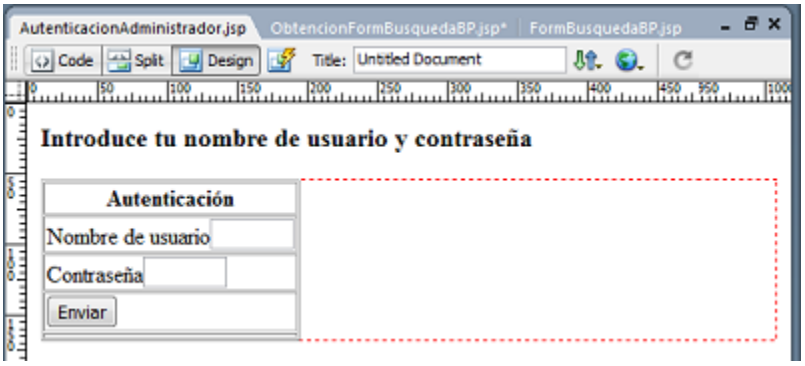
| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Ingresa el nombre de usuario y contraseña | 2 | Verifica que el nombre de usuario y contraseña sean correctos | E1, E2 |
| - | | 3 | Muestra la pantalla con las funcionalidades permitidas para el administrador | - |

**Excepciones:**

| ID | NOMBRE | ACCIÓN |
|---|---|---|
| E1 | Datos faltantes o incorrectos | El sistema presenta un mensaje de error solicitando que se vuelva a introducir los datos. |
| E2 | Fallo en la conexión con la $BD_5$ | El sistema presenta un mensaje de error solicitando se intente nuevamente más tarde. |

**Poscondiciones:** El administrador logró ingresar al sistema y se le otorgaron las funcionalidades otorgadas.

## Prototipo



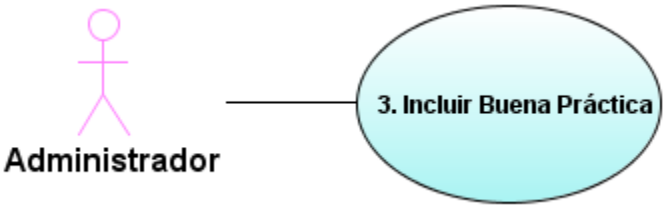


## Plan de prueba

| Caso de Uso | Entradas | Resultados Esperados |
|---|---|---|
| 2 | Se introducen nombre de usuario y contraseña correctos y se presiona el botón “Aceptar”. | El sistema permite el acceso con privilegios adecuados. |
| 2 | Se introduce de manera incorrecta el nombre de usuario o contraseña y se presiona el botón “Aceptar”. | El sistema no permite el acceso. |
| 2 | No se introduce ningún dato y se presiona el botón “Aceptar”. | El sistema no permite el acceso. |

**Caso de uso 3:** Incluir Buena Práctica

**Actor:** Administrador




**Descripción:** El administrador desea incluir o modificar una buena práctica.

**Precondición:** El administrador se ha autentificado y desea incluir o modificar una buena práctica.

**Flujo:**

3. Incluir Buena Práctica

| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Al estar revisando las buenas prácticas elige la opción “Incluir Buena Práctica” | 2 | Envía los datos a la base de datos | E2 |
| - | - | 3 | Envía un mensaje de que la buena práctica fue incluida correctamente | - |

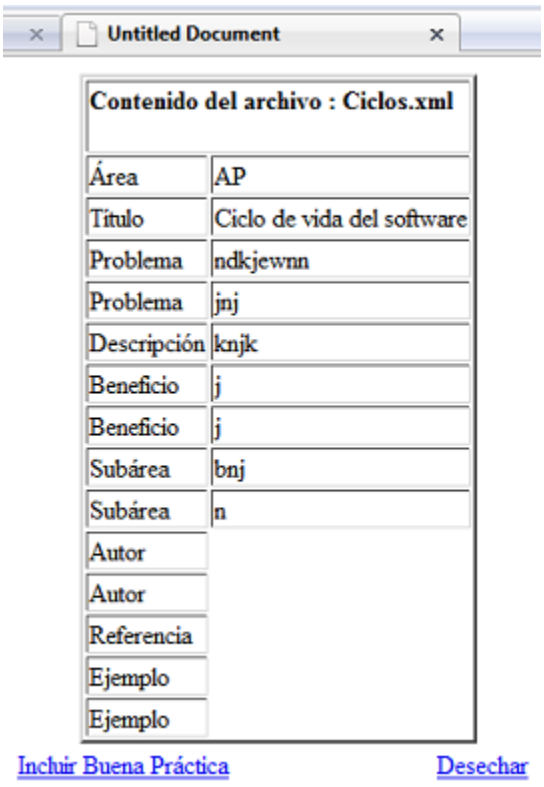
**Excepciones:**

| ID | NOMBRE | ACCIÓN |
|---|---|---|
| E2 | Fallo en la conexión con la BD | El sistema presenta un mensaje de error solicitando se intente nuevamente más tarde. |

**Poscondiciones**:

- El administrador incluyó o modificó una buena práctica.
- El sistema despliega las buenas prácticas.

## Prototipo



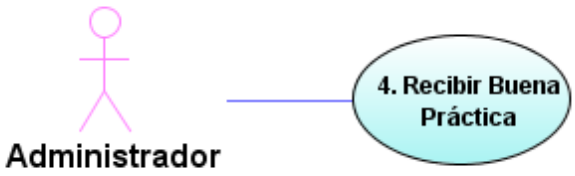


## Plan de Prueba

| Caso de Uso | Entradas | Resultados Esperados |
|---|---|---|
| 3 | Se presiona el botón “Incluir Buena Práctica”. | El sistema envía el archivo a la base de datos |

**Caso de uso 4:** Recibir Buena Práctica

**Actor:** Administrador




**Descripción:** El administrador desea revisar una buena práctica recibida.

**Precondiciones:**

- El administrador se ha autentificado y desea revisar una buena práctica.
- El administrador desea subir una Buena Práctica al depósito.

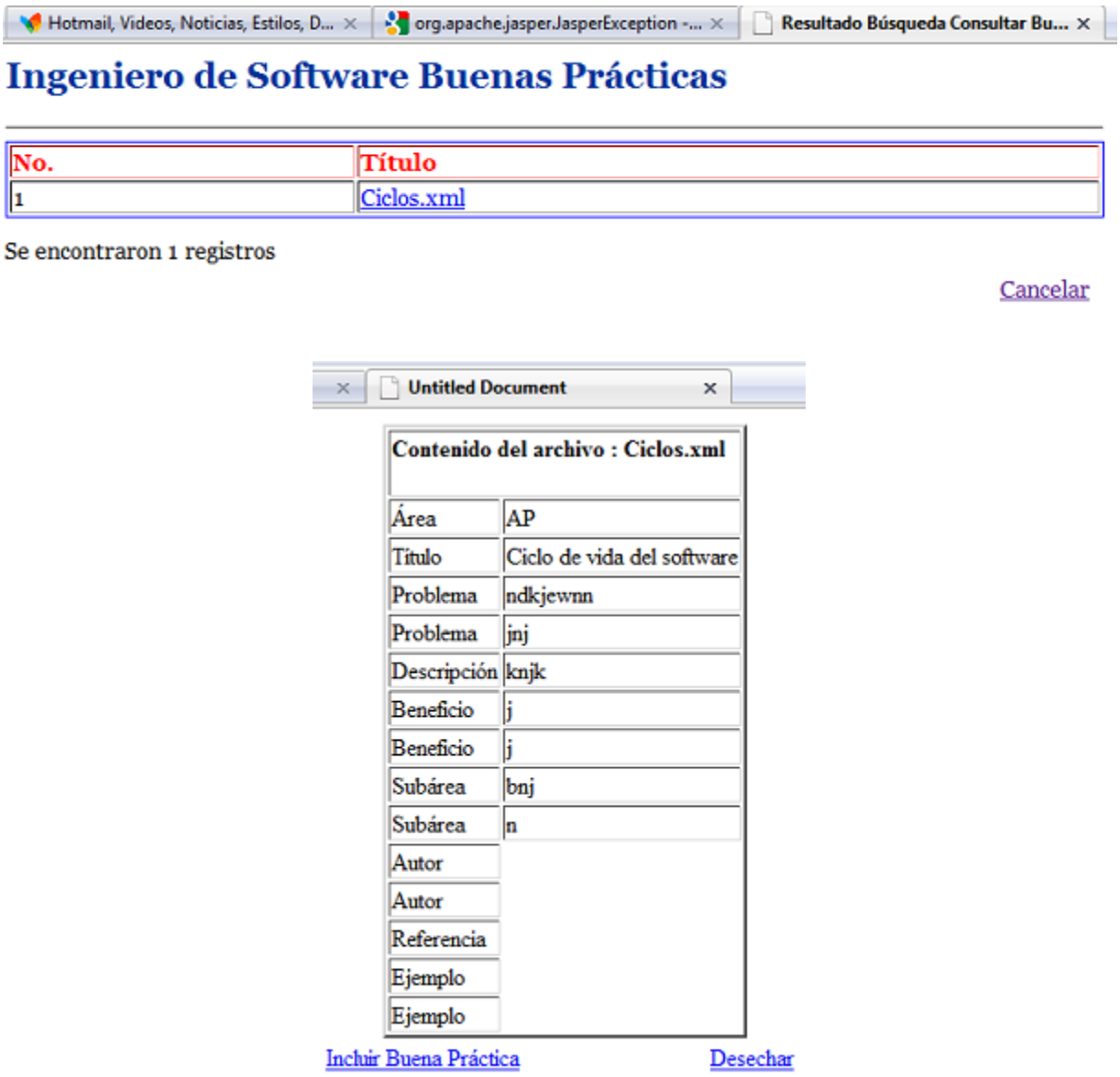
**Flujo:**

| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Elige la opción "Revisar Buena Práctica" del menú principal del Administrador | 2 | Muestra la lista de Buenas Prácticas enviadas. | E2 |
| 3 | Elige una Buena Práctica y presiona el botón donde está el título de la Buena Práctica. | 4 | Muestra en pantalla el archivo de la Buena Práctica. | E2 |

**Poscondiciones**:

- El administrador ha revisado una buena práctica enviada por algún ponente.
- El sistema despliega las buenas prácticas recibidas.

**Prototipo**




## Plan de Prueba

| Caso de Uso | Entradas | Resultados Esperados |
|---|---|---|
| 4 | Elige la opción “Revisar Buena Práctica”. | El sistema muestra la lista de los archivos de las buenas prácticas. |
| 4 | Elige una buena práctica y presiona el donde está el título de la Buena Práctica. | El sistema muestra la buena práctica. |

# Capítulo 4

# Análisis

Flujo de trabajo fundamental cuyo propósito principal es analizar los requisitos descritos en la captura de requisitos, mediante su refinamiento y estructuración. El objetivo de esto es (1) lograr una comprensión más precisa de los requisitos, y (2) obtener una descripción de los requisitos que sea fácil de mantener y que nos ayude a dar estructura al sistema en su conjunto –incluyendo su arquitectura [10].

## 4.1. Diagramas de clases

Los diagramas de clases ayudan a representar elementos involucrados en cada capa del sistema y su relación entre ellos. Para esta aplicación se agruparon en tres capas de acuerdo a su función principal.

### 4.1.1. Presentación

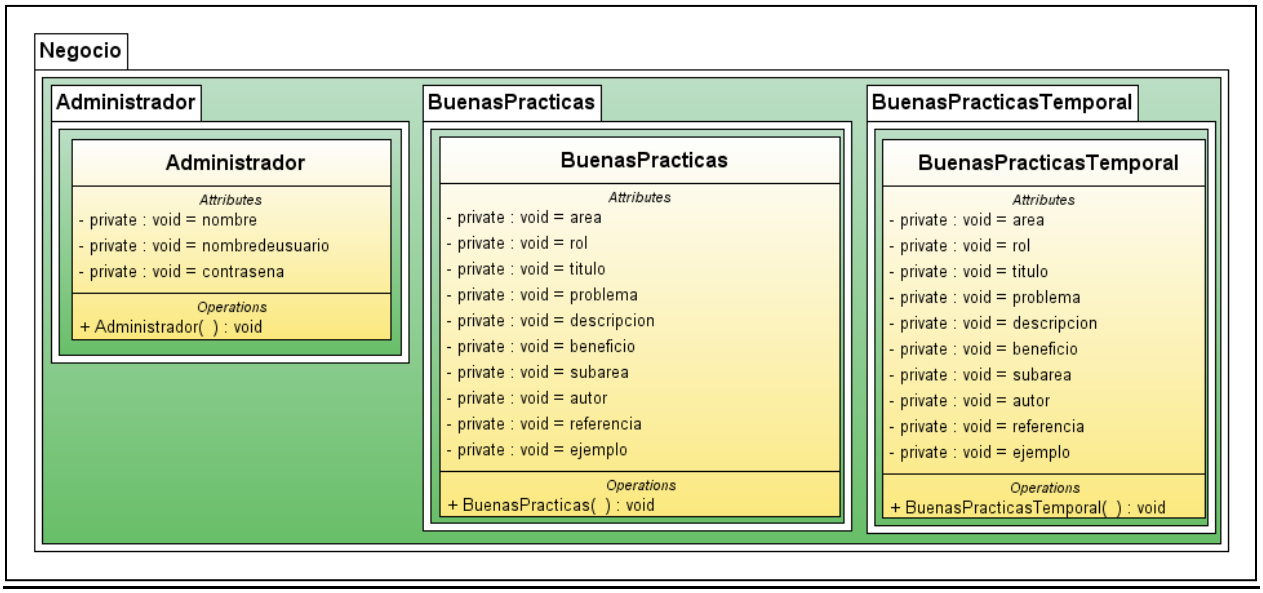
Una clase de presentación representa una “pantalla” o una “Forma de entrada” que solicita información al usuario.

A continuación se incluyen las clases, una por cada pantalla, por cada funcionalidad importante.


Presentación
<<interface>>
PáginaPrincipal
Attributes
Operations
+ PaginaPrincipal( ) : void
<<interface>>
AutenticacionAdministrador
Attributes
Operations
+ AutenticacionAdministrador( ) : void
<<interface>>
PrincipalAdministrador
Attributes
Operations
+ PrincipalAdministrador( ) : void
<<interface>>
FormularioAltaAdmin
Attributes
Operations
+ FormularioAltaAdmin( ) : void
<<interface>>
CapturaAltaAdmin
Attributes
Operations
+ CapturaAltaAdmin( ) : void
<<interface>>
FormDetalleBajaAdmin
Attributes
Operations
+ FormDetBajaAdmin( ) : void
<<interface>>
ResBusquedaBajaAdmin
Attributes
Operations
+ ResBusBajaAdmin( ) : void
<<interface>>
FormBusBajaAdmin
Attributes
Operations
+ FormBusBajaAdmin( ) : void
<<interface>>
ConfirmarBajaAdmin
Attributes
Operations
+ ConfirmaBajaAdmin( ) : void
<<interface>>
ConfirmacionBajaAdmin
Attributes
Operations
+ ConfirmarBajaAdmin( ) : void
<<interface>>
ResBusModificarAdmin
Attributes
Operations
+ ResBusModificaAdmin( ) : void
<<interface>>
FormModificarAdmin
Attributes
Operations
+ FormModificarAdmin( ) : void
<<interface>>
FormBusModificarAdmin
Attributes
Operations
+ FormBusModificaAdmin( ) : void
<<interface>>
CapturaModificarAdmin
Attributes
Operations
+ CapturaModificaAdmin( ) : void
<<interface>>
FormDetalleConsultarAdmin
Attributes
Operations
+ FormDetalleConsultaAdmin( ) : void
<<interface>>
ConsultarAdmin
Attributes
Operations
+ ConsultarAdmin( ) : void
<<interface>>
ResBusConsultarAdmin
Attributes
Operations
+ ResBusConsultaAdmin( ) : void
<<interface>>
CargarDirectorio
Attributes
Operations
+ CargarDirectorio( ) : void
<<interface>>
CapturaIncluirBP
Attributes
Operations
+ CapturaIncluirBP( ) : void
<<interface>>
detalleCargarDirectorio
Attributes
Operations
+ detalleCargarDirectorio( ) : void
<<interface>>
subiryeliminarxml
Attributes
Operations
+ subiryeliminarxml( ) : void
<<interface>>
FormBusquedaBP
Attributes
Operations
+ FormBusquedaBP( ) : void
<<interface>>
FormDetalleBusquedaBP
Attributes
Operations
+ FormDetalleBusquedaBP( ) : void
<<interface>>
ResBusquedaBP
Attributes
Operations
+ ResBusquedaBP( ) : void
<<interface>>
ProponerBP
Attributes
Operations
+ ProponerBP( ) : void
<<interface>>
ResProponerBP
Attributes
Operations
+ ResProponerBP( ) : void

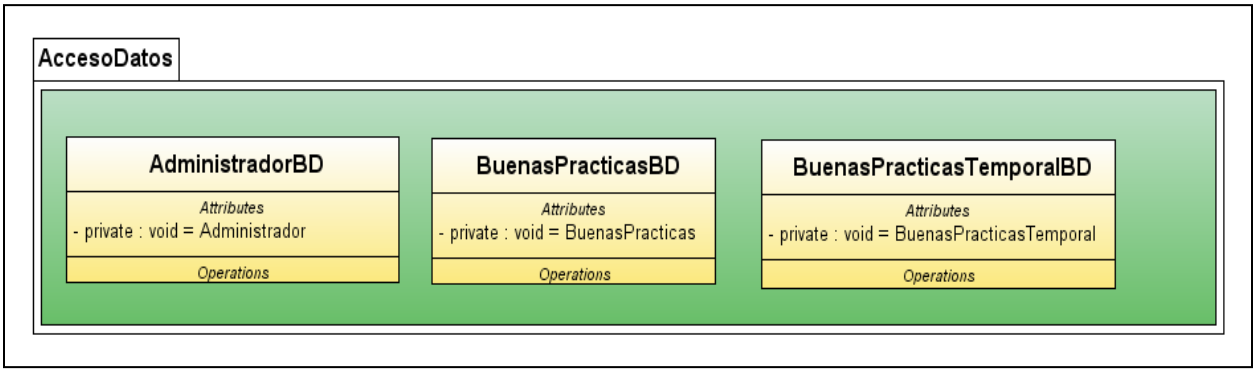
### 4.1.2. Negocio

Las clases de negocio encapsulan el comportamiento y manejan el control del flujo del sistema.




### 4.1.3. Acceso a Datos

Las clases de acceso a datos se crean para las clases de negocio que requieren que los datos sean persistentes. En este caso son dos, ya que una es para los administradores, y la otra es referente a las buenas prácticas.




## 4.2. Diagramas de secuencia

Ya que se identificaron las clases de interfaz, control y acceso a datos, ahora se presentará cómo se realiza la interacción entre los elementos de las tres capas.

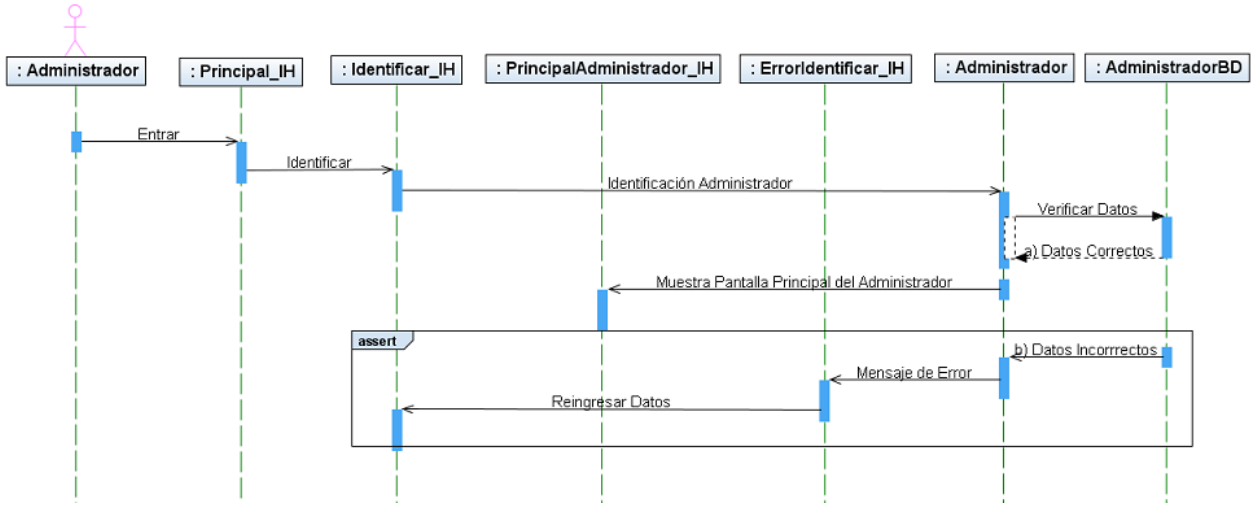
Los diagramas de secuencia muestran las clases que participan en un caso de uso, el usuario (actor) que puede realizar la petición, la secuencia y los mensajes que se le enviarán entre las clases que intervienen en cada uno.

Se muestran las secuencias de operaciones cuando todo sale correctamente y cuando ocurre algún error.

Para no excederse en este punto, sólo se mostrarán los diagramas de aquellas operaciones que se ha creído tienen mayor relevancia. Los demás diagramas se podrán visualizar en el Anexo B2.

Caso de Uso: Identificar

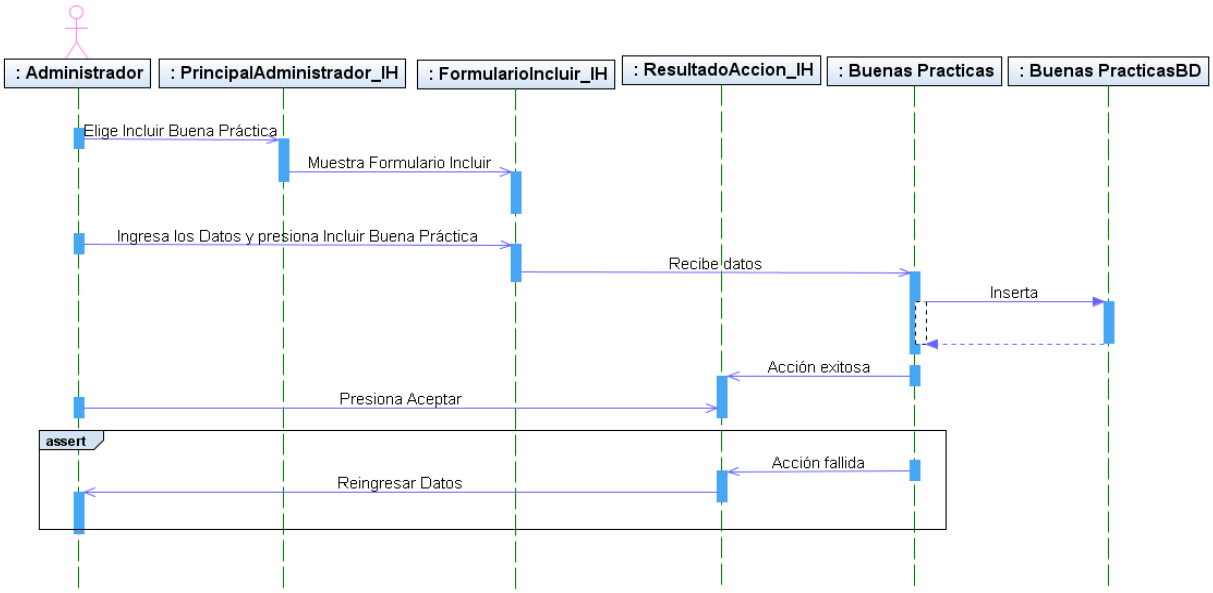
Únicamente el administrador podrá autenticarse para tener ciertas funcionalidades que otros usuarios no tendrán. La clase de control verificará que el nombre de usuario y contraseña sean correctos, para poder entrar a la página principal del administrador.




Caso de Uso: Incluir Buena Práctica

Para incluir una Buena Práctica se ingresan los datos, se validan y la clase Buenas Prácticas incluye la buena práctica.

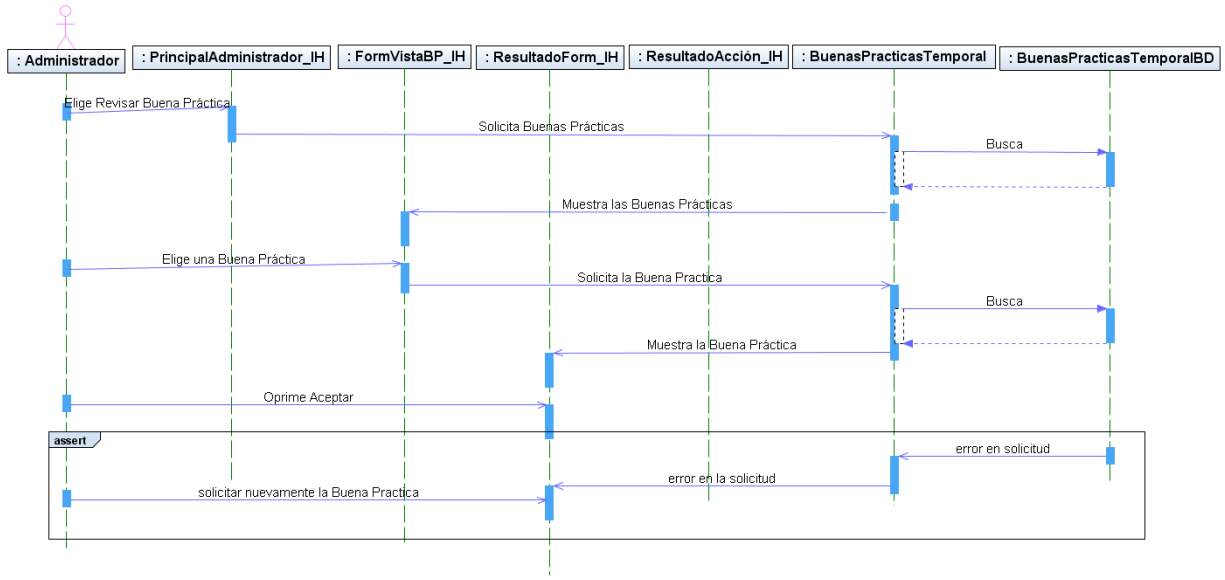
Si existe algún error con la conexión a la base de datos, se mostrará un mensaje para informar al administrador.




Caso de Uso: Recibir Buena Práctica

El administrador solicita las buenas prácticas enviadas por los ponentes, la clase Buenas PrácticasTemporal las muestras.

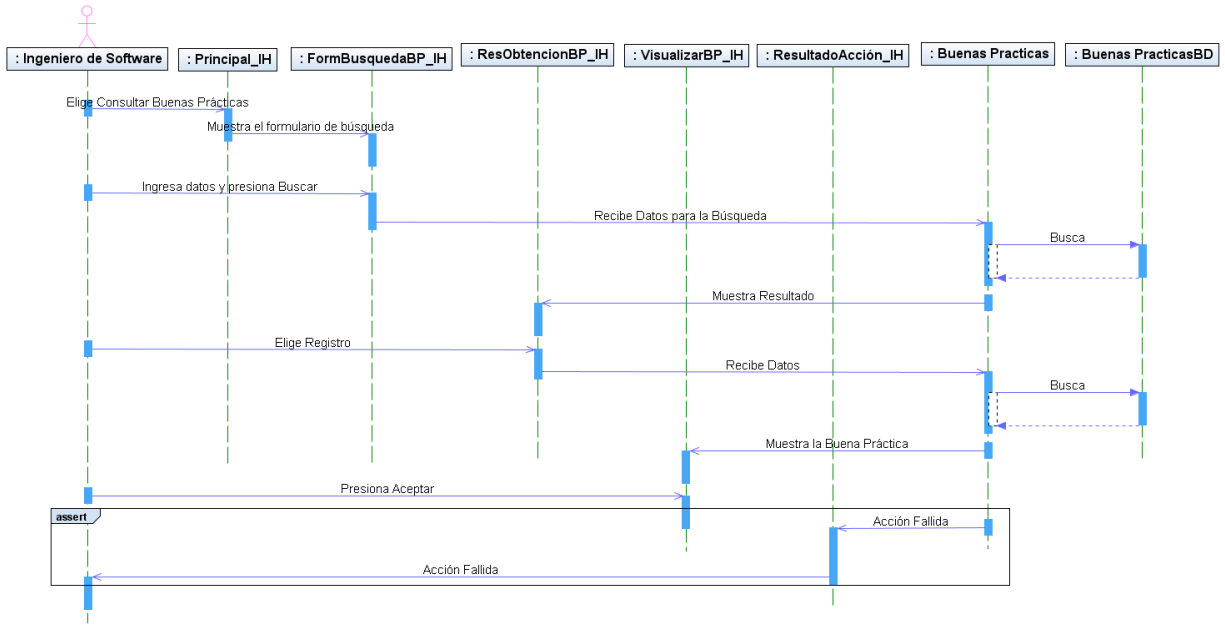
Si existe algún error con la conexión se muestra un mensaje.




Caso de Uso: Consultar Buenas Prácticas

Lee y despliega los datos de todas las buenas prácticas almacenadas en la base de datos, según el criterio seleccionado: Área, Rol, Subárea, Título, Problema, Autor.

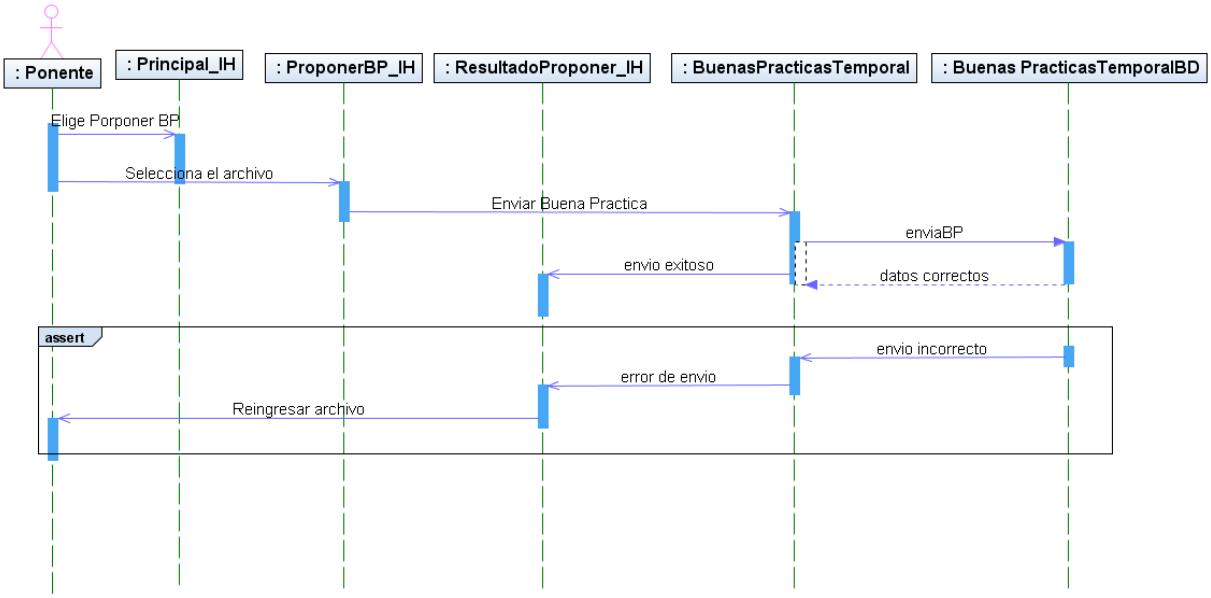
Si existe algún error con la conexión a la base de datos, se mostrará un mensaje para informar al usuario.




Caso de Uso: Proponer Buena Práctica

El ponente selecciona y envía su buena práctica, el sistema verifica los datos y los almacena de manera temporal.

En caso de que exista algún error se mandará un mensaje al ponente, para informarle de dicho error.



## 4.3. Diagramas de navegación

Los diagramas de estados muestran el comportamiento de un objeto, es decir, el conjunto de estados por los cuales pasa un objeto durante su vida, junto con los cambios que permiten pasar de un estado a otro y se usan para modelar la navegación

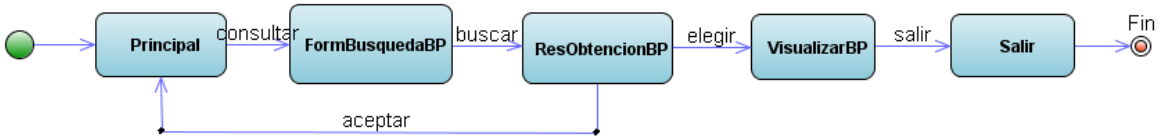
En los diagramas de navegación existen por lo menos dos estados especiales que son el inicial y el final.

A continuación, se explicará la navegación para cada actor del sistema

### 4.3.1 Ingeniero de Software

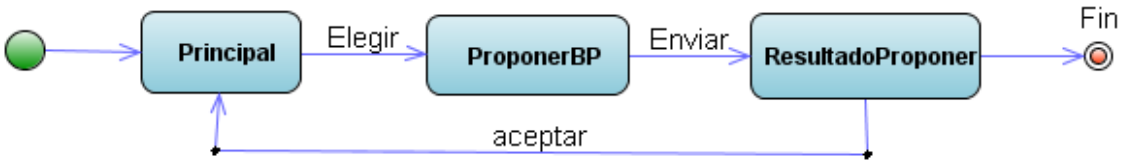
Para el ingeniero de software, se mostrará las “pantallas” y las “Formas de entrada” que se definieron en el diagrama de clases de interfaz, cómo se relacionan y cómo se navega en el sistema.

El estado inicial es la pantalla Principal, y de acuerdo a las acciones del ingeniero se pasa de un estado a otro. Recordemos que el ingeniero únicamente podrá consultar las Buenas Prácticas.



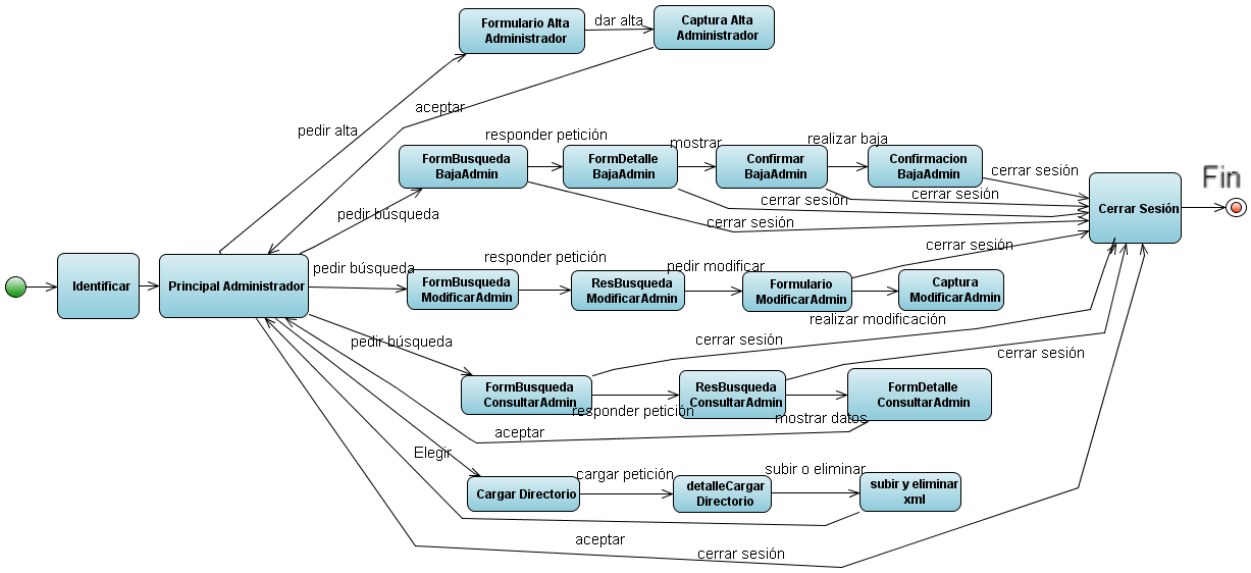

### 4.3.2 Ponente

El Ponente es quien propondrá las buenas prácticas, y después el administrador decidirá si es apropiada para ser incluida en el depósito.



### 4.3.3 Administrador

El administrador es quien lleva la mayor parte de las funciones en el depósito, ya que administrará tanto a los administradores como a las buenas prácticas, además de ser quien decida si las buenas prácticas estarán o no en el depósito.



# Capítulo 5

# Diseño

Flujo de trabajo fundamental cuyo propósito principal es el de formular modelos que se centran en los requisitos no funcionales y el dominio de la solución, y que prepara para la implementación y pruebas del sistema [10].

## 5.1. Descripción de la Arquitectura General

Para este sistema se utilizará el patrón de Multicapas J2EE ya que facilita la organización del sistema dividiendo en tres capas propuestas por éste.
La aplicación de Administración utiliza las siguientes tres capas Presentación, Negocio y Acceso a datos, como se muestra en la figura 5.1 -1.

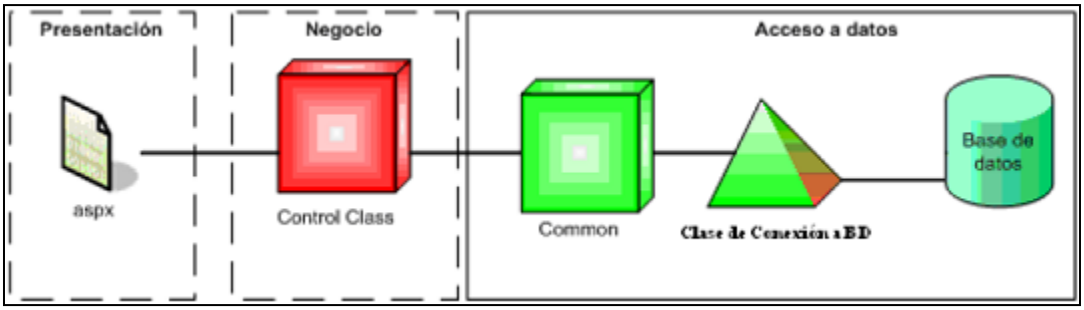



Figura 5.1-1 Modelo del sistema

Capa de Presentación

Esta capa contempla las páginas la interfaz, por ejemplo, las páginas HTML, jsp´s, imágenes, controles web, etc. Son las páginas que observa el usuario y su principal funcionalidad es la de solicitar o mostrar información; también en esta capa se realizan ciertas validaciones de la captura de información.

Capa de Negocio

Esta capa contiene un conjunto de clases de control que contienen las reglas de negocio que contempla el sistema. Aquí se realizan validaciones más complejas, se aplican reglas sobre ciertas funciones de negocio, transacciones, manejo de errores, etc. En otras palabras, son los componentes que dan la funcionalidad al sistema y son los encargados de procesar las peticiones que se realizan en la interfaz.

Capa de Acceso a datos

Esta capa contiene una serie de clases generales por entidad que contienen las principales funciones de acceso a datos del sistema.

Algunas de las funciones de datos más importantes, están concentradas en procedimientos almacenados que se encuentran en la base de datos de la aplicación, que se obtienen a través de la capa de Presentación y de la capa de Negocio.

## 5.2. Ambiente de Desarrollo

A continuación se presenta el desglose de la especificación del ambiente de desarrollo:

- Ambiente de Desarrollo Integrado (IDE)
  NetBeans IDE 6.1

- Compilador
  JDK 6 update 10

- Herramienta de Diagramación UML
  Plug-in UML NetBeans

- Sistema Manejador de Bases de Datos
  eXist XML Database y PostgreSQL

- Servidor de aplicación
  Apache Tomcat 6.0.x

- Diseñador de páginas Web
  Dreamweaver CS3

## 5.3. Diagrama de Distribución

Se muestra la distribución de los nodos (servidor y clientes) del sistema, así como los componentes de cada uno de ellos.
El modelo está formado por un servidor en el que se encuentra la aplicación y las dos bases de datos, y los usuarios que ejecutan desde un navegador web en un cliente.

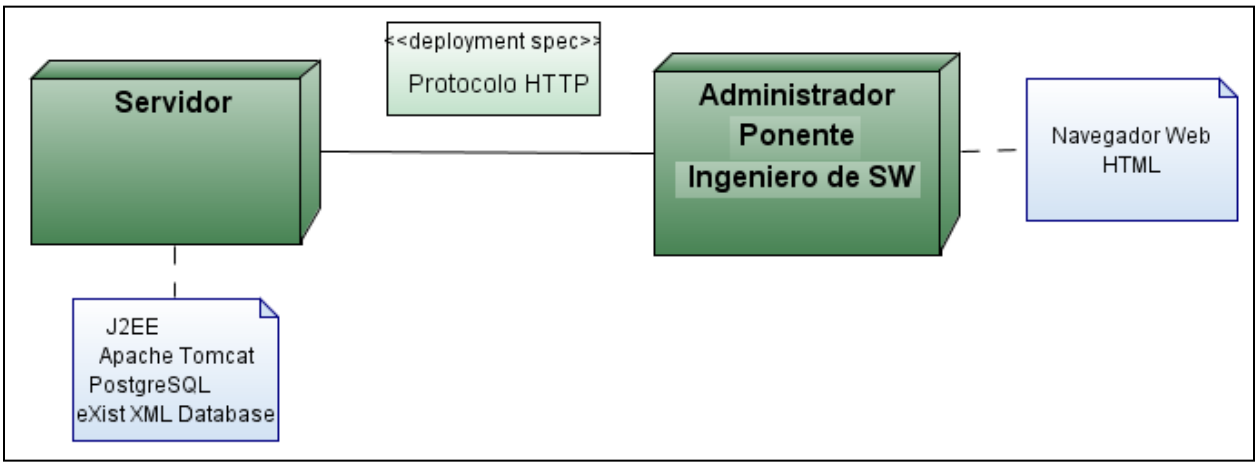


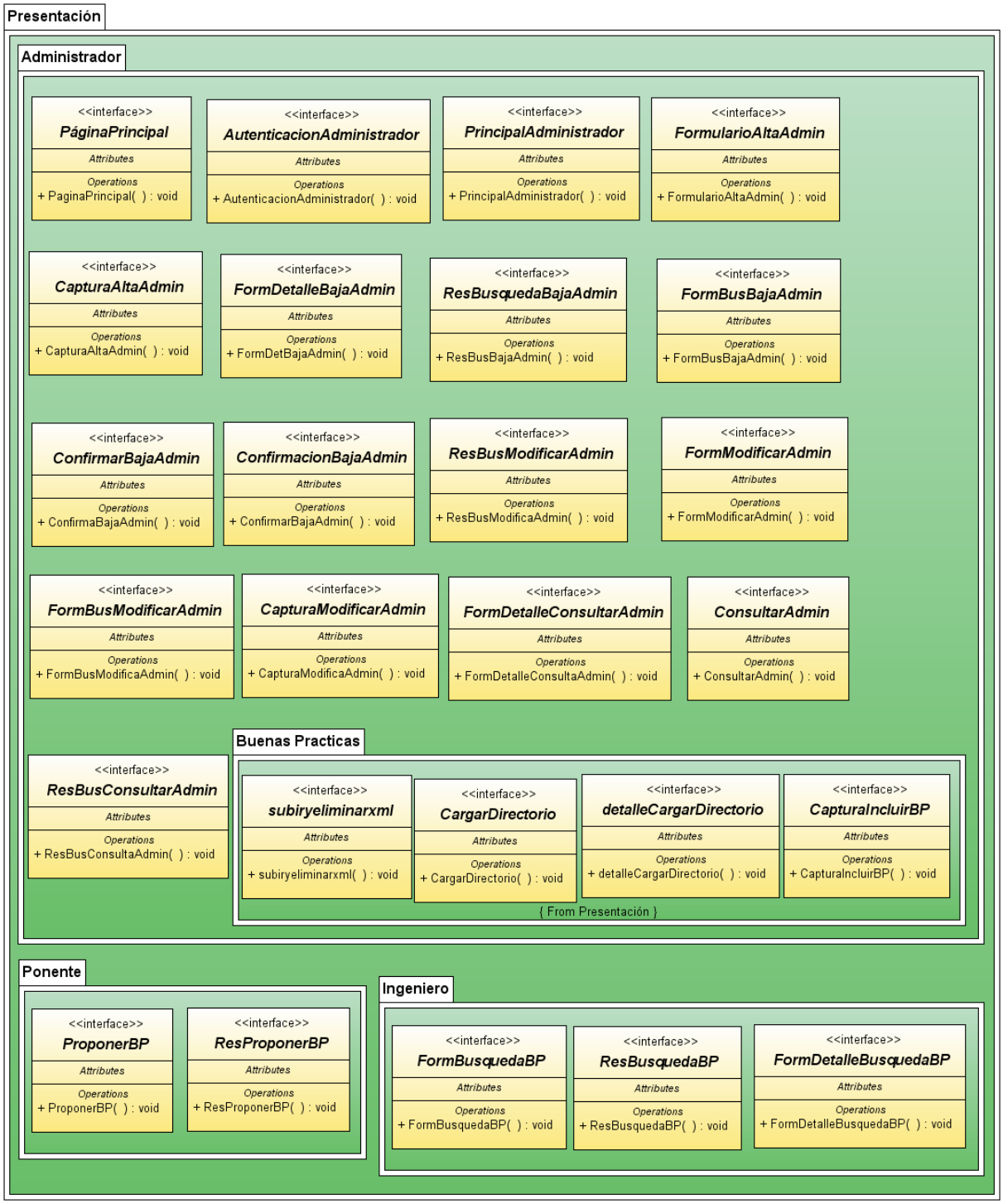

Figura 5.3-1 Diagrama de Distribución

## 5.4. Diagrama de Clases del Diseño Detallado

Es el diagrama que muestra un conjunto de clases, interfaces y colaboraciones y las relaciones entre éstos; los diagramas de clases muestran el diseño de un sistema desde un punto de vista estático; un diagrama que muestra una colección de elementos declarativos [10].

### a. Presentación

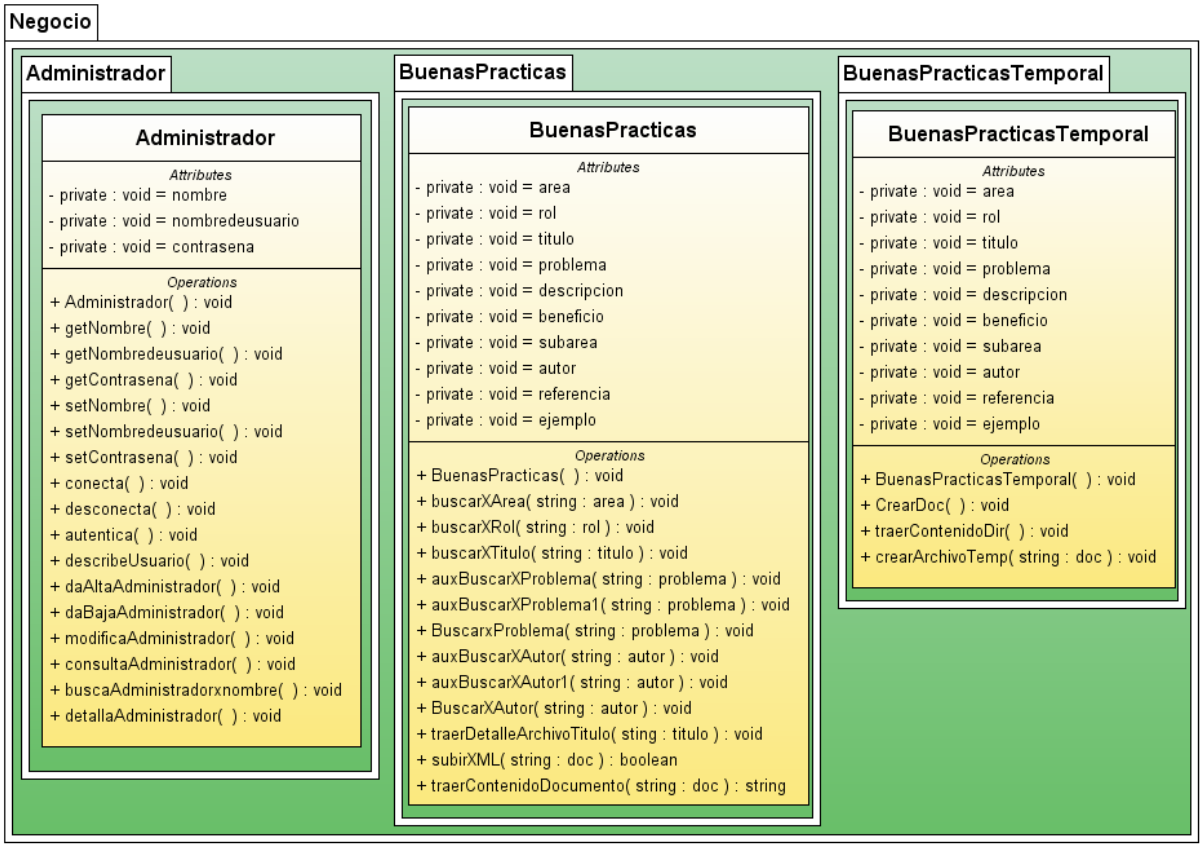
Se tiene el paquete Presentación y dentro se tiene los paquetes: Administrador, Buenas Prácticas, Ponente e Ingeniero. Cada uno contiene las interfaces necesarias para que se pueda mostrar o solicitar la información.




## b. Negocio

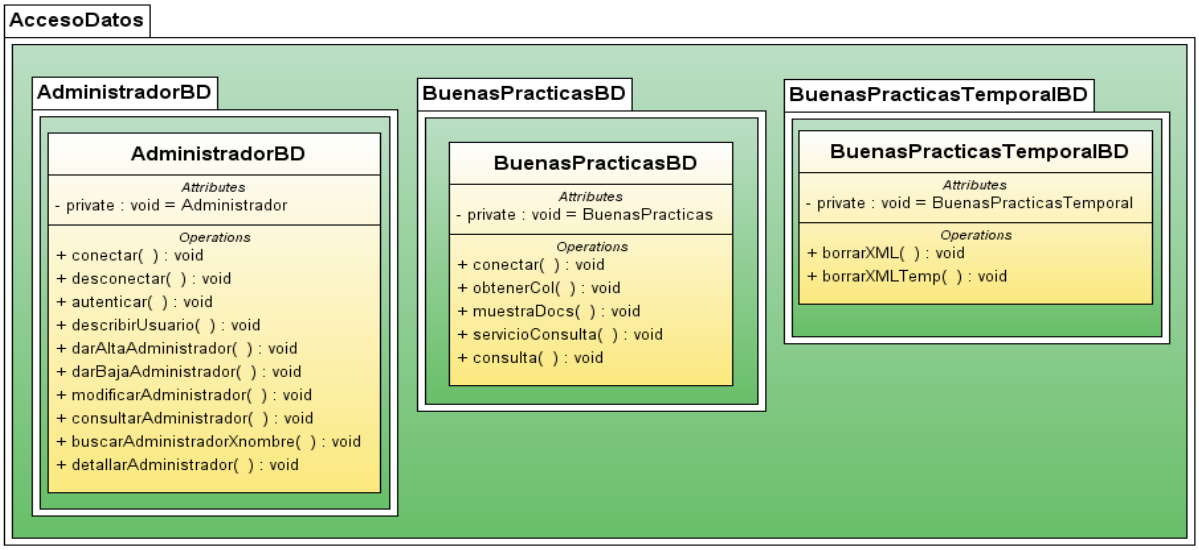
Dentro del paquete Negocio se encuentran los paquetes Administrador, Buenas Prácticas y Buenas Prácticas Temporal, dentro de estos paquetes se encuentran las clases que contiene las reglas de negocio del sistema, por ejemplo, la clase Buenas Prácticas contiene un método llamado subir XML, éste método es el encargado de hacer la petición a la base de datos de poder agregar un documento XML a la base de

datos. Se hace una validación, si la base de datos dice si se puede la sube, si dice que no se genera un mensaje indicando el error.




## c. Acceso a Datos

En el caso del acceso a datos también están las clases dentro de paquetes, la clase de AdministradorBD consulta dentro de la bases de datos relacional, mientras que las otras dos clases consultan en la base de datos nativa.



## 5.5. Diseño de la Base de Datos

Al ser una aplicación sencilla el diseño de las bases de datos resulta ser: (1) para la parte relacional tenemos una tabla donde sólo hay tres campos (nombre, nombre de usuario y contraseña), la cual servirá para la administración de los administradores, y como son los responsables de decidir si las buenas prácticas se incluyen en el depósito o no, es por esta razón que es importante tener registrados a los administradores. Además, hice uso de esta tabla para poder implementar la seguridad ya que no se trata de utilizar la herramienta “moderna” si no de utilizar lo más sencillo pero útil y una base de datos relacional es lo mejor en cuanto a seguridad se refiere.

En la Figura 5.5-1 Diagrama de la Base de Datos parte Relacional, se muestra la entidad con sus campos, en este caso no hay relaciones pues sólo se cuenta con una entidad; y (2) para la parte nativa XML se muestra la Figura 5.5-2 Diseño de la Base de Datos parte Nativa XML, que es el esquema para definir la plantilla de las buenas prácticas.

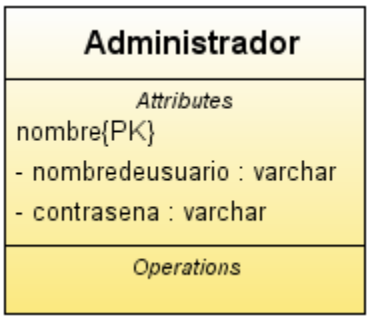



Figura 5.5-1 Diseño de la Base de Datos parte Relacional

El XML Schema es un lenguaje utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así una percepción del tipo de documento con un nivel alto de abstracción.

```
<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>

 <xsd:element name='practica'>
  <xsd:complexType>
   <xsd:sequence minOccurs='0' maxOccurs='unbounded'>
    <xsd:element ref='area'/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>

 <xsd:element name='area'>
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref='rol'/>
    <xsd:element ref='subarea' minOccurs='0' maxOccurs='unbounded'/>
    <xsd:element ref='titulo'/>
    <xsd:element ref='problema' maxOccurs='unbounded'/>
    <xsd:element ref='descripcion'/>
    <xsd:element ref='beneficio' maxOccurs='unbounded'/>
    <xsd:element ref='autor' minOccurs='0' maxOccurs='unbounded'/>
    <xsd:element ref='referencia' minOccurs='0' maxOccurs='1'/>
    <xsd:element ref='ejemplo' minOccurs='0' maxOccurs='unbounded'/>
   </xsd:sequence>
   <xsd:attribute name='nombre' default='IS'>
    <xsd:simpleType>
     <xsd:restriction base='xsd:string'>
      <xsd:enumeration value='IS'/>
      <xsd:enumeration value='BD'/>
      <xsd:enumeration value='AP'/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:attribute>
  </xsd:complexType>
 </xsd:element>

 <xsd:element name='rol' type='xsd:string'>
 </xsd:element>

 <xsd:element name='subarea' type='xsd:string'>
 </xsd:element>

 <xsd:element name='titulo' type='xsd:string'>
 </xsd:element>

 <xsd:element name='problema' type='xsd:string'>
 </xsd:element>

 <xsd:element name='descripcion' type='xsd:string'>
 </xsd:element>

 <xsd:element name='beneficio' type='xsd:string'>
 </xsd:element>

 <xsd:element name='autor' type='xsd:string'>
 </xsd:element>

 <xsd:element name='referencia' type='xsd:string'>
 </xsd:element>

 <xsd:element name='ejemplo' type='xsd:string'>
 </xsd:element>
</xsd:schema>
```

Figura 5.5-2 Diseño de la Base de Datos parte Nativa XML

Explicando este esquema: el elemento raíz se llama “practica” y tiene un hijo llamado “area” el cual tiene ocho hijos y un atributo. Los hijos son “rol”, “subarea”, “titulo”, “problema”, “descripcion”, “beneficio”, “autor”, “referencia”, “ejemplo”. Los elementos “titulo” y “descripcion”, deben aparecer una vez, los elementos “problema” y “beneficio” pueden aparecer de una a dos veces, los elementos “subarea”, “autor” y “ejemplo” pueden aparecer de cero a dos veces y el elemento “referencia” puede aparecer de cero a una vez, El hecho de que estén agrupados en una secuencia indica que los elementos deben aparecer en orden, es decir, primero el “titulo”, luego el “problema”, y así sucesivamente hasta que el ultimo debe ser el “ejemplo”. Los ocho elementos son de tipo *string*. El atributo de “area” se llama “nombre” es de tipo *string* y sólo puede tomar tres valores que son “IS”, “BD” y “AP”.

# Capítulo 6

# Implementación y Pruebas

La Implementación es un flujo de trabajo fundamental cuyo propósito esencial es construir el sistema en términos de componentes, es decir, código fuente, guiones, ficheros binarios, ejecutables, etc. [10]

## 6.1. Código de Clases

A continuación se mostrará algunos métodos en el sistema utilizados para el manejo de las buenas prácticas.

Para hacer la conexión a la base de datos eXist, se carga el driver, se instancia y se registra la base de datos.

```
public ArrayList buscarXArea(String area) {
    ArrayList info_prac = new ArrayList();
    try {
     System.out.println("Areas");
     String consulta = "let $collection:= '/db/DepositoBP/' for $file in collection($collection)
               let $name := util:document-name($file) let $doc := concat ($collection,$name)
               for $b in doc($doc)//practica/area
               where $b[contains(upper-case(@nombre),upper-case('"+area+"'))]
               return concat($name,concat(',',$b/titulo))";

     ResourceSet result = BuenasPracticasBD.Consulta(consulta);
      try {
         //ResourceSet result = rs;
         ResourceIterator i = result.getIterator();
         while (i.hasMoreResources()) {
             Resource r = i.nextResource();
             info_prac.add((String) r.getContent());
             System.out.println((String) r.getContent());
         }

     } catch (XMLDBException ex) {
         Logger.getLogger(BuenasPracticasBD.class.getName()).log(Level.ALL, null, ex);
     }

     } catch (Exception ex) {
         Logger.getLogger(BuenasPracticasBD.class.getName()).log(Level.ALL, null, ex);
     }
     return info_prac;
 }
```

Todas las búsquedas son similares lo único que cambia son las consultas. En este caso se hace la consulta por áreas, entonces primero se hace la consulta y se lanza con *ResourceSet result = BuenasPracticasBD.Consulta(consulta)*, esto es un documento que contiene a la plantilla que cumple con la condición, después se procesa el resultado (*while(i.hasMoreResources())*).

```
protected static String driver = "org.exist.xmldb.DatabaseImpl";
protected static String URI = "xmldb:exist://localhost:8080/exist/xmlrpc/db/DepositoBP";
protected static Database database= null;

public static void conectar() {
try
{
    Class cl = Class.forName(driver);
    Logger.getLogger("Loading driver.");
    database = (Database)cl.newInstance();
    DatabaseManager.registerDatabase(database);
    System.out.println("conectado a exist");
}
catch (XMLDBException ex){
          Logger.getLogger(BuenasPracticasBD.class.getName()).log(Level.ALL,null, ex);
    }
catch (InstantiationException ex) {
        Logger.getLogger(BuenasPracticasBD.class.getName()).log(Level.ALL, null, ex);
    }
catch (IllegalAccessException ex) {
        Logger.getLogger(BuenasPracticasBD.class.getName()).log(Level.ALL, null, ex);
    }
catch (ClassNotFoundException ex) {
        Logger.getLogger(BuenasPracticasBD.class.getName()).log(Level.ALL, null, ex);
    }
}
```

Para subir el documento XML a la base de datos eXist, en la primera línea se obtiene la colección donde se desea almacenar el xml, en la segunda línea se dan los datos del documento *doc* a subir a la base de datos, en la siguiente línea se indica que es un archivo llamado doc por si hay algún problema al leerlo.

Con *document.setContent(archi);* se sube el archivo a la base de datos. Y con *archi.delete();* se borra el archivo de la ruta temporal donde se guardó.

```
public boolean subirXML(String doc){
    try{
  Collection coleccion = BuenasPracticasBD.ObtenerCol();
  XMLResource document = (XMLResource) coleccion.createResource(doc, "XMLResource");
  File archi = new File ("c:/Users/Owner/Desktop/Carpeta/" + doc);
  if (!archi.canRead()) {
  System.err.println("can't read file " + doc);
 return false;
      }
  else
  {
  document.setContent(archi);
  System.out.print("storing document ...");
  coleccion.storeResource(document);
  System.out.println("ok.");
  archi.delete();
  }
  }
    catch (XMLDBException ex)
   {
      Logger.getLogger(Practica.class.getName()).log(Level.ALL, null, ex);
  }
    return true;
}
```

Aquí sólo se mostró una parte del código, se entregará un CD con el proyecto completo y los archivos referentes a la versión final del código de clases se encuentran en el directorio *src* de Deposito.

## 6.2. Reporte del plan de prueba del Sistema

Las pruebas son un flujo de fundamental cuyo propósito esencial es comprobar el resultado de la implementación mediante las pruebas de cada construcción.

A continuación se mostrarán los reportes de pruebas del sistema, de acuerdo a cada actor.

### Ingeniero de Software

**Caso de uso 6:** Consultar Buenas Prácticas

**Plan de Prueba**

| Caso de Uso | Entradas | Resultados Esperados | Resultados Obtenidos |
|---|---|---|---|
| 6 | Se introducen criterios de búsqueda adecuados y se presiona el botón “Buscar”. | El sistema muestra el listado de resultados. | El sistema muestra el listado de resultados. |
| 6 | Se introducen criterios de búsqueda incompletos o incorrectos y se presiona el botón “Buscar”. | El sistema muestra una pantalla solicitando nuevamente los datos. | El sistema muestra una pantalla solicitando nuevamente los datos. |
| 6 | Selecciona un registro y presiona el botón “Consultar” | El sistema muestra la pantalla del registro. | El sistema muestra la pantalla del registro. |
| 6 | Selecciona un registro y no presiona el botón “Consultar” | El sistema no cambia el estado. | El sistema no cambia el estado. |

### Ponente

**Caso de uso 7:** Proponer Buena Práctica

**Plan de Prueba**

| Caso de Uso | Entradas | Resultados Esperados | Resultados Obtenidos |
|---|---|---|---|
| 7 | Se introducen los datos requeridos | El sistema muestra pantalla de confirmación. | El sistema muestra pantalla de confirmación. |
| 7 | No se introducen los datos requeridos | Se muestra mensaje, con la petición de llenar los datos faltantes. | Se muestra mensaje, con la petición de llenar los datos faltantes. |

## Administrador

**Caso de uso 2:** Identificar

**Plan de prueba**

| Caso de Uso | Entradas | Resultados Esperados | Resultados Obtenidos |
|---|---|---|---|
| 2 | Se introducen nombre de usuario y contraseña correctos y se presiona el botón “Aceptar”. | El sistema permite el acceso con privilegios adecuados. | El sistema permite el acceso con privilegios adecuados. |
| 2 | Se introduce de manera incorrecta el nombre de usuario o contraseña y se presiona el botón “Aceptar”. | El sistema no permite el acceso. | El sistema no permite el acceso. |
| 2 | No se introduce ningún dato y se presiona el botón “Aceptar”. | El sistema no permite el acceso. | El sistema no permite el acceso. |

**Caso de uso 3:** Incluir Buena Práctica

**Plan de Prueba**

| Caso de Uso | Entradas | Resultados Esperados | Resultados Obtenidos |
|---|---|---|---|
| 3 | Se presiona el botón “Incluir Buena Práctica”. | El sistema envía el archivo a la base de datos. | El sistema envía el archivo a la base de datos. |

**Caso de uso 4:** Recibir Buena Práctica

**Plan de Prueba**

| Caso de Uso | Entradas | Resultados Esperados | Resultados Obtenidos |
|---|---|---|---|
| 4 | Elige la opción “Revisar Buena Práctica”. | El sistema muestra la lista de los archivos de las buenas prácticas. | El sistema muestra la lista de los archivos de las buenas prácticas. |
| 4 | Elige una buena práctica y presiona el registro donde está el título de la Buena Práctica. | El sistema muestra la buena práctica. | El sistema muestra la buena práctica. |

# Capítulo 7

# Navegación del Sistema

En este capítulo se proporcionará una guía para entender el funcionamiento del sistema y uso del depósito.

La navegación se dividirá en tres: una donde participan el ingeniero de software, otra donde participa el ponente, y la otra donde participa el Administrador.

## 7.1. Navegación del Ingeniero de Software

La navegación corresponde a la consulta de buenas prácticas.

De la figura 7.1-1**Página Principal** se elige la opción "*Consultar Buenas Prácticas*".

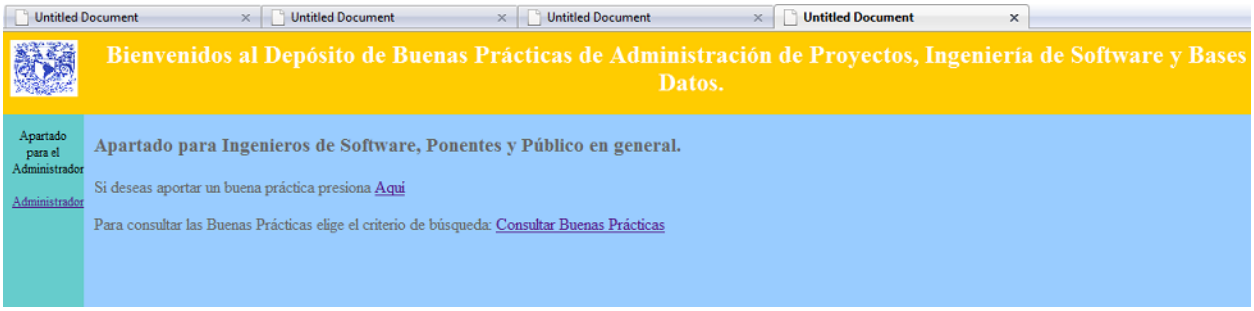



Figura 7.1-1 Página Principal

Se mostrará la figura 7.1-2 **Formulario Búsqueda Buenas Prácticas**, aquí hay cuatro opciones (Área, Título, Problema y Autor) dependiendo del criterio por el cual se quiera realizar la búsqueda. Si se ingresa más de un criterio al mismo tiempo, la búsqueda se realizará con el primer criterio llenado. Ya que se eligió un criterio se da clic en *Buscar*.

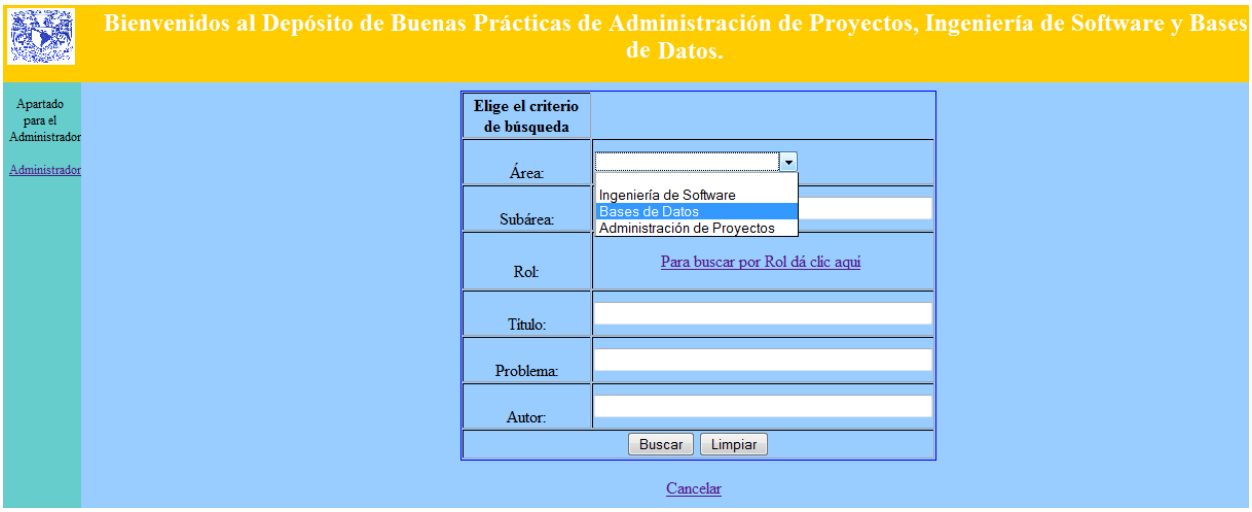



Figura 7.1-2 Formulario Búsqueda Buenas Prácticas

En caso de que no exista lo que se busca aparecerá la Figura 7.1-3 **Registros nulos**, la cual indica que no se encontraron registros con la petición dada.

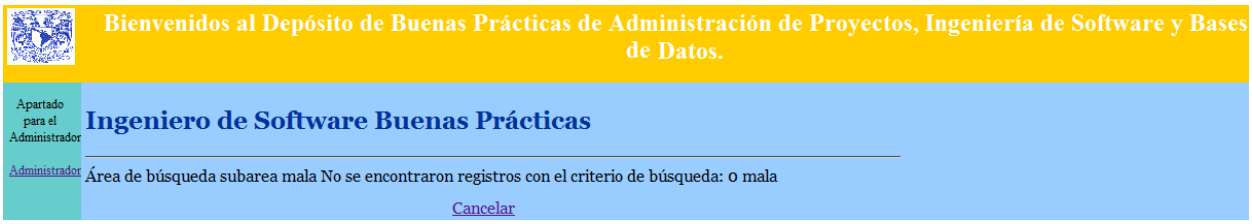



Figura 7.1-3 Registros Nulos

Cuando hay registros encontrados aparece un listado con las buenas prácticas que se encuentran en la base de datos, como se muestra en la Figura 7.1-4. **Registros Encontrados**

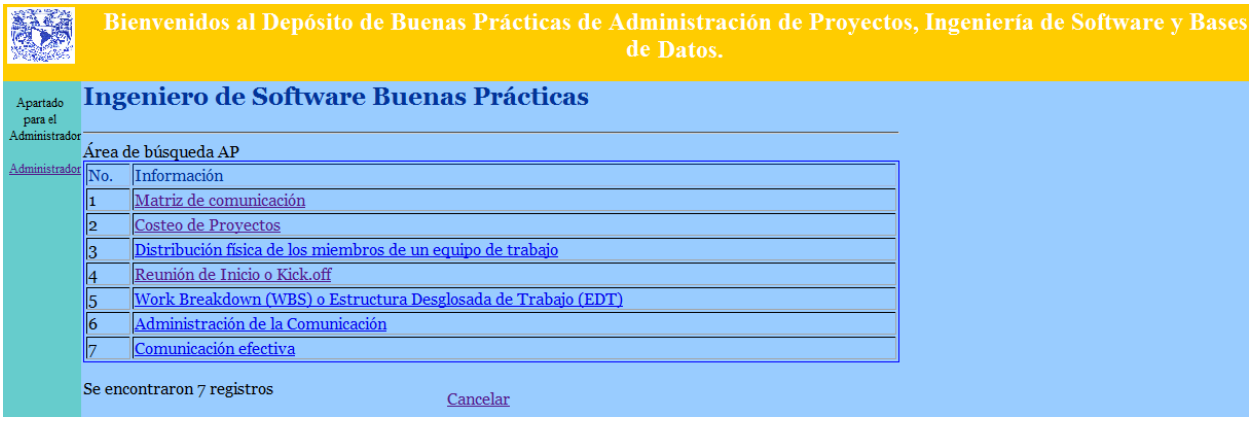



Figura 7.1-4 Registros Encontrados

Al elegir una se da clic en ella y aparece una ventana con el documento de la buena práctica,

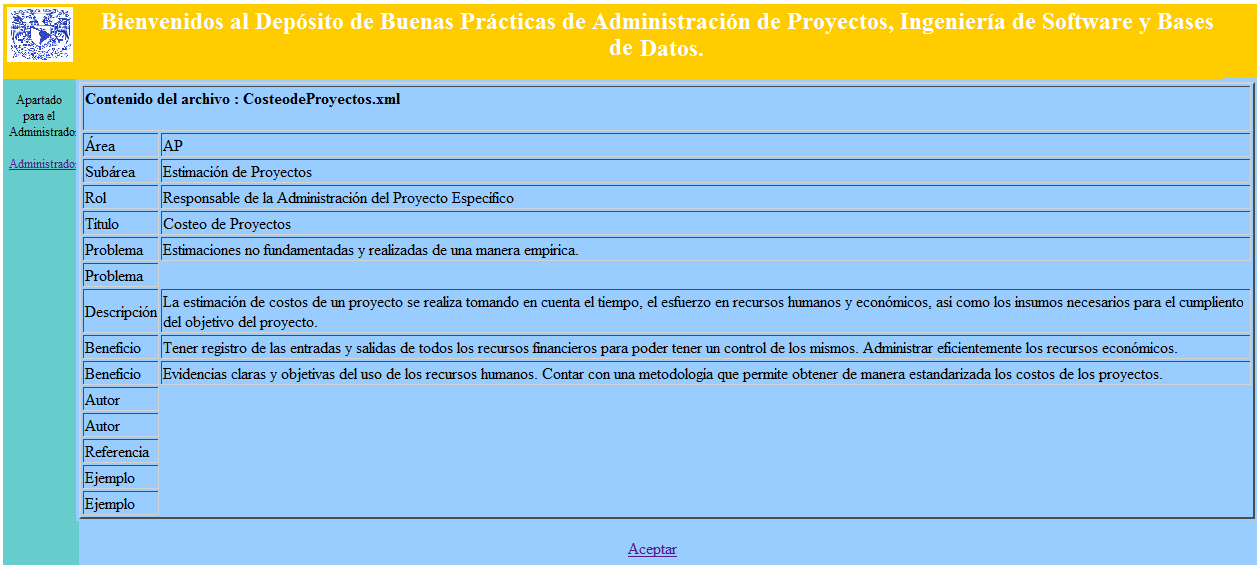



Figura 7.1-5 Documento Encontrado

Al dar clic en *Aceptar* regresará a la Figura 7.1-1.

## 7.2. Navegación del Ponente

Partiendo de la Figura 7.1-1 al dar clic en *Aquí*, aparecerá la Figura 7.2-1 **Ingreso de buena práctica**, la cual es el formulario para generar el archivo de la buena práctica.

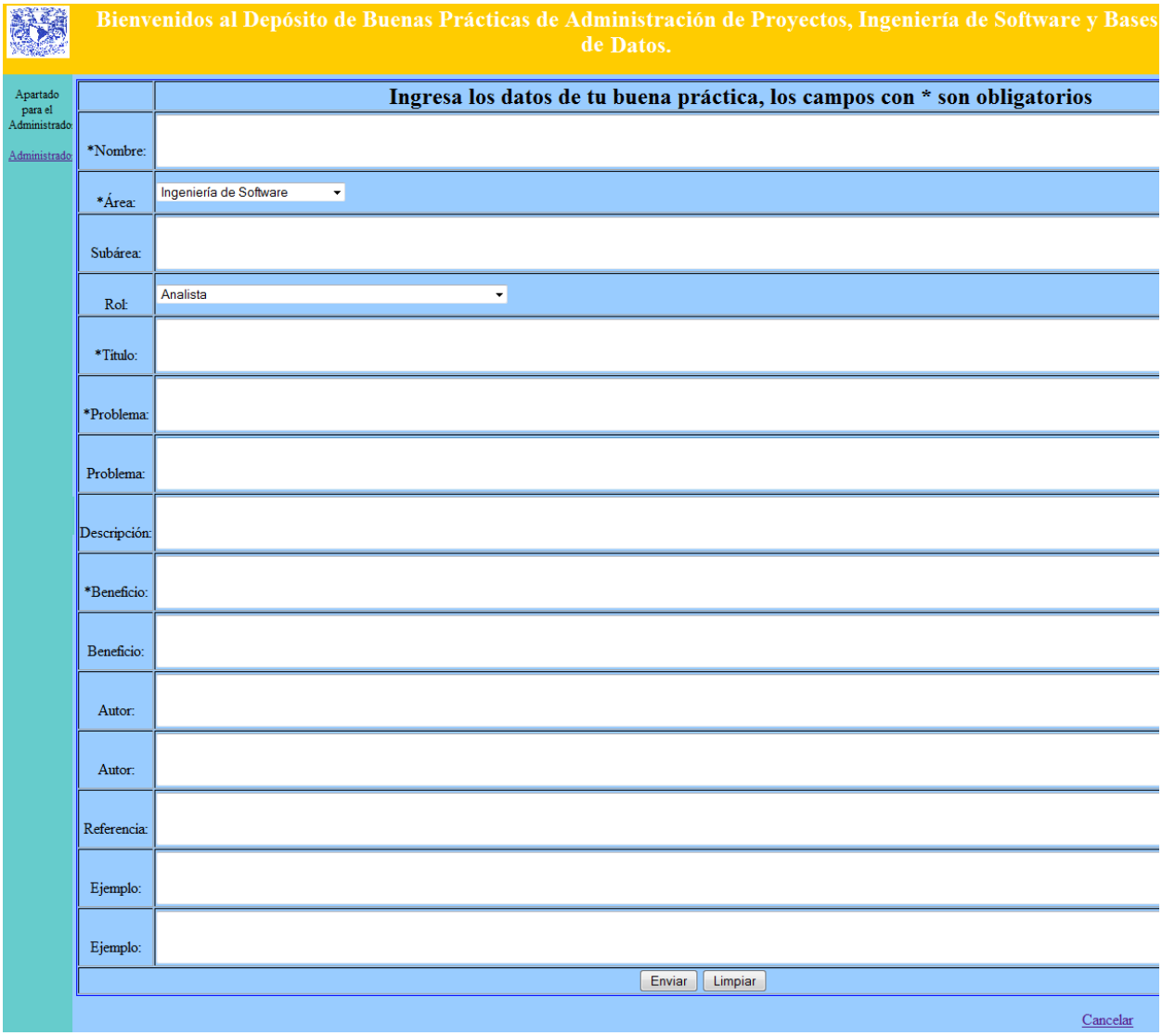



Figura 7.2-1 Ingreso de buena práctica

En caso de que se envíe el archivo sin algunos de los datos requeridos aparecerá un mensaje indicando que faltan datos, tal como se muestra en la Figura 7.2-2 **Mensaje Error**

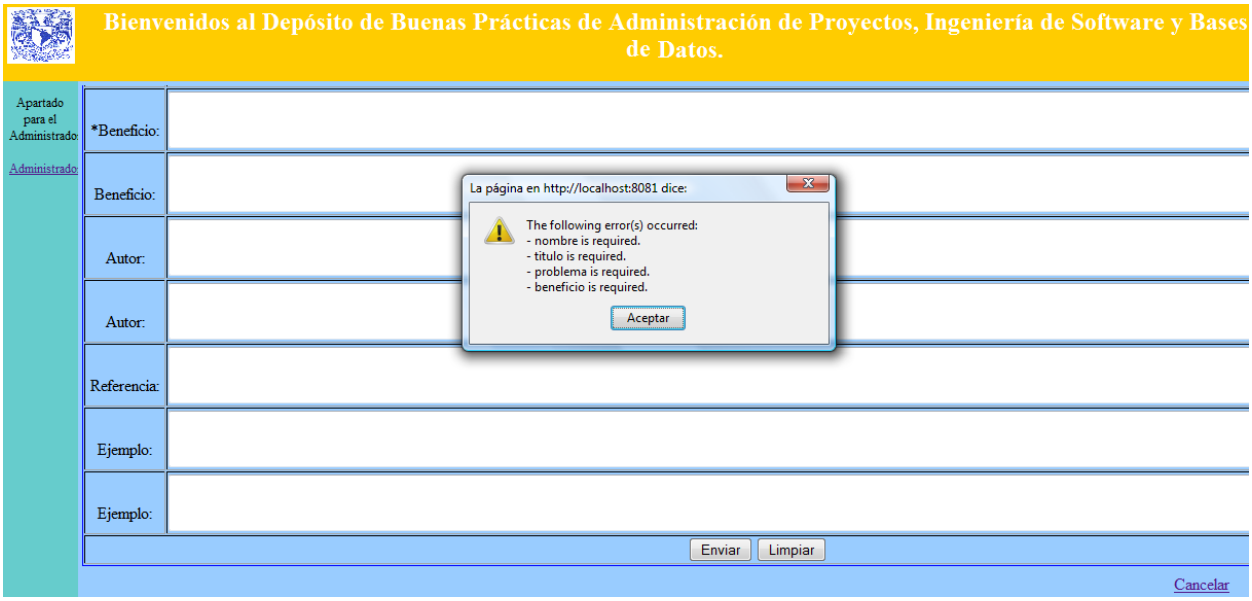



Figura 7.2-2 Mensaje Error

Cuando se llenan los campos necesarios, aparece la un mensaje indicando que la buena práctica fue enviada, como se ve en la Figura 7.2-3 **Buena Práctica enviada**.

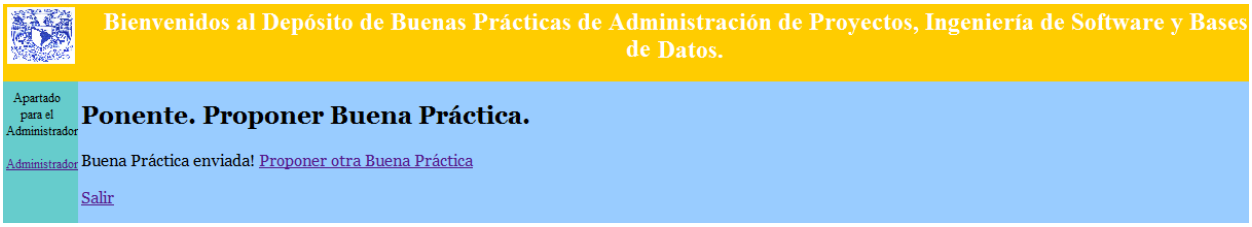



Figura 7.2-3 Buena Práctica enviada

De esta figura al elegir la opción de *Proponer Buena Práctica*, regresará la Figura 7.2-1 y al elegir *Salir* aparecerá la Figura 7.1-1.

## 7.3. Navegación del Administrador

En la página principal de lado izquierdo aparece un apartado para el administrador, para poder ingresar a esta parte se deberá tener el rol de Administrador, al ingresar se pedirá Nombre de usuario y Contraseña, como se muestra en la Figura 7.3-1 **Autenticar**.

Cabe recalcar que la contraseña es sensible, lo cual significa que hay que respetar las mayúsculas y minúsculas en la misma.


Figura 7.3-1 Autenticar

En caso de no anotar uno o ambos datos aparecerá un mensaje pidiendo los datos necesarios, como se muestra en la Figura 7.3-2 **Mensaje Error**, se presiona *Aceptar* y se incorporan los datos correctos.

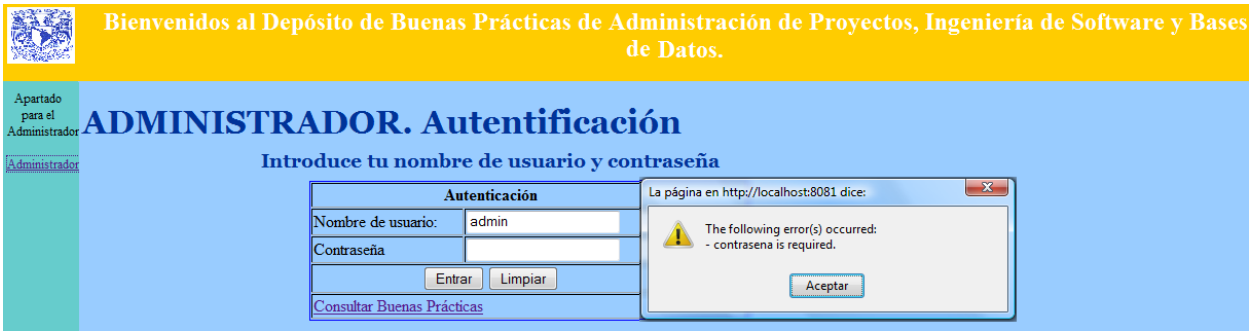



Figura 7.3-2 Mensaje Error

Una vez ingresado su Nombre de usuario y Contraseña, dé clic en *Entrar*.

A continuación se mostrará la Figura 7.3-3 **Principal del Administrador**, donde se podrá elegir entre la administración de los administradores, la administración de las buenas prácticas o cerrar la sesión.

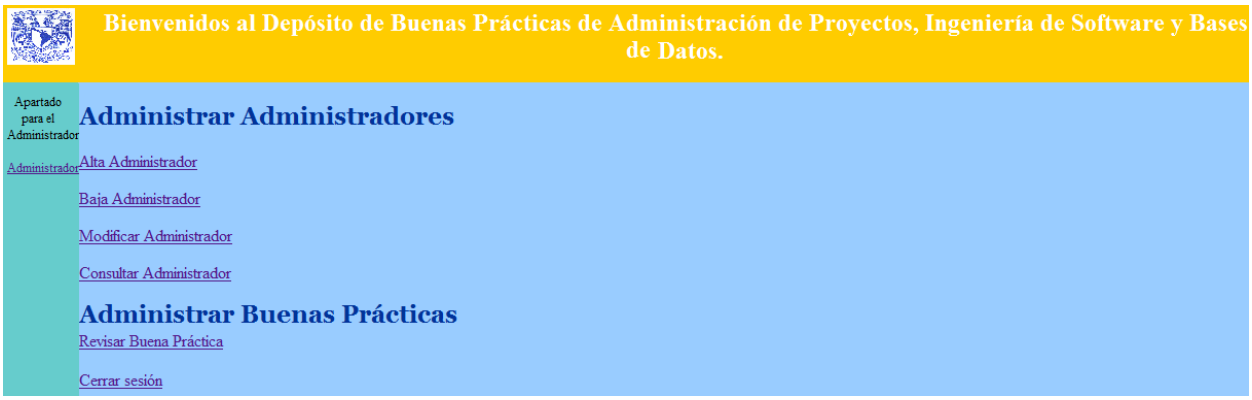



Figura 7.3-3 Principal del Administrador

A continuación se muestra el detalle de cada uno:

### 7.3.1. Administrar Administradores:

Dentro de Administrar Administradores hay cuatro opciones a elegir: Alta, Baja, Modificar y Consultar Administradores.

Al escoger *Alta de Administradores* aparecerá la Figura 7.3.1-1 **Formulario Alta Administrador**, aquí se pide el nombre, nombre de usuario y contraseña del nuevo administrador.

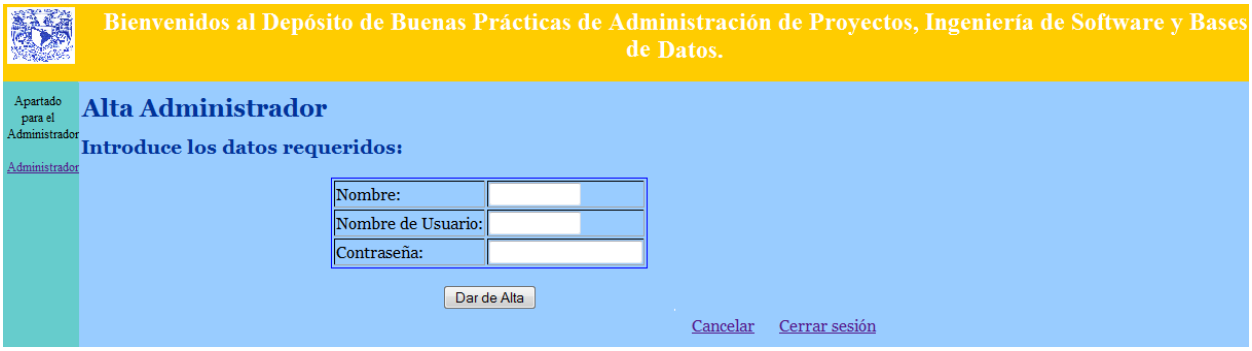



Figura 7.3.1-1 Formulario Alta Administrador

Al llenar los datos solicitados y dar clic en *Dar de Alta*, los datos se envían a la base de datos y aparece la Figura 7.3.1-2 **Mensaje de Administrador registrado**.

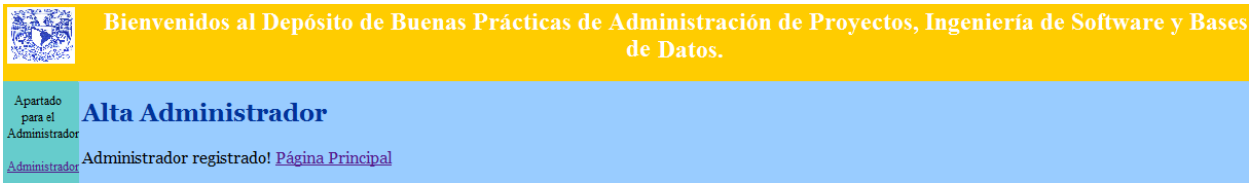



Figura 7.3.1-2 Mensaje de Administrador registrado

Al escoger *Baja de Administradores* aparecerá la Figura 7.3.1-3 **Formulario Baja Administrador**, aquí se pide el nombre del administrador a dar de baja.

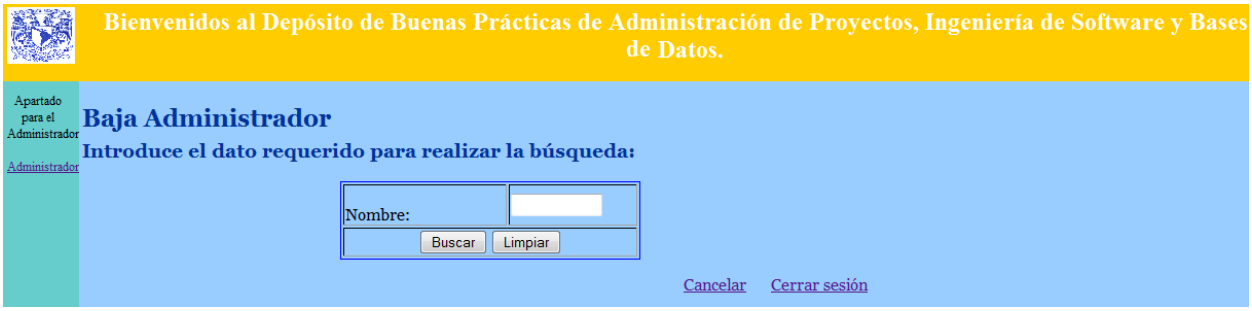



Figura 7.3.1-3 Formulario Baja Administrador

Después de introducir el nombre se oprime la opción *Buscar*, y aparece la Figura 7.3.1-4 **Respuesta al Formulario Baja Administrador**, aquí se detallan los datos del administrador a dar de baja al oprimir en *Dar de Baja* se despliega la Figura 7.3.1-5 **Confirmar al Formulario Baja Administrador,** la cual es para confirmar si realmente se quiere dar de baja al administrador.


Figura 7.3.1-4 Respuesta al Formulario Baja Administrador

Ya que se decidió que si se quiere dar de baja se oprime *Confirmar Baja* y los datos de ese administrador son borrados de la base de datos y se muestra la Figura 6.5-6 **Mensaje de Baja del Administrador**.


Figura 7.3.1-5 Confirmar al Formulario Baja Administrador

Esta pantalla sólo confirma que ya no existe tal administrador.

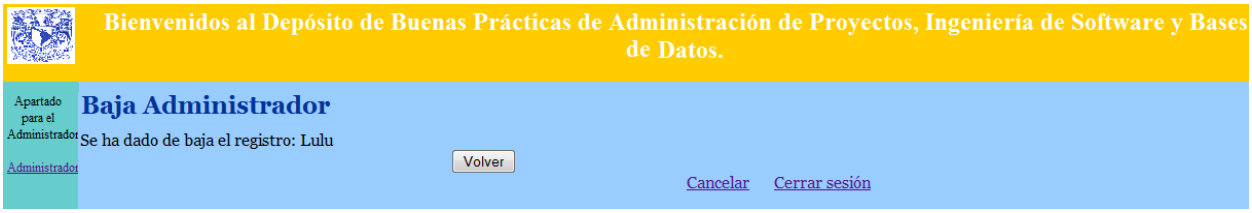



Figura 7.3.1-6 Mensaje de Baja del Administrador

Al escoger *Modificar Administrador* aparecerá la Figura 7.3.1-7 **Búsqueda Modificar Administradore**s, aquí se pide el nombre del administrador que se desea modificar.

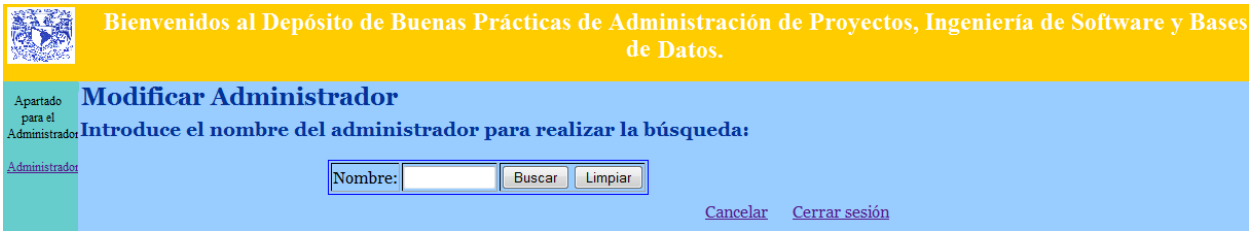



Figura 7.3.1-7 Búsqueda Modificar Administradores

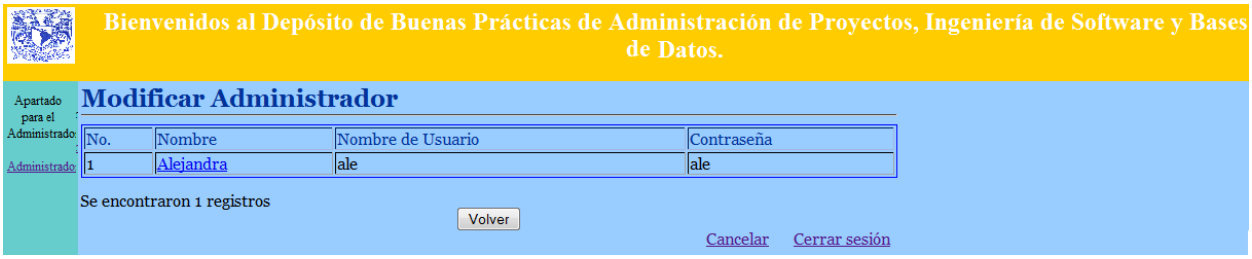



Figura 7.3.1-8 Respuesta a la Búsqueda Modificar Administradores

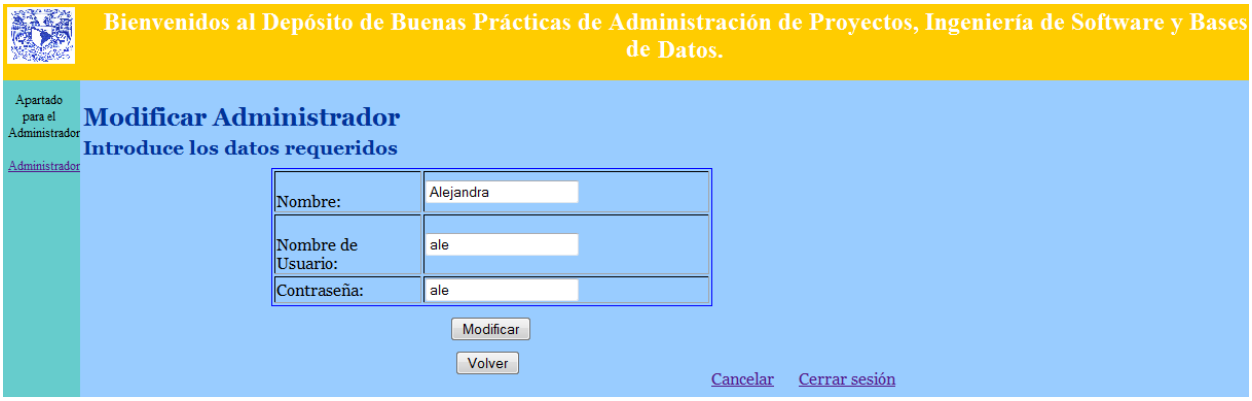



Figura 7.3.1-9 Formulario Modificar Administradores

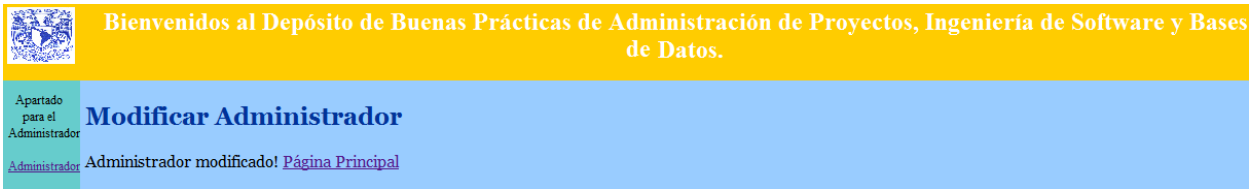



Figura 7.3.1-10 Respuesta a Modificar Administradores

Al escoger *Consultar Administrador* aparecerá la Figura 7.3.1-11 **Formulario Consultar Administrador**, aquí se pide el nombre del administrador a consultar y se presiona *Buscar* y aparecerá la Figura 7.3.1-12 **Búsqueda Formulario Consultar Administrador**.

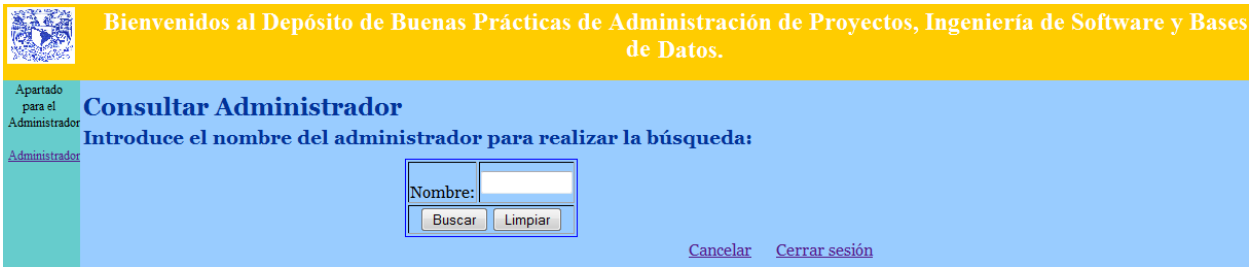



Figura 7.3.1-11 Formulario Consultar Administrador

Aquí se muestra un listado con los nombres de los administradores que existen y para consultar sus datos es necesario dar clic *en el nombre escogido* y aparecerá la Figura 7.3.1-13 **Respuesta Formulario Consultar Administrador**.

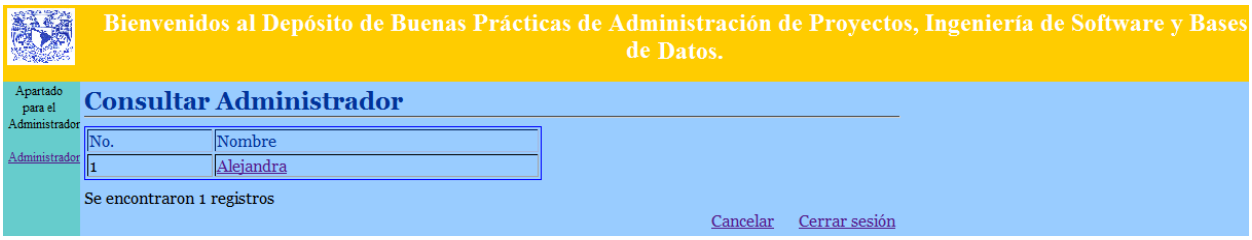



Figura 7.3.1-12 Búsqueda Formulario Consultar Administrador

Aquí es en donde se detalla la información del administrador consultado.

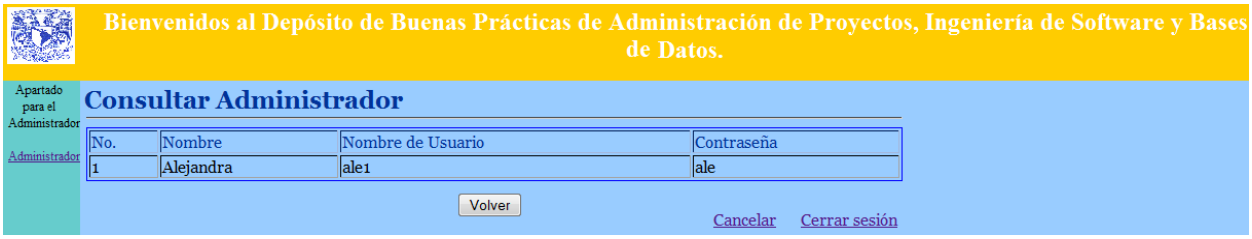



Figura 7.3.1-13 Respuesta Formulario Consultar Administrador

Al escoger *Volver* se muestra la Figura 7.3.1-12 **Búsqueda Formulario Consultar Administrador** y si se escoge *Cerrar se*sión se muestra la Figura 7.1-1 **Página Principal.**

### 7.3.2. Administrar Buenas Prácticas:

Dentro de *Administrar Buenas Prácticas* está *Revisar Buena Práctica* y dentro está la opción *Incluir Buena Práctica*.

El administrador antes de incluir una Buena Práctica debe revisar si es o no candidata a estar en el depósito. Entonces, al escoger *Revisar Buenas Prácticas* aparece un listado con las Buenas Prácticas propuestas como se muestra en la Figura 7.3.2-1 **Búsqueda Buenas Prácticas**, que es un listado de las Buenas Prácticas que se han propuesto. Al escoger un título se muestra la Figura 7.3.2-2 **Buena Práctica**

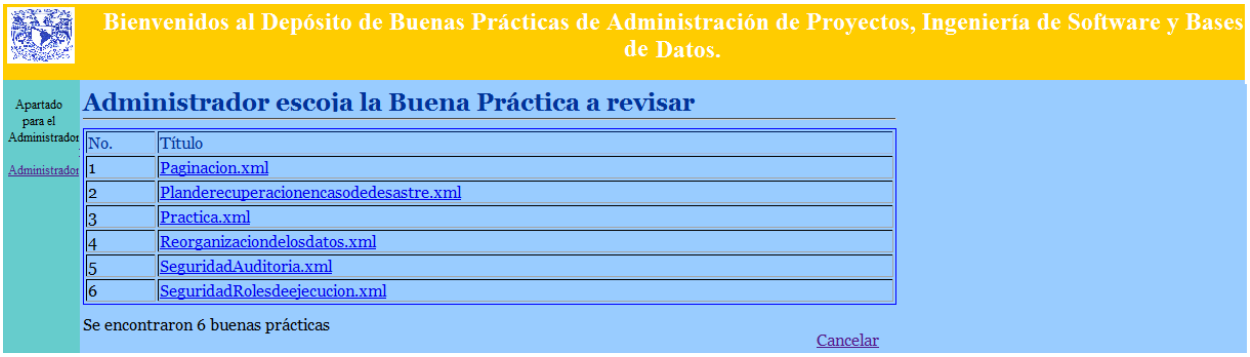



Figura 7.3.2-1 Búsqueda Buenas Prácticas

Lo que se muestra es toda la información referente a la Buena Práctica, si el administrador decide que sí se almacene, escogerá la opción *Incluir Buena Práctica* y se mostrará la Figura 7.3.2-3 **Mensaje de envío**.

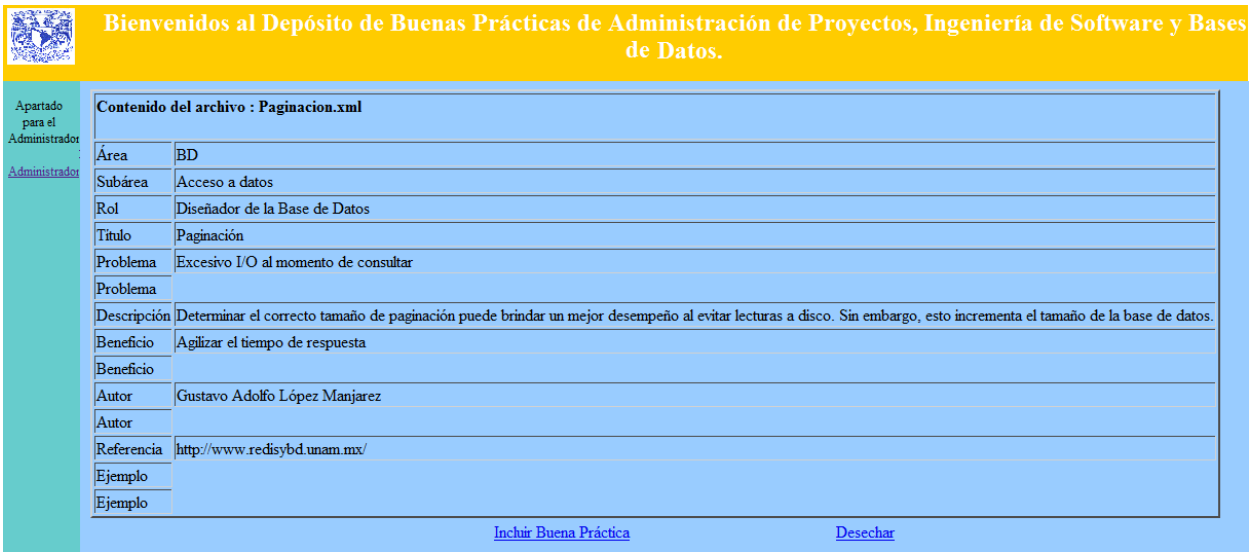



Figura 7.3.2-2 Buena Práctica

Simplemente se indica que el archivo fue enviado a la base de datos con éxito, pero si eligió eliminar la Buena Práctica escogerá la opción *Desechar* y aparecerá otro mensaje como el de la Figura 7.3.2-4 **Mensaje de Eliminación**.

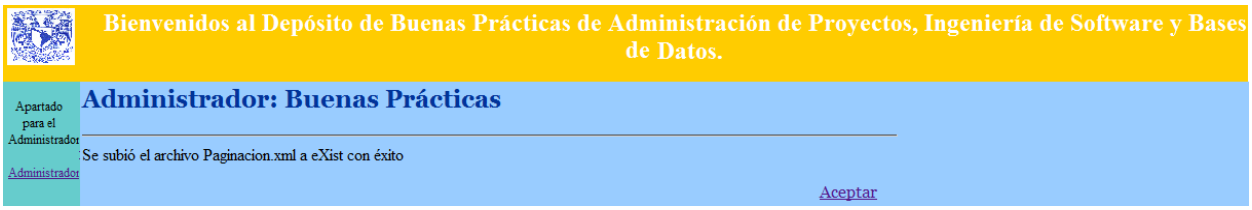



Figura 7.3.2-3 Mensaje de envío

Se indica que el archivo fue eliminado con éxito.

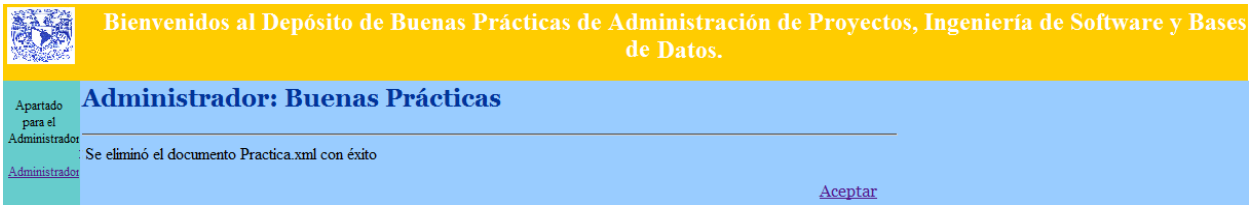



Figura 7.3.2-4 Mensaje de Eliminación

### 7.3.3. Cerrar Sesión:

Al elegir esta opción aparecerá la Figura 7.1-1 **Página Principal**.

# Conclusiones

## Logros Alcanzados

Se identificaron un conjunto de buenas prácticas que pueden ser usadas por las organizaciones de desarrollo de software para mejorar en la madurez en el proceso de construcción de software. Para poder identificarlas, almacenarlas y consultarlas, se definió una plantilla que estandariza el contenido de cada práctica y su ámbito de uso.
Se clasificaron las buenas prácticas en tres áreas que son: ingeniería de software, bases de datos y administración de proyectos.

Además se construyó un depósito para que cada organización almacene y consulte las buenas prácticas.

En el depósito las buenas prácticas pueden ser consultadas mediante los siguientes criterios: Área, rol, subárea, título, problema y autor.

Al hacer búsquedas por los distintos criterios, el depósito queda ordenado, pues al realizar una consulta se mostrarán listados, pero sólo del criterio que se pidió, y de este listado se podrá escoger uno.

Las buenas prácticas se muestran en tablas, donde se ve claramente a cada uno de sus elementos, los cuales están divididos para evitar confusiones sobre donde empieza o termina cada elemento.

Para guardar y administrar estos documentos XML se hizo uso de bases de datos nativas XML dado que es un poco más transparente el proceso, evitando el mapeo innecesario de elementos a campos y/o tablas de una base de datos relacional.

Y, por otro lado, se definió una base de datos relacional para la administración de usuarios (administradores), pues es más práctico y seguro.

El administrador será quien decida si la buena práctica es o no enviada al depósito.

## Aportaciones

Se realizó un análisis para clasificar las buenas prácticas, y se generó una plantilla con elementos necesarios para la organización y clasificación de las buenas prácticas.

Se generó un depósito con buenas prácticas de diferentes temas y clasificadas.

Derivado de la construcción de la plantilla y de la implementación del sistema que se pondrá en Internet para que los especialistas puedan accesar, consultar e incluir las buenas prácticas, se puede concluir que va a ver una retroalimentación entre los especialistas para la generación de otras buenas prácticas y se adopten otras derivadas de otros especialistas, de esta manera la comunidad se verá beneficia ya que el sistema al estar en Internet estará abierto para todos.

## Trabajos Futuros

- Para un trabajo futuro se dejo abierto que las buenas prácticas se aporten de otras formas a la mostrada en este trabajo de tesis, como por ejemplo mediante un archivo ya hecho.
- También se deja pendiente que estos archivos sean validados en la base de datos, pues como en este trabajo se origina el archivo en la aplicación pues no es necesaria la validación.
- Se puede extender el depósito para que además de almacenar buenas prácticas, también se agregue un apartado para patrones.
- El uso del depósito por algunas organizaciones para demostrar su utilidad o descubrir oportunidades de mejora.
- Incluir más tipos de buenas prácticas.
- Más filtros de búsqueda y clasificación.

# Anexo A

## Plantillas de Buenas Prácticas

Se presenta un depósito inicial de buenas prácticas de cada área.

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Administración de la Comunicación |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Matriz de comunicación |
| Problema | El número total de canales de comunicación en un grupo de n personas es n(n-1)/2, es decir es muy difícil saber con quién se puede comunicar, para qué y cómo. Esta matriz ayuda a resolver este problema estableciendo los canales de comunicación desde la planeación del proyecto. |
| Descripción | Primero se determinan a todos los colaboradores que participan en el proyecto(n personas), posteriormente se elabora una matriz de n por n con todas las personas y se determina en cada celda el tipo de comunicación y los motivos para comunicarse para los fines del proyecto.<br>La matriz de comunicación tiene que ser difundida y validada por todos los involucrados |
| Beneficio | Brinda mucha claridad sobre la organización del proyecto |
| Beneficio | Disminuye el número de incidencias por una mala comunicación |
| Autor | Cristina Múzquiz Fragoso |
| Ejemplo | Fue aplicada en varios proyectos de la Subdirección de Sistemas |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Asignación y Administración de Recursos Humanos |
| Rol | Responsable de Recursos Humanos y Ambiente de Trabajo |
| Título | Distribución física de los miembros de un equipo de trabajo |
| Problema | Falta de comunicación, integración y coordinación de grupos de trabajo |
| Descripción | Uno de los factores más críticos dentro de un proyecto es la integración y comunicación entre el equipo de trabajo. Cuando éste se encuentra disperso la comunicación generalmente se da en reuniones o por medio de mecanismos como son mensajeros y correo electrónicos, dilatando el intercambio de información e incluso generando malos entendidos. En cambio, cuando existe cercanía entre los colaboradores, la información más rápidamente, creando un ambiente de apertura y pertenencia al proyecto, lo que genera un mayor compromiso y productividad. |
| Beneficio | Comunicación efectiva e inmediata. Refuerza el enfoque a procesos y proyectos, promoviendo la apertura y el trabajo en equipo, rompiendo así las barreras jerárquicas y físicas. Mayor productividad. Genera compromiso e identidad con el proyecto. |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Estimación de Proyectos |
| Rol | Responsable de Gestión de Proyectos |
| Título | Work Breakdown (WBS) o Estructura Desglosada de Trabajo (EDT) |
| Problema | En la mayoría de los proyectos es difícil tener una visión completa del alcance del mismo que involucre los entregables (internos y externos) y actividades más detalladas, por eso el valor del WBS que sirve para representar de forma gráfica y entendible el alcance total del proyecto. |
| Descripción | Es una técnica que define y organiza el alcance de un proyecto, es una representación jerárquica en estructura de árbol (tipo organigrama). El WBS es creado por el equipo de trabajo del proyecto, lo que también contribuye a tener un entendimiento en común sobre la forma en que se abordará el proyecto. |
| Beneficio | Han clarificado el alcance tanto al interior del equipo de trabajo de la SS así como por parte de los clientes o usuarios. |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Estimación de Proyectos |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Costeo de Proyectos |
| Problema | Estimaciones no fundamentadas y realizadas de una manera empírica. |
| Descripción | La estimación de costos de un proyecto se realiza tomando en cuenta el tiempo, el esfuerzo en recursos humanos y económicos, así como los insumos necesarios para el cumpliento del objetivo del proyecto. |
| Beneficio | Tener registro de las entradas y salidas de todos los recursos financieros para poder tener un control de los mismos.<br>Administrar eficientemente los recursos económicos. |
| Beneficio | Evidencias claras y objetivas del uso de los recursos humanos.<br>Contar con una metodología que permite obtener de manera estandarizada los costos de los proyectos. |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Gestión de seguimiento a proyectos |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Reunión de Inicio o Kick.off |
| Problema | Aunque parezca ilógico, en varias ocasiones y dependiendo del rol, es difícil saber el día en que inicia un proyecto, las personas que están involucradas y lo que se espera de ellas. La reunión de kick-off tiene como objetivo dar a conocer los elementos principales del proyecto después de una fase de aprobación y bosquejo del alcance. |
| Descripción | En un ámbito descentralizado, en proyectos complejos o con colaboradores de diferentes instituciones o ciudades, las reuniones de Kick-off son una gran oportunidad para conocer a los demás colaboradores y el plan general de |

| | |
|---|---|
| | proyecto. |
| Beneficio | Es el primer indicio de formalidad de la administración del proyecto.<br>Difundir los detalles del proyecto. |
| Beneficio | Conocer en persona a todos los colaboradores del proyecto, conocer sus expectativas y obtener su compromiso. |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Ejecución del Proyecto |
| Rol | Responsable de Gestión de Proyectos |
| Título | Administración de la Comunicación |
| Problema | No saber determinar que tipo de información enviar, a quien mandarla, cuando o que tan frecuentemente y como manejarla o formatearla para que sea efectiva. |
| Descripción | Se deben monitorear continuamente las actividades de comunicación para asegurarse que el proceso está establecido y es mantenido de manera efectiva. |
| Beneficio | Establecer y mantener expectativas comunes de las prácticas de administración. |
| Referencia | http://www.tidap.gob.mx/Presentaciones/talleres/t_AnaBrise%F1o.pdf |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Ejecución del Proyecto |
| Rol | Responsable de Gestión de Proyectos |
| Título | Comunicación efectiva |
| Problema | Cada quién entiende una cosa y no se realiza lo que se pide. |
| Descripción | Preparar el mensaje anticipadamente, escribir el mensaje en términos que el receptor pueda traducirlo fácilmente, y ser consistente en el uso de los componentes verbales y no verbales. Mandar el mensaje oportunamente. Considera los valores, sentimientos y personalidad del receptor. Escoger un medio apropiado de transmisión. Escuchar atentamente la retroalimentación del receptor. Asegurarse que el mensaje ha sido entendido, aceptado y se está actuando sobre él. |
| Beneficio | Todos los involucrados entiendan lo mismo y se promueva la comunicación efectiva. |
| Referencia | PMI Standards Committee. A guide to the Project Management Body of Knowledge (PMBOK) Project Management Institute. Version 3. |
| Ejemplo | Las manos apoyan nuestra comunicación y permiten confirmar lo que decimos. Frotarse las manos significa una expectativa positiva, un buen entendimiento entre las partes. |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Administración estratégica y cuantitativa |

| | |
|---|---|
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Enfoque en el control cuantitativo |
| Problema | Planificar un proyecto de mejora para un producto-proceso software que es desconocido para los encargados de mejorar y que es de gran volumen no es trivial; justificar un programa de mejora no es fácil |
| Descripción | La manera más rápida de tener visibilidad sobre el estado y profundidad del problema es recogiendo métricas, generalmente de producto (más sencillas y "baratas") |
| Beneficio | Identificar síntomas para poder aplicar soluciones correctas y tener argumentos para defender y concientizar sobre la mejora. |
| Autor | Javier Garzás |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Administración estratégica y cuantitativa |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Automatización de la captura masiva de datos |
| Problema | El costo de realizar mediciones software de manera puntual puede ser costoso |
| Descripción | Automatizar todo lo posible (el análisis estático de código, la compilación del proyecto, el lanzamiento de pruebas, el control de incidencias, etc), en la actualidad es posible y no es costoso |
| Beneficio | Ahorro de tiempo en la captura de datos. |
| Autor | Javier Garzás |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Administración estratégica y cuantitativa |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Síntesis de datos |
| Problema | Automatizar genera mucho volumen de información (p.e hasta más de 500 mediciones por aplicación, fuente y cada noche) |
| Descripción | Síntesis de datos, periodicidad y escalado. Si se pueden tener medidas de manera periódica, con poco costo, se podrá controlar los efectos de las mejoras ("que arreglar una cosa no rompa otra cosa"), controlar de manera remota. Nunca se debe perder de vista el objetivo final de todos los datos recopilados. |
| Beneficio | Generar tendencias, dejar claro el objetivo de la medición y permite llegar a la alta dirección y a otra áreas de la empresa |
| Autor | Javier Garzás |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Administración por Objetivos |
| Rol | Responsable de Proceso |

| | |
|---|---|
| Título | Administración por objetivos |
| Problema | No se alcanzan los objetivos planteados en tiempo y forma |
| Descripción | Es una técnica de dirección de esfuerzos a través de la planeación y el control administrativo basada en el principio de que, para alcanzar resultados, la organización necesita antes definir en qué negocio está actuando y a dónde pretende llegar. Inicialmente se establecen los objetivos anuales de la empresa, formulados sobre la base de un plan de objetivos a largo plazo (que pueden ser quinquenales o decenales), y los objetivos de cada gerente o departamento, con base en los objetivos anuales de la empresa. |
| Beneficio | A través de ella los superiores y los subordinados, conjuntamente, definen aspectos prioritarios; |
| Referencia | http://www.elprisma.com/apuntes/administracion_de_empresas/administracionporobjetivos/default.asp |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Administración por Objetivos |
| Rol | Responsable de Proceso |
| Título | Criterios para la selección de objetivos |
| Problema | Se requiere establecer objetivos (resultados), en un determinado periodo y en términos cuantitativos, dimensionando las metas. |
| Descripción | Los criterios para la selección de objetivos se deben establecer de acuerdo con las prioridades y con su contribución al alcance de los resultados claves de la empresa.<br>Algunos criterios son:<br>a)buscar las actividades que tengan mayor impacto sobre los resultados;<br>b)el objetivo debe ser específico en cuanto a los datos concretos: qué, cuánto, cuándo. Los resultados esperados deben enunciarse en términos cuantificables y bastante claros;<br>c)centrar los objetivos en el trabajo y no en el hombre;<br>d)detallar cada objetivo en metas derivadas;<br>e)utilizar un lenguaje comprensible para los gerentes;<br>f)mantenerse dentro de los principios de la administración. Concentrarse en los propósitos vitales del negocio y no dispersarse en actividades secundarias;<br>g)el objetivo debe indicar los resultados por alcanzar, pero no debe limitar la libertad para escoger los métodos. Debe indicar cuánto, pero no indicar cómo;<br>h)el objetivo debe ser difícil de alcanzar; exigir un esfuerzo especial, pero no al punto de ser imposible;<br>i)el objetivo debe representar una tarea suficiente para todo el ejercicio fiscal de la empresa;<br>j)el objetivo debe tener alguna relación remota con el plan de utilidades de la empresa, que generalmente es el objetivo final. |
| Beneficio | Hacer compatibles los objetivos que están en conflicto |
| Referencia | http://www.elprisma.com/apuntes/administracion_de_empresas/administracionporobjetivos/default.asp |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Administración del Portafolio |

| | |
|---|---|
| Rol | Responsable de Gestión de Proyectos |
| Título | Administración de Portafolio de Proyectos |
| Problema | No hay optimización en la distribución de recursos |
| Descripción | Hay siete aspectos básicos para determinar la disposición organizacional.<br>La organización debe contar con una estrategia coherente.<br>La organización debe saber cómo administrar proyectos.<br>La organización debe conocer qué proyectos tiene, es decir, su inventario.<br>Los proyectos de la organización deben estar debidamente documentados.<br>La información de los proyectos de la organización debe ser confiable. La organización debe conocer quién se encuentra disponible para trabajar en los proyectos y cuándo.<br>La organización debe contar con una Oficina de Administración de Proyectos (OAP). |
| Beneficio | Integración de la información de los proyectos para Determinar el grado de disposición y nivel de madurez de la organización para emprender la administración de portafolio de proyectos (APP). |
| Referencia | http://greatcomments.com.mx/home/?p=86 |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Oficina de Proyectos |
| Rol | Grupo Directivo |
| Título | Modelo de la oficina de proyectos |
| Problema | No contar con un modelo específico para implementar una oficina de proyectos |
| Descripción | El proceso de puesta en marcha de una oficina de proyectos debe definir un modelo de la misma, es decir;<br>Fuerte: En este modelo la organización PMO opera como una organización central de la administración de los proyectos. Este tipo de modelos maneja la cartera de proyectos (portafolio) a su interior y los gerentes de proyecto dependen del administrador de la oficina.<br>Consultivo/Staff: En este modelo la oficina de proyectos entrega capacitación, acompañamiento, coaching y mentoring a los gerentes de proyecto de las diferentes unidades de negociación. La ventaja de este modelo es que es menos invasiva a la organización.<br>Mixto: En el cual la PMO toma control de algunos proyectos (más críticos, en problemas, u otro criterio) y otros quedan en las unidades de negocio utilizando un modelo más consultivo. |
| Beneficio | Mejor resultados de los proyectos, más aún cuando los niveles de complejidad aumentan. |
| Referencia | http://www.alejandrobarros.com/content/view/120668 |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Oficina de Proyectos |
| Rol | Grupo Directivo |

| | |
|---|---|
| Título | Implementación de una Oficina de Proyectos |
| Problema | El responder éstas preguntas:<br>¿Qué tan alineados están mis proyectos con los objetivos estratégicos del negocio?<br>¿Tengo un entendimiento adecuado del valor ganado de los proyectos de mi organización?<br>¿Cuáles son los Riesgos claves de mis proyectos? |
| Problema | ¿Cómo interactúan (se complementan) los proyectos en mi organización?<br>¿Cómo mi organización prioriza los proyectos?<br>¿Cuál es el nivel y capacidad de absorción de mi organización al cambio?<br>¿Cuáles y por qué son los proyectos con bajo desempeño en mi organización? |
| Descripción | Para abordar este tema es a través del establecimiento de una organización estructurada de gestión de proyectos denominada “oficina de proyectos (PMO)”. Una estrategia para establecerla al interior de la organización, es la siguiente: En primer lugar debemos definir el tipo de PMO que queremos desarrollar. El proceso de puesta en marcha de PMO debe además definir el modelo de la misma. El proceso de establecimiento debe considerar todas las componentes necesarias: Enfoque metodológica, Herramientas, Estructura Organizacional, Gestión del Conocimiento. En resumen una buena forma de mejorar el gobierno corporativo de las compañías de tecnologías de información en cualquier organización es establecer una oficina de proyectos con modalidad que más se adecua a la cultura organizacional particular. |
| Beneficio | Mejorar el gobierno corporativo de las compañías de tecnologías de información en cualquier organización |
| Referencia | http://www.alejandrobarros.com/content/view/120668 |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Oficina de Proyectos |
| Rol | Grupo Directivo |
| Título | Roles para desarrollar la oficina de proyectos |
| Problema | Cómo definir el tipo de oficina de proyectos que se quiere desarrollar. |
| Descripción | La oficina de proyectos se quiere como una organización con un rol consultivo, rol de gestión de conocimiento, o bien, de definición y gestión de estándares.<br>Consultivo. Consultoría a proyectos en problemas, asistencia en la puesta en marcha de buenas prácticas, difundir las lecciones aprendidas, auditorías de proyectos, Apoyo a las unidades de negocios en el diseño de proyectos, selección de proveedores y en el proceso de desarrollo de los proyectos.<br>Gestión del Conocimiento. Recopilar el conocimiento organizacional y estructurar el aprendizaje, identificar y documentar las buenas prácticas de la organización, generar y proveer acceso a repositorios de conocimiento, generar material de entrenamiento, capacitar a los gerentes de proyecto, desarrollar repositorio documental (libros, papers, journals, conferencias y |

| | |
|---|---|
| | otros)<br>Gestión de Estándares. Definir los estándares del proceso de proyectos, crear las herramientas (artefactos) para utilizar en las diferentes disciplinas de los proyectos: estimación, diseño, seguimiento y control, definición y puesta en marcha de los tableros de mando de los proyectos. |
| Beneficio | Tener definido el rol para saber qué actividades se llevarán acabo. |
| Referencia | http://www.alejandrobarros.com/content/view/120668 |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Dirección de Proyectos |
| Rol | Responsable de Gestión de Proyectos |
| Título | Aislarse del equipo |
| Problema | En algunas organizaciones, tener el título de director de proyecto da algunos derechos que el resto del equipo no goza. A veces, el título se considera suficiente para tener una oficina propia, o por lo menos un espacio de trabajo más grande. |
| Descripción | Acercarse físicamente al resto del equipo de trabajo, tener a los compañeros tan cerca como sea posible es importante. Entender los problemas del equipo y ser considerado “el director apropiado” es fundamental y para ello se tiene que estar en las mismas circunstancias que el equipo |
| Beneficio | Estar en constante comunicación ayuda a mitigar riesgos en el proyecto. |
| Referencia | http://www.tufuncion.com/direccion_proyecto |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Dirección de Proyectos |
| Rol | Responsable de Gestión de Proyectos |
| Título | Empleo de técnicas de motivación |
| Problema | Empleo de técnicas de motivación inadecuadas. |
| Descripción | Cada tipo de persona debe ser motivada de una manera diferente. Los programadores y los encargados tienen diferentes naturalezas y por lo tanto diferentes maneras de ser motivados. Los encargados valoran mucho la opinión del resto del equipo, mientras que los programadores tienden a fijarse más en lo práctico y funcional, y valoran las cosas que pueden utilizar y le den cierta ventaja en su día a día. Los programadores ven ell tipo de recompensa de los encargados superficiales y triviales. |
| Beneficio | Que el ambiente de trabajo sea agradable y por lo tanto, los trabajadores tengan un mejor desempeño. |
| Referencia | http://www.tufuncion.com/direccion_proyecto |
| Ejemplo | Una buena recompensa para el encargado sería recibir una placa para exhibir en su pared.<br>Y para el programador algo útil sería un segundo monitor, algo más de memoria RAM, una CPU más rápida, periféricos nuevos, o una silla más cómoda serán premios realmente gratificantes para ellos. |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Generación de Software |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Iniciar las pruebas en etapas tempranas dentro del proyecto |
| Problema | Anteriormente las pruebas de software se consideraban sólo una actividad que realizaba el programador para encontrar fallas en sus productos. |
| Descripción | Que las especificaciones de pruebas se realicen al mismo tiempo que el diseño de software; la propuesta es iniciar el análisis del testware junto con el análisis del software |
| Beneficio | Se identifica qué se quiere y cuáles son los resultados esperados (criterios de aceptación)<br>Ejecutar las pruebas tan pronto como el software está listo.<br>No sólo descubrir errores, sino evitarlos. |
| Autor | Elsa Ramírez |
| Referencia | http://www.sg.com.mx/content/view/684/99999999/ |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Generación de Software |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Capacitar a especialistas responsables de las pruebas |
| Problema | Creer que sólo el programador se dedica a probar |
| Descripción | Crear conciencia sobre la importancia de las pruebas y tener un equipo de personas dedicadas a esta actividad que puedan integrarse a un proyecto y sean responsables de su calidad. |
| Beneficio | Las pruebas sólo deben obtener un producto práctico con la calidad y funcionalidad requeridas. |
| Autor | Elsa Ramírez |
| Referencia | http://www.sg.com.mx/content/view/684/99999999/ |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Gestión de Requerimientos |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Validar requisitos |
| Problema | Para mejorar el éxito de los proyectos es crítico que se validen los requisitos de forma adecuada. |
| Descripción | Utilizar técnicas como la revisión de requerimientos (los requerimientos son analizados sistemáticamente por un equipo de revisores, pueden ser formales o informales, los conflictos, contradicciones, errores y omisiones deben señalarse y registrarse formalmente) y/o la construcción de prototipos (Versión inicial de un sistema que se desarrolla para dar a los usuarios una impresión completa de las capacidades del sistema. Por lo tanto, el prototipo ayuda a establecer y validar requerimientos). |

| | |
|---|---|
| Beneficio | Entregar al usuario lo que espera y ayudarlo a descubrir lo que necesita.<br>Comprender y describir el problema, acordar sobre la naturaleza del problema. |
| Referencia | http://www.scribd.com/doc/6932030/Administracion-de-Requerimientos |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Gestión de Requerimientos |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Alcance y visión del proyecto |
| Problema | No se cuenta con un documento donde definir, verificar y controlar el alcance y visión del proyecto. |
| Descripción | Documentar el alcance (identificar los productos de software, explicar que hará y que no hará cada uno, describir la aplicación) y visión (Describir que contiene la especificación de requerimientos de software y explicar cómo está organizada la especificación de requerimientos de software) del proyecto. |
| Beneficio | Permitirá tener un mejor entendimiento de los requisitos y asegurará que todas las personas involucradas en el proyecto trabajen hacia la misma meta. |
| Referencia | http://www.scribd.com/doc/6932030/Administracion-de-Requerimientos |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Gestión de Requerimientos |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Verificar requisitos |
| Problema | No se puede mejorar el éxito de los proyectos, pues no se validan los requisitos de forma adecuada. |
| Descripción | El usuario final añade criterios de aceptación para cada requisito. Además apoya el hecho de que los requisitos han de ser correctos antes de que sean entregados a los diseñadores y desarrolladores. |
| Beneficio | Evitar fugas de requisitos. Los requisitos, algunas veces, aparecven en las especificaciones sin que nadie realmente sepa de donde viene o qué valor añaden al producto.<br>Evitar que los errores conduzcan a problemas posteriores en el diseño e implementación del sistema. |
| Referencia | http://www.scribd.com/doc/6932030/Administracion-de-Requerimientos |

| | |
|---|---|
| Área | Administración de Proyectos |
| Subárea | Gestión de Requerimientos |
| Rol | Responsable de la Administración del Proyecto Específico |
| Título | Priorizar requisitos |
| Problema | El software resultante puede no satisfacer a los usuarios, las interpretaciones múltiples de los requerimientos pueden causar desacuerdos entre clientes y |

| | |
|---|---|
| | desarrolladores. Gastar tiempo y dinero construyendo el sistema erróneo |
| Descripción | Hacer una lista de los requisitos y decidir qué requisito se debe realizar primero tomando en cuenta al usuario final y a los desarrolladores. |
| Beneficio | Determinar aquellos que se deberían cumplir en la primera versión o producto y aquellos que pueden llevarse a cabo en sucesivas versiones |
| Referencia | http://www.scribd.com/doc/6932030/Administracion-de-Requerimientos |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Modelado y Diseño |
| Rol | Diseñador de la Base de Datos |
| Título | Normalización y Rendimiento |
| Problema | Incongruencia entre el esquema de la base de datos y el modelo de datos |
| Descripción | Asegurarnos de normalizar los datos cuando menos hasta la tercera forma normal, sin comprometer el rendimiento |
| Beneficio | Mejor tiempo de respuesta y congruencia del modelo físico con las bondades del manejador |
| Autor | María del Pilar Ángeles |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Modelado y Diseño |
| Rol | Programador |
| Título | Uso de Diccionario de Datos |
| Problema | Problemas de Integridad e inconsistencia |
| Descripción | Usar tipos de datos definidos por usuario si una columna en particular se repite en varias tablas de tal forma que el dominio será consistente en todas esas tablas |
| Beneficio | Optimización y estandarización del dominio de datos a utilizarse en los atributos y por ende se minimiza el riesgo de capturar valores incongruentes en éstos |
| Autor | María del Pilar Ángeles |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Modelado y Diseño |
| Rol | Diseñador de la Base de Datos |
| Título | Restricción de dominio según requerimientos |
| Problema | Problemas de integridad e inconsistencia |
| Descripción | Definir aquellos tipos de datos cuya longitud y precisión (dominio) se ajuste mejor el rango de valores requeridos válidos |

| | |
|---|---|
| Beneficio | Optimización y estandarización del dominio de datos a utilizarse en los atributos |
| Autor | María del Pilar Ángeles |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Modelado y Diseño |
| Rol | Diseñador de la Base de Datos |
| Título | Manejo de Textos y BLOB |
| Problema | Tiempo de acceso inaceptable limitada o nula manipulación de datos |
| Descripción | Un manejador de bases de datos ha sido programado para manipular atributos con ciertos tipos de datos. Posteriormente, se permitió definir texto, BLOB XML, por ejemplo. Sin embargo, no es posible un manejo óptimo de dichos tipos de datos. Se recomienda por tanto evitar guardar estos tipos de datos en la base de datos y guardar en su lugar la ruta de donde se encuentra a fin de que el software correspondiente lo manipule correctamente. |
| Beneficio | Reducción de tiempo de respuesta |
| Autor | María del Pilar Ángeles |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Programación SQL |
| Rol | Programador |
| Título | Manejo de recursos |
| Problema | Tiempo de ejecución, uso de memoria excesivos, poca concurrencia. |
| Descripción | No hacer llamadas a funciones repetidamente dentro de un programa. Hacer la llamada una vez y guardarlo en una variable |
| Beneficio | Utilización optima de memoria y procesamiento, mejor tiempo de respuesta |
| Autor | María del Pilar Ángeles |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Diseño y modelado |
| Rol | Diseñador de la Base de Datos |
| Título | Capacity Planning |
| Problema | Dimensionamiento incorrecto de la base de datos |
| Descripción | Dimensionar en base al crecimiento paulatino del sistema los recursos necesarios |
| Beneficio | Escalabilidad de la solución |
| Autor | Gustavo Adolfo López Manjarez |

| | |
|---|---|
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Modelado de datos |
| Rol | Diseñador de la Base de Datos |
| Título | Nomenclatura |
| Problema | Falta de coherencia entre diferentes procesos y desarrollos |
| Descripción | Establecer un diccionario de datos común a todos los DBAs y desarrolladores. Establecer estandares entre las distintas entidades involucradas |
| Beneficio | Agilidad de mantenimiento y facilidad en la documentación |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Diseño físico de la base de datos |
| Rol | Diseñador de la Base de Datos |
| Título | Normalización |
| Problema | Tiempos de respuesta inaceptables |
| Descripción | Implementación de un sistema de BD normalizado hasta por lo menos la 2FN y así evitar problemas de rendimiento. Controlar la integridad de los datos usando procesos de Integridad declarativa o procedural o algún otro método efectivo |
| Beneficio | Rapidez en el tiempo de respuesta de consultas y procesos |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Seguridad en Bases de Datos Tecnología SGBD. Seguridad. Encriptación |
| Rol | Diseñador de la Base de Datos |
| Título | Seguridad. Roles de ejecución |
| Problema | Todos ven toda la información |
| Descripción | Restringir el acceso a datos sensibles al negocio mediante el uso de roles de ejecución, GRANT o REVOKE. Establecer roles bien delimitados para funciones de mantenimiento, respaldos, etc. |
| Beneficio | Garantizar la integridad de la información |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |

| | |
|---|---|
| Subárea | Tecnología SGBD. Seguridad. Encriptación |
| Rol | Diseñador de la Base de Datos |
| Título | Seguridad. Auditoría |
| Problema | Información (extremadamente sensible) visible. |
| Descripción | Implementar (preferentemente a nivel atómico) (columna) algún proceso ágil, no intrusivo de encriptación de información. |
| Beneficio | Ocultar información relevante a personas o aplicaciones no autorizadas |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Tecnología SGBD. Acceso a la información |
| Rol | Diseñador de la Base de Datos |
| Título | Reorganización de los datos |
| Problema | Posible lentitud en el RDBMS |
| Descripción | Prever la necesidad de reorganizar las tablas dinámicas de la base de datos y su reindexación. Calendarizar las ventanas de tiempo para llevar a cabo esta tarea. Considerar que al reorganizar una tabla, todos los índices asociados a ella deberán ser reconstruidos para poder determinar su uso por parte del optimizador. |
| Beneficio | Promover el acceso rápido a datos |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Indexación |
| Rol | Diseñador de la Base de Datos |
| Título | Desempeño. Indexación |
| Problema | Tiempos de respuesta no óptimos. |
| Descripción | Identificar el mejor método de acceso a la información conociendo el tipo de consultas de la aplicación y determinando los SEARCH ARGUMENTS. Teniendo esta información poder determinar el tipo y cantidad de índices que deberán ser construidos. |
| Beneficio | Acceso rápido a datos |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Acceso a datos |
| Rol | Diseñador de la Base de Datos |
| Título | Paginación |

| | |
|---|---|
| Problema | Excesivo I/O al momento de consultar |
| Descripción | Determinar el correcto tamaño de paginación puede brindar un mejor desempeño al evitar lecturas a disco. Sin embargo, esto incrementa el tamaño de la base de datos. |
| Beneficio | Agilizar el tiempo de respuesta |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Diseño de la BD. Acceso a datos |
| Rol | Diseñador de la Base de Datos |
| Título | El optimizador y sus estadisticas |
| Problema | Eleccion errónea de path de ejecución acentuada con el paso del tiempo |
| Descripción | Entender el funcionamiento de las diferentes pasos, estadísticas, densidades de información que se manejan y que modifican las condiciones iniciales de creacion de los índices y procesos almacenados. Regenerar o recompilar con frecuencia estos objetos en tablas dinámicas ayuda a evitar situaciones de desempeño anómalo |
| Beneficio | Tiempos de repuesta óptimos |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Programación, Consultas |
| Rol | Programador |
| Título | Consultas |
| Problema | Tiempos de respuesta altos debido al uso de cláusulas no óptimas |
| Descripción | Evitar el uso de sentencias SQL que pueden causar TABLE SCAN y que afectan el desempeño del RDBMS, por ejemplo, UNION, el uso de cursores, LIKE, IN, NOT IN, SELECT *, etc. |
| Beneficio | Mejorar el tiempo de respuesta |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Diseño de la BD y mantenimiento |
| Rol | Diseñador de la Base de Datos |
| Título | Generación de respaldos |
| Problema | Fallas en la consistencia de los datos o de SW; fallas de HW. Falta de punto de retorno |
| Descripción | Elaborar en base al conocimiento de la sensibilidad de la información, un |

| | |
|---|---|
| | esquema de generación de respaldos adecuado (completo e incremental) |
| Beneficio | Garantizar la operación continua del negocio |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Diseño de la base de datos. Recuperación de Información |
| Rol | Diseñador de la Base de Datos |
| Título | Alta disponibilidad |
| Problema | Daños físicos que pueden corromper la información de la base de datos |
| Descripción | Implementar en la medida de lo posible alguna solución (de HW y/o SW) que garantice la disponibilidad de la información hasta el ultimo momento previo a la falla. |
| Beneficio | Operación continua del negocio |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Diseño de la base de datos. Recuperación de Información |
| Rol | Administrador de la Base de Datos |
| Título | Plan de recuperación en caso de desastre |
| Problema | Fallas de hardware |
| Descripción | Incorporar una solución de DRP al esquema de manejo de información que garantice la información del negocio incluyendo, en la medida de lo posible, alguna solución que contemple la NO perdida de información que este radicando en memoria |
| Beneficio | Operación continua del negocio |
| Autor | Gustavo Adolfo López Manjarez |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Base de datos Orientada a Objetos |
| Rol | Programador |
| Título | Desajuste por Impedancia |
| Problema | A las limitaciones para convertir objetos en registros y viceversa se llama Desajuste por Impedancia |
| Descripción | En la arquitectura de aplicación, todo el modelo de entidades de negocio queda en objetos que se instancian y se ejecutan en una capa lógica (y quizás física) conocida como Capa de Negocio, ya que las reglas del negocio quedan plasmadas en la conducta de los objetos de esta capa Ahora bien, en lugar de incluir ahí mismo la lógica *CRUD* (por Create, |

| | |
|---|---|
| | Retrieve, Update and Delete) ponemos una Capa de Acceso a Datos cuyos "habitantes" van a ser objetos que implementen *CRUD*. A estos objetos se los conoce como Objetos de Acceso a Datos, o más comunmente DAOs |
| Beneficio | Se desacopla la lógica de lidiar con el Desajuste por Impedancia de la lógica funcional de la aplicación. Evitando así el llamado código spaghetti. |
| Autor | Sun |
| Referencia | http://www.programacion.com/java/tutorial/patrones2/8/ |
| Ejemplo | Se puede ver el código de un DAO de ejemplo para un objeto persistente que representa información de un cliente. CloudscapeCustomerDAO crea un **Transfer Object** Customer cuando se llama al método findCustomer():<br><br>// CloudscapeCustomerDAO implementation of the CustomerDAO interface. This<br>//class can contain all Cloudscape specific code and SQL statements. The client is<br>//thus shielded from knowing these implementation details.<br><br>import java.sql.*;<br><br>public class CloudscapeCustomerDAO implements<br>CustomerDAO {<br><br>public CloudscapeCustomerDAO() { // initialization<br>}<br>// The following methods can use CloudscapeDAOFactory.createConnection()<br>// to get a connection as required<br><br>public int insertCustomer(...) {<br>// Implement insert customer here.<br>// Return newly created customer number or a -1 on error<br>}<br>public boolean deleteCustomer(...) {<br>// Implement delete customer here Return true on success, false on failure<br>}<br>public Customer findCustomer(...) {<br>// Implement find a customer here using supplied argument values as search<br>//criteria Return a Transfer Object if found, return null on error or if not found<br>}<br>public boolean updateCustomer(...) {<br>// implement update record here using data from the customerData Transfer Object<br>// Return true on success, false on failure or error<br>}<br>public RowSet selectCustomersRS(...) {<br>// implement search customers here using the supplied criteria. Return a RowSet.<br>}<br>public Collection selectCustomersTO(...) {<br>// implement search customers here using the supplied criteria.<br>// Alternatively, implement to return a Collection of Transfer Objects.<br>}<br>...<br>}<br>Abajo podemos ver el código para utilizar este DAO:<br>...<br>// create the required DAO Factory<br>DAOFactory cloudscapeFactory =<br>DAOFactory.getDAOFactory(DAOFactory.DAOCLOUDSCAPE); |

| | |
|---|---|
| | // Create a DAO<br>CustomerDAO custDAO =<br>  cloudscapeFactory.getCustomerDAO();<br><br>// create a new customer<br>int newCustNo = custDAO.insertCustomer(...);<br><br>// Find a customer object. Get the Transfer Object.<br>Customer cust = custDAO.findCustomer(...);<br><br>// modify the values in the Transfer Object.<br>cust.setAddress(...);<br>cust.setEmail(...);<br>// update the customer object using the DAO<br>custDAO.updateCustomer(cust);<br><br>// delete a customer object<br>custDAO.deleteCustomer(...);<br>// select all customers in the same city<br>Customer criteria=new Customer();<br>criteria.setCity("New York");<br>Collection customersList =<br>  custDAO.selectCustomersTO(criteria);<br>// returns customersList - collection of Customer Transfer Objects. iterate through this<br>//collection to get values.<br>... |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Operación Continua de las Bases de Datos |
| Rol | Diseñador de la Base de Datos |
| Título | Continuidad del Negocio |
| Problema | Interrupción de la Operación del Negocio por Bases de Datos No Disponibles |
| Descripción | Creación de Sistemas de Alta Disponibilidad y Sitios de Recuperación de Desastres. |
| Beneficio | Garantizar la continuidad de la operaciones de la organización, con impacto mínimo en pérdida de información y tiempo de no disponibilidad del servicio. |
| Autor | Héctor Bautista Vázquez |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Protección de la Información de las Bases de Datos |
| Rol | Diseñador de la Base de Datos |
| Título | Seguridad de la Información |
| Problema | Violación y uso indebido de la información que se encuentra en las bases de datos. |
| Descripción | Implementación de esquemas de encriptación de información en las bases de datos. |
| Beneficio | Proteger la información contenida en las bases de datos de las organizaciones, permitiendo el acceso a ésta solamente a las personas con |

| | |
|---|---|
| | los privilegios adecuados. |
| Autor | Héctor Bautista Vázquez |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Minería de Datos |
| Rol | Diseñador de la Base de Datos |
| Título | Fases de un proyecto de minería de datos |
| Problema | Cuando se requiere reunir varias áreas como la Estadística, la Inteligencia Artificial, la Computación Gráfica, Bases de Datos y el Procesamiento Masivo, principalmente usando como materia prima las Bases de Datos |
| Descripción | Los pasos a seguir para la realización de un proyecto de minería de datos son siempre los mismos, independientemente de la técnica específica de extracción de conocimiento usada.<br>Fases: Filtrado de datos, Selección de Variables, Extracción de Conocimiento e Interpretación y Evaluación. |
| Beneficio | Predicción automatizada de tendencias y comportamientos, Descubrimiento automatizado de modelos previamente desconocidos. Las técnicas de minería de datos pueden redituar los beneficios de automatización en las plataformas de hardware y software existentes y puede ser implementadas en sistemas nuevos a medida que las plataforma existentes se actualicen y nuevos productos sean desarrollados. |
| Ejemplo | En una empresa se lo usan prediciendo el tamaño de las audiencias televisivas. La Bristish Broadcasting Corporation (BBC) del Reino Unido emplea un sistema para predecir el tamaño de las audiencias televisivas para un programa propuesto, así como el tiempo óptimo de exhibición. El sistema utiliza redes neuronales y árboles de decisión aplicados a datos históricos de la cadena para determinar los criterios que participan según el programa que hay que presentar. La versión final se desempeña tan bien como un experto humano con la ventaja de que se adapta más fácilmente a los cambios porque es constantemente reentrenada con datos actuales. |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Almacenes de Datos |
| Rol | Diseñador de la Base de Datos |
| Título | Asegurarse de que ha siso suministrado un Diccionario de Datos antes de empezar con las etapas fuertes de desarrollo. |
| Problema | Los Data Warehouse que son construidos sin un diccionario de datos resultarán en duplicidad de datos y procedimientos, así como confusiones y problemas con los resultados finales. |
| Descripción | Es común que algunas tareas de desarrollo comiencen mientras otras tareas de análisis están realizándose en paralelo. En este tipo de escenario los desarrolladores juegan a los “detectives” por tiempos indefinidos, mientras realizan la codificación o el mapeo de datos, generando restricciones de |

| | |
|---|---|
| | tiempo. Por eso un diccionario de datos debe estar disponible antes de que cualquier proceso crítico inicie. Que al menos contenga reglas, validaciones y descripciones. Este diccionario debe estar almacenado en la Intranet de la empresa y disponible tanto para usuarios finales como para desarrolladores. |
| Beneficio | Ahorro de tiempo y de confusiones para los programadores. |
| Autor | Bruno Escarcega |
| Referencia | http://www.gravitar.biz/index.php/datawarehouse/mejores-practicas-para-construir-un-data-warehouse/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Almacenes de Datos |
| Rol | Diseñador de la Base de Datos |
| Título | Guardar planes de Consultas, tiempos de ejecución y referencias de rendimiento en la Base de Datos. |
| Problema | No tener dónde idea de donde atacar los problemas. |
| Descripción | Esto se puede realizar fácilmente. Por ejemplo, guardar un tiempo de inicio y un fin de tiempo para cada proceso crítico, puede ser implementado a través de Procedimientos Almacenados o Scripts. |
| Beneficio | Guardar puntos de referencia en la base de datos ayuda a señalar los puntos problemáticos en el rendimiento, ayudando a que el equipo de desarrollo se enfoque en la resolución de estos. Guardar puntos de referencia de los procesos como parte de los meta datos, es una extensión lógica para fortalecerlos. Forma de mejorar mediante el rastreo de la actividad del usuario, identificación de cuellos de botella y queries más usados, obtención de accesos a la base de datos, estadísticas de los queries y mucho más. |
| Autor | Bruno Escarcega |
| Referencia | http://www.gravitar.biz/index.php/datawarehouse/mejores-practicas-para-construir-un-data-warehouse/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Almacenes de Datos |
| Rol | Diseñador de la Base de Datos |
| Título | Optimización de la Base de Datos |
| Problema | Tiempos de respuesta largos |
| Descripción | Comprender la naturaleza de los datos y su relación con las demás entidades. Antes de ejecutar una consulta se puede compilar y ejecutar en modo non-exec verificando el Query Plan. Limitar los resultados de las consultas cuando se esté en fase de prueba. |
| Beneficio | Entender siempre la optimización de la Base de Datos y los planes de Consultas (Query Plans) |
| Autor | Bruno Escarcega |
| Referencia | http://www.gravitar.biz/index.php/datawarehouse/mejores-practicas-para-construir-un-data-warehouse/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Almacenes de Datos |
| Rol | Diseñador de la Base de Datos |
| Título | Usar la Integridad Referencial cuidadosamente. |
| Problema | Utilizar de más a la integridad referencial se producen se hace más lento el sistema. |
| Descripción | Es recomendable no abusar de la integridad referencial. Mientras que las restricciones en llaves foráneas ayudan a la integridad de los datos, tienen asociado un costo en cada inserción, actualización o eliminación. Se puede considerar la ventaja de implementar ciertas validaciones en los ETL's. |
| Beneficio | Obtención de acceso a datos más rápidos. |
| Autor | Bruno Escarcega |
| Referencia | http://www.gravitar.biz/index.php/datawarehouse/mejores-practicas-para-construir-un-data-warehouse/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Bases de Datos Columnares |
| Rol | Diseñador de la Base de Datos |
| Título | Bases de Datos Columnares |
| Problema | Una convencional base de datos relacional almacena los datos por filas, por lo que toda la información de un registro (fila) es inmediatamente accesible. Esto tiene sentido par las consultas transaccionales, que suelen referirse a un registro a la vez. |
| Descripción | Como su nombre lo indica, las bases de datos están organizados por columna en lugar de la fila: es decir, todos los casos de un solo elemento de datos (por ejemplo, Nombre de cliente) se almacenan de modo que se puede acceder como una unidad. |
| Beneficio | Eficacia en las consultas analíticas, como las lista de selecciones, que a menudo lee unos pocos elementos de datos, pero necesitamos ver todas las instancias de estos elementos. |
| Autor | Nelica Valero |
| Referencia | http://www.gravitar.biz/index.php/bi/base-datos-columnar/ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Diseño de la base de datos |
| Rol | Diseñador de la Base de Datos |
| Título | Nombramiento de las columnas de la base de datos |
| Problema | Un DBMS no facilita diseñar correctamente una base de datos. Muchas veces debemos haber pasado por lo mismo. Nos piden diseñar la base de datos para un sistema, entonces procedemos a crear tablas y nombrarlas del modo que se nos ocurra o entendamos, pero cuando empezamos a desarrollar reportes para el sistema, puede ser dificultoso. |

| | |
|---|---|
| Descripción | Una forma que puede ayudarnos mucho y ahorrarnos tiempo es llamar a la columna A de la tabla 1, llamarla también A en la tabla 2.<br>Tipo de columna Sufijo Descripción<br>Nombre del elemento name Usada para describir el nombre de una clave principal<br>Descrip. del elemento desc Usada para describir el nombre con más detalle<br>Fecha de los datos entry_date Usada para marcar con fecha una fila |
| Beneficio | Ahorro de tiempo |
| Referencia | http://ikanus3000.blogspot.com/2008/06/buenas-practicas-diseno-base-datos.html |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Diseño de la base de datos |
| Rol | Diseñador de la Base de Datos |
| Título | Nombramiento de procedimientos almacenados de la base de datos |
| Problema | Un DBMS no facilita diseñar correctamente una base de datos. Muchas veces debemos haber pasado por lo mismo. Nos piden diseñar la base de datos para un sistema, entonces procedemos a crear tablas y nombrarlas del modo que se nos ocurra o entendamos, pero cuando empezamos a desarrollar reportes para el sistema, puede ser dificultoso. |
| Descripción | Una convención que puede utilizarse es ‘usr_<>, de modo que cuando se depure la aplicación se sabe que un objeto con ‘usr_’ es un procedimiento almacenado. Hay que incluir dentro del nombre algo que indique que es lo que hace el procedimiento almacenado. |
| Beneficio | Ahorro de tiempo en la elaboración de la base de datos y en la documentación. |
| Referencia | http://ikanus3000.blogspot.com/2008/06/buenas-practicas-diseno-base-datos.html |
| Ejemplo | Si un procedimiento almacenado actualiza la categoría de un producto, se le puede llamar ‘usr_ActualizarCategoria’ |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Diseño de la base de datos |
| Rol | Diseñador de la Base de Datos |
| Título | Nombramiento de la base de datos |
| Problema | Un DBMS no facilita diseñar correctamente una base de datos. Muchas veces debemos haber pasado por lo mismo. Nos piden diseñar la base de datos para un sistema, entonces procedemos a crear tablas y nombrarlas del modo que se nos ocurra o entendamos, pero cuando empezamos a desarrollar reportes para el sistema, puede ser dificultoso. |
| Descripción | Es similar a la convención de nombre para procedimientos almacenados, como usar el prefijo ‘dev’ y el ‘prod’ para ditnguir entre una base de datos de desarrollo y una de producción |
| Beneficio | Rápida identificación de las bases de datos. |

| | |
|---|---|
| Referencia | http://ikanus3000.blogspot.com/2008/06/buenas-practicas-diseno-base-datos.html |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | XML |
| Rol | Diseñador de la Base de Datos |
| Título | Esquema volátil |
| Problema | Si el esquema de los datos cambia con frecuencia, entonces la representación de los datos en forma relacional tiene como resultado que se incurre en costos y en una sobrecarga por la modificación del esquema relacional. Mientras que algunas formas de modificación al esquema son relativamente sencillas en las bases de datos relacionales, como el agregar una nueva columna a una tabla, otras formas son más complicadas, como el eliminar una columna o el modificar el tipo de una columna. Aún así, hay otras formas de modificaciones de esquemas que son absolutamente difíciles, como la normalización de una tabla en múltiples tablas. La modificación de las tablas significa entonces que las aplicaciones necesitan modificar las instrucciones SQL que tienen acceso a éstas. |
| Descripción | Las partes del esquema que son volátiles pueden expresarse como una única columna XML. Las modificaciones en el formato del documento XML tienen lugar sin necesidad de modificar tablas o columnas en la base de datos y, por lo general, sin desarticular las consultas XML existentes. |
| Beneficio | La naturaleza autodescriptiva y susceptible de ampliarse de XML permite el manejo a la perfección de la variabilidad y la evolución de los esquemas. |
| Referencia | ftp://ftp.software.ibm.com/software/es/db2/purexml-ebook_spanish.pdf |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | XML |
| Rol | Diseñador de la Base de Datos |
| Título | Datos jerárquicos de manera intrínseca por su naturaleza |
| Problema | Los datos que son jerárquicos o recursivos de manera intrínseca con frecuencia son difíciles de representar en esquemas relacionales. Como ejemplos de esto se incluyen listas de materiales, objetos de ingeniería o datos biológicos. La explosión de una lista de materiales puede almacenarse en una base de datos relacional pero el reconstruirla en partes o en su totalidad podría requerir el uso recursivo de SQL. |
| Descripción | Dado que XML es un modelo de datos jerárquico, éste es una representación mucho más natural para los datos de negocios jerárquicos de manera intrínseca. |
| Beneficio | El utilizar XML permite un acceso a los datos simple y de navegación para reemplazar un conjunto de operaciones complejo si éste mismo se representara en un formato tabular. |
| Referencia | ftp://ftp.software.ibm.com/software/es/db2/purexml-ebook_spanish.pdf |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | XML |
| Rol | Diseñador de la Base de Datos |
| Título | Cuando los datos representan objetos de negocios |
| Problema | Si los datos de la aplicación representan objetos de negocios, como formularios de reclamación de un seguro, entonces, con frecuencia es benéfico mantener juntos los elementos de datos que integran una reclamación en particular, en lugar de esparcirlos en un conjunto de tablas. Esto es particularmente cierto cuando los elementos de los datos individuales de un formulario de reclamación no tienen significado comercial válido por sí mismos y sólo pueden interpretarse en el contexto del formulario completo. La normalización de reclamaciones a lo largo de docenas de tablas relacionales significa que la aplicación tiene que lidiar con una fragmentación compleja y poco natural de sus datos de negocios. Esto incrementa la complejidad y la probabilidad de que se presenten errores. |
| Descripción | XML le permite representar incluso objetos de negocios complejos como documentos cohesivos y definidos al mismo tiempo que sigue capturando todas las relaciones entre los elementos de los datos que integran el objeto de negocios. |
| Beneficio | La representación de cada uno de los formularios de reclamación (el objeto de negocios) como un documento XML único en una única fila de una tabla ofrece un modelo de almacenamiento muy intuitivo para el desarrollador de aplicaciones y permite un rápido desarrollo de las aplicaciones. |
| Referencia | ftp://ftp.software.ibm.com/software/es/db2/purexml-ebook_spanish.pdf |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | XML |
| Rol | Diseñador de la Base de Datos |
| Título | Cuando los objetos tienen atributos escasos |
| Problema | Algunas aplicaciones tienen un gran número de posibles atributos, la mayoría de los cuales son escasos, esto es, los atributos son aplicables a muy pocos objetos. Un ejemplo clásico es un catálogo de productos en donde el número de atributos de los diferentes productos es enorme, lo que incluye: tamaño, color, peso, longitud, altura, estilo, tipo de tejido, voltaje, resolución, resistencia al agua y una lista casi interminable de otras propiedades. Para cualquier producto dado, sólo un subconjunto de estos atributos es relevante. Un enfoque relacional posible es almacenar estos datos para tener una columna por atributo, lo que significa que un gran porcentaje de las celdas en la tabla contenga valores NULOS. Esto no es deseable y puede resultar ineficaz. Un enfoque relacional diferente para esos datos escasos es una tabla con tres columnas que almacene varios pares de nombres/valores para cada identificador de producto. Esto significa que los nombres de los |

| | |
|---|---|
| | atributos no son los nombres de las columnas sino valores en una columna del tipo VARCHAR (de longitud variable). Esto evita que los sistemas de bases de datos relacionales estimen con exactitud una selectividad de restricciones y generen planes de consulta eficaces. Asimismo, el definir e imponer restricciones, como la singularidad de un cierto atributo, es extremadamente complejo. |
| Descripción | La belleza de XML es que los elementos y los atributos pueden ser opcionales, de modo que simplemente se omiten si no se aplican a un producto específico. Ni los valores NULOS ni los pares de nombres/valores son necesarios. El esquema XML puede definir un gran número de elementos opcionales; sin embargo, sólo unos cuantos de éstos se utilizan para cualquier objeto dado. Mientras que en una tabla relacional cada fila debe tener el número exacto de columnas, los documentos XML en columnas XML pueden tener diferentes elementos de una fila a la otra. Asimismo, un índice XML para un elemento opcional será muy pequeño si este elemento aparece sólo en un porcentaje pequeño de documentos (filas). |
| Beneficio | Ésta es una clara ventaja con respecto a los índices relacionales que tienen exactamente una entrada por fila. |
| Referencia | ftp://ftp.software.ibm.com/software/es/db2/purexml-ebook_spanish.pdf |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | XML |
| Rol | Diseñador de la Base de Datos |
| Título | Cuando los datos necesitan intercambiarse |
| Problema | Si usted exporta un conjunto de filas de una tabla relacional y lo envía a una aplicación u organización distinta, el destinatario no puede interpretar los datos sin los metadatos adicionales que describan las columnas. Esto es particularmente cierto si su esquema relacional se ha modificado desde la última vez que envió los datos. |
| Descripción | Los datos XML son autodescriptivos. Las etiquetas XML son metadatos que describen los valores que les acompañan. |
| Beneficio | Facilidad para de identificar los datos |
| Referencia | ftp://ftp.software.ibm.com/software/es/db2/purexml-ebook_spanish.pdf |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Web semántica |
| Rol | Diseñador de la Base de Datos |
| Título | Técnicas para la implementación de la web semántica |
| Problema | La carencia de un modelo bien definido de representación de la información en la web ha traído consigo problemas de cara a diversos aspectos relacionados con su procesamiento. |
| Descripción | Los principales medios con los cuales se persiguen los objetivos de la web semántica son: Mediante una codificación de páginas en la cual las etiquetas |

| | |
|---|---|
| | transportes una carga semántica. (Este apartado corresponde al estándar denominado XML). Aportando descripciones (metadatos) de las páginas y sitios web con un formato que sea compatible con la estructura general de la www y con diversas categorías de páginas, e interoperable entre distintos sistemas informáticos. (De esto se ocupa la norma rdf). Además, mediante un sistema de ontologías que permitan especificar conceptos de diverso dominios del conocimiento mediante el uso de un lenguaje fuertemente basado en lógica y susceptible, por tanto, de ser eventualmente "interpretado" por un ordenador. (de este aspecto se ocupa el denominado Owl). |
| Beneficio | XML permite definir tipos de documentos y los conjuntos de etiquetas necesarias para codificarlos. La idea es que, una vez que están marcados o codificados con una colección de etiquetas xml, es posible procesarlos y explotarlos de forma automática con diversos propósitos.<br>Rdf habilita la extracción del significado de la estructura del documento, descrita en xml, con el fin de garantizar la interoperabilidad entre aplicaciones sin necesidad de intervención humana.<br>Owl, con el objeto de que las máquinas puedan realizar tareas de razonamiento útil sobre los recursos de la web semántica, permite compara y combinar documentos (recursos) con distinta estructura. A estos lenguajes o herramientas se les denomina ontologías y básicamente incluyen las definiciones de los conceptos, denominadas "clases", de un dominio y las relaciones entre ellos |
| Autor | Rafael Pedraza, Luis Codina y Cristofol Rovira |
| Referencia | http://www.lluiscodina.com/webSemanticaOntologias2007.pdf |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Bases de Datos Activas |
| Rol | Diseñador de la Base de Datos |
| Título | Bases de Datos Activas |
| Problema | Se desea una evolución de las bases de datos de manera autónoma. |
| Descripción | **Modelo de conocimiento.** Describe la situación y la reacción correspondiente, en definitiva especifica que se puede decir acerca de las reglas de un SGBD activo. Estas reglas se denominan ECA porque constan de Evento, Condición y Acción: cuando ocurre el evento se evalúa la condición y si esta se satisface se ejecuta la acción. Por ejemplo, ante la subida de un valor bursátil (evento), si este incremento es superior a veinte por ciento (condición), emitir órdenes de venta de los títulos (acción).<br>Por lo que respecta al evento se pueden distinguir las siguientes dimensiones:<br>Fuente, que puede ser una operación (p. ej. Insertar una fila en una tabla de la base de datos), un envío de mensajes, una gestión de transacciones (COMMIT), una excepción, el reloj (cada fin de mes), una aplicación (dar al botón del ratón).<br>Granularidad, que puede ser a nivel de registro (por cada registro se define un |

| | |
|---|---|
| | evento) o a nivel de conjunto (se define un evento por un conjunto de registros).<br>Tipo de evento, que puede ser primitivo (insertar una fila) o compuesto (actualizar el sueldo del empleado y, además, subirle de categoría).<br>Papel, que indica si el evento es opcional u obligatorio.<br>En cuanto a la condición, se puede especificar:<br>Papel, si es obligatorio que aparezca la condición o si se contemplan las reglas evento-acción.<br>Ámbito, que indica si en la condición se puede hacer referencia al estado inicial de la base de datos o al estado en que se encuentre cuando se evalúa la condición<br>Para la acción se contemplan las siguientes dimensiones:<br>Opciones, es el tipo de tareas que se puede especificar: operación, envío de mensajes, actualización de reglas, abortar la transacción, hacer en lugar de,..<br>Ámbito, al igual que en el caso de la condición.<br>**Modelo de ejecución**. El modelo se encarga de realizar un seguimiento de la situación, gestionando el comportamiento activo. Se pueden distinguir la siguientes etapas en la ejecución de una regla.<br>Señalización, que trata de la aparición de las ocurrencias del evento.<br>Disparo, que toma los eventos producidos y dispara las reglas correspondientes.<br>Evaluación, de las condiciones de las reglas disparadas.<br>Planificación, indica cómo se procesa el conjunto de las reglas.<br>Ejecución, que lleva a cabo las acciones de las reglas escogidas. |
| Beneficio | Se consigue un nuevo nivel de independencia de datos: la independencia de conocimiento. El conocimiento que provoca una reacción se elimina de los programas de aplicación y se codifica en forma de reglas activas. Además se hace posible el integrar distintos subsistemas (control de accesos, gestión de vistas, etc) y se extiende el ámbito de aplicación de la tecnología de bases de datos a otro tipo de aplicaciones. |
| Referencia | http://www.ganimides.ucm.cl/aurrutia/doc_pdf/El%20Futuro%20de%20las%20Bases%20de%20Datos.pdf |
| Ejemplo | El modelo de conocimiento va asociado a la sintaxis del lenguaje de definición de reglas, a continuación se muestra un ejemplo en una sintaxis genérica:<br>ON UPDATE TO precio OF Producto<br>WHEN NEW.precio  100.000<br>DO UPDATE Descuento<br>SET valor = valor * (NEW.precio/OLD.precio)<br>WHERE Descuento.código = Producto.código;<br>En este caso se define una regla en la que la fuente del evento es la operación de actualización del precio de un producto, la granularidad a nivel de registro y el tipo de evento primitivo. La condición hace referencia al nuevo valor del campo precio y en la acción (que especifica una tarea de actualización del campo valor de la tabla descuento) se referencia tanto al valor antiguo como al nuevo. |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Bases de Datos Deductivas |
| Rol | Diseñador de la Base de Datos |
| Título | Bases de Datos Deductivas |
| Problema | Con el afán de ofrecer una respuesta a las necesidades planteadas por los usuarios y por las aplicaciones avanzadas, en donde se necesitan herramientas semánticamente más ricas que las provistas por la Bases de Datos Relacionales, aparecen aplicaciones de los sistemas de bases de datos que consisten en ofrecer recursos para definir Reglas Deductivas que permitan deducir, inferir u obtener información nueva a partir de los datos almacenados o sucesos condicionados. |
| Descripción | Una base de datos deductiva es un programa lógico; mapeo de relaciones base hacia hechos, y reglas que son usadas para definir nuevas relaciones en términos de las relaciones base y el procesamiento de consultas. |
| Beneficio | Almacenamiento de pocos datos y reglas que permiten crear combinaciones de datos. Una Base de Datos Deductiva utiliza dos tipos de especificaciones: hechos y reglas.<br>Los hechos se especifican de manera similar a como se especifican las relaciones, excepto que no es necesario incluir los nombres de los atributos. En una BDD, el significado del valor del atributo en una tupla queda determinado exclusivamente por su posición dentro de la tupla.<br>Las reglas se parecen un poco a las vistas relacionales. Especifican relaciones virtuales que no están almacenadas realmente, pero que se pueden formar a partir de los hechos aplicando mecanismos de inferencia basados en las especificaciones de las reglas. La principal diferencia entre las reglas y las vistas es que en las primeras puede haber recursividad y por lo tanto pueden producir vistas que no es posible definir en términos de las vistas relacionales estándar.<br>Las BDD buscan derivar nuevos conocimientos a partir de datos existentes proporcionando interrelaciones del mundo real en forma de reglas a partir de datos existentes proporcionando interrelaciones del mundo real en forma de reglas. Utilizan mecanismos internos para la evaluación y la optimización.<br>Características: Una BDD debe contar al menos con las siguientes características:<br>Tener la capacidad de expresar consultas por medio de reglas lógicas.<br>Permitir consultas recursivas y algoritmos eficientes para su evaluación.<br>Contar con negociaciones estratificadas.<br>Soportar objetos y conjuntos complejos.<br>Contar con métodos de optimización que garanticen la traducción de especificaciones dentro de planes eficientes de acceso. |
| Referencia | http://www.quadernsdigitals.net/datos_web/hemeroteca/r_1/nr_502/a_6850/6850.htm |
| Referencia | http://html.rincondelvago.com/bases-de-datos-deductivas.html |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Bases de Datos Federadas |
| Rol | Diseñador de la Base de Datos |
| Título | Bases de Datos Federadas |
| Problema | Sustituir las prácticas tradicionales de procesamiento de datos por bases de datos.<br>La proliferación de bases de datos heterogéneas en las empresas, en las que a veces cada departamento tiene su propio gestor de bases de datos.<br>La compartición de datos de diversas bases de datos en la empresa.<br>La consolidación de recursos software, hardware y de personal.<br>La necesidad de mantener la autonomía de las bases de datos locales. |
| Descripción | Cuando hay varios sistemas de bases de datos en uso, con diferentes modelos y lenguajes que tiene que coexistir, y además, hay que tener en cuenta la dificultad que supone para un usuario acostumbrado a un sistema centralizado, acceder a datos almacenados en distintos sistemas. Existen dos soluciones ampliamente admitidas:<br>Construir un “frontal” (fron-end) sobre los sistemas existentes, que soporta un único modelo de datos y sólo un lenguaje de consulta.<br>Crear una vista temporal con los datos pertinentes a la consulta del usuario. |
| Beneficio | Compartir solo la información que quieran compartir las entidades participante, además de que los usuarios locales podrán acceder de forma transparente los demás datos compartidos y ver los suyos, como si fuera una sola base de datos. |
| Referencia | http://www.ganimides.ucm.cl/aurrutia/doc_pdf/El%20Futuro%20de%20las%20Bases%20de%20Datos.pdf |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Bases de Datos Distribuidas |
| Rol | Diseñador de la Base de Datos |
| Título | Bases de Datos Distribuidas |
| Problema | Maximizar el paralelismo y minimizar el tráfico de la red.<br>Generar algoritmos avanzados de gestión de datos replicados. |
| Descripción | Se pueden clasificar los SGBD de acuerdo a tres dimensiones:<br>Distribución: en la que se considera que los datos pueden estar distribuidos físicamente entre múltiples nodos, o bien almacenados en uno sólo (BD centralizadas).<br>Autonomía: que se refiere al control d ela distribución e indica el grado en el que un SGBD puede operar de forma independiente. Así, se puede hablar de un sistema altamente integrado (en el que los usuarios disponen de una sola imagen de la base de datos) o de un sistema semiautónomo, (en el que los SGBD operan de forma independiente pero que han sido diseñados para participar en una federación), o por último, de un sistema completamente autónomo (multibases de datos). |

| | |
|---|---|
| | Heterogeneidad: a distintos niveles: plataforma (hardware, sistema operativo, protocolos de comunicación), SGBD (modelos y lenguajes), semántica de la base de datos (conflictos a nivel extensional e intencional), etc. |
| Beneficio | Funcionan en arquitecturas cliente/servidor (la mayoría) fue la incorporación de facilidades para crear procedimientos almacenados (que residen en la base de datos) y que permiten disminuir el número de datos y mensajes intercambiados entre cliente y servidor. |
| Referencia | http://www.ganimides.ucm.cl/aurrutia/doc_pdf/El%20Futuro%20de%20las%20Bases%20de%20Datos.pdf |

| | |
|---|---|
| Área | Bases de Datos |
| Subárea | Bases de Datos Móviles |
| Rol | Diseñador de la Base de Datos |
| Título | Bases de Datos Móviles |
| Problema | Asistiendo a cambios tan revolucionarios en las comunicaciones como la expansión de las comunicaciones celulares, LAN (redes de área local) inalámbricas, servicios de satélites, extensión de los ordenadores portables (como los PDA, Personal Digital Assitant, palmtop, laptop, etc.) que ofrecen a los usuarios “móviles” la posibilidad de acceder a la información en cualquier momento y desde cualquier lugar. Se crea este nuevo paradigma. |
| Descripción | Algunos aspectos a tener en cuenta a la hora de diseñar e implementar SGDB móviles, son los siguientes:<br>Desconexión, no hay que olvidar que los terminales móviles están a menudo desconectados y que esta desconexión no se considere un fallo común como los sistemas tradicionales, sino que, en todo caso, se podrían ver como “fallos planificados”<br>Pequeño tamaño y peso de los terminales, que entre otras cosas hace necesario buscar protocolos y algoritmos eficientes en “energía”, debido a las restricciones de baterías que presentan este tipo de equipos. Es imprescindible también llegar a conseguir un equilibrio entre la memoria y disco, por ejemplo, las técnicas de comprensión permiten ahorrar disco pero al descomprimir la información se consume CPU y, por lo tanto, energía. |
| Beneficio | Acceder a la información en cualquier momento y desde cualquier lugar. |
| Referencia | http://www.ganimides.ucm.cl/aurrutia/doc_pdf/El%20Futuro%20de%20las%20Bases%20de%20Datos.pdf |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Requerimientos, Análisis, Diseño, Construcción, Pruebas. |
| Rol | Analista |
| Título | Registro de Rastreo |
| Problema | ¿Dónde está implementado el requerimiento?<br>¿Cuáles decisiones de diseño afectan la implementación del requerimiento? |

| | |
|---|---|
| | ¿Todos los requerimientos están localizados en todas las fases del proceso de desarrollo de software?<br>¿La implementación se ajusta con los requerimientos?<br>¿Qué pruebas serán utilizadas para verificar un requerimiento?<br>¿En qué parte del código se encuentra implementado cada requerimiento?<br>¿Cuál es el impacto de los cambios en los requerimientos? |
| Descripción | De acuerdo a MoProsoft, es la relación entre los requerimientos, elementos análisis y diseño, componentes y pruebas |
| Beneficio | Permite la trazabilidad de los requerimientos a lo largo de las fases del ciclo de vida de desarrollo de software.<br>Facilita la administración de requerimientos, sobre todo cuando los requerimientos del usuario/cliente cambian.<br>Permite seguir una traza o pista hacia delante y hacia atrás de los requerimientos.<br>Facilita las tareas de mantenimiento del software.<br>Permite al Administrador del Proyecto determinar los costos (humanos, financieros y materiales) de la modificación a los requerimientos del usuario/cliente. |
| Ejemplo | Se aplicó en Proyectos de desarrollo de Software |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Requerimientos, Análisis, Diseño, Construcción, Pruebas. |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Reporte de Actividades |
| Problema | Estimar tiempo a las actividades basadas en la experiencia.<br>Seguimiento en tiempo a las actividades y los productos de trabajo. |
| Descripción | El reporte de actividades contempla los datos necesarios para el registro de las actividades cotidianas seccionados por semana y mes, proporciona de forma automática las actividades a seleccionar de acuerdo al proceso elegido, así como la administración de dichas actividades, y la generación de un reporte por proyecto y/o mes |
| Beneficio | Permite: Dar seguimiento puntual a los Planes, estimar de forma más precisa la planeación de las actividades, revisar en que se consume más tiempo al realizar actividad (generación, revisión o corrección), descubrir las actividades que no se contemplan en la planeación y en los procesos. |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Administración de la Configuración |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Base de Conocimiento |
| Problema | Diseminación, pérdida, indisponibilidad y falta de control de la información generada en la organización. |
| Descripción | La información generada en la organización es uno de los activos más |

| | |
|---|---|
| | importantes. El contar con un repositorio centralizado que permita concentrar y administrar la información, facilita la operación del día y día, creando valor en el corto, mediano y largo plazo. La integración de la Base de Conocimiento debe considerar los medios en los que se almacena la información ya sea de manera impresa o en forma electrónica. También crear una estructura estándar que apoye a la identificación y ubicación de la información, los mecanismos de operación que garanticen la confiabilidad, integridad y disponibilidad de la información, de acuerdo con las necesidades y características de la organización. |
| Beneficio | Disponibilidad inmediata de la información.<br>Facilita el control de versiones y el trabajo en grupo.<br>Reutilización y Productividad. |
| Autor | Irene Sánchez García |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Diseño e Implementación |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Layers |
| Problema | Supongamos un sistema cuya característica dominante es una mezcla de cuestiones de alto y bajo nivel. |
| Descripción | Es un patrón de arquitectura que ayuda a estructurar aplicaciones que pueden descomponerse en grupos de subáreas de forma que las tareas de cada grupo se encuentran en el mismo nivel de abstracción. |
| Beneficio | Posteriores cambios en el código no deberían propagarse a través de todo el sistema. Las partes del sistema deberían ser intercambiables. Las interfaces deberían ser estables. |
| Referencia | Design Patterns: Elements of Reusable Object-Oriented Software", de Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, 1995, Addison-Wesley, también conocido como el libro GOF |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Requerimientos, Análisis, diseño, implementación, construcción, integración y pruebas. |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Dividir un ciclo en fases con productos intermedios |
| Problema | Aparecen requerimientos en cualquier etapa en el desarrollo del sistema |
| Descripción | Realizar entregables para incorporar nuevos requerimientos |
| Beneficio | El equipo y el cliente: logran visibilidad del avance a través de productos intermedios, logran consistencia con requisitos a través de revisiones intermedias, pueden tomar decisiones si continuar o no, tienen bases para los siguientes ciclos. |
| Autor | Hanna Oktaba |
| Referencia | http://www.redisybd.unam.mx/ |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Estimación de productos |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Estimación exploratoria: Bottom-up para la estimación en el proceso de desarrollo de proyectos de software a la medida. |
| Problema | Cuando no se tenga información histórica de estimaciones previas |
| Descripción | Utilice un WBS estándar de la compañía para crear el WBS del proyecto. En caso de no tenerlo, se debe construir uno de acuerdo con el nivel de detalle que se tenga con los requerimientos del proyecto. Si el WBS tiene tareas muy detalladas que no se puedan estimar en esta etapa, se deben realizar agrupaciones de éstas. El resultado de este agrupamiento es que la estimación posiblemente será aún menos exacta. |
| Beneficio | Estimación eficiente, permite que el proceso de aprobación y/o comercialización del proyecto sea lo más ágil posible. |
| Referencia | http://es.calameo.com/read/0000018161cd6f6345c16 |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Estimación de productos |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Estimación de Presupuesto: Puntos de Casos de Uso |
| Problema | Cuando no se cuenta con un método en el modelado del sistema con casos de uso. |
| Descripción | Cuando el modelado del sistema se realiza a través de casos de uso. Esta técnica se puede ejecutar rápidamente si dentro del proceso de análisis, el analista es acompañado por el especialista en estimación y una vez se esté especificando el caso de uso se establezcan los criterios de peso del caso de uso. |
| Beneficio | Haciendo uso de una buena herramienta puede generar una estimación prácticamente al instante, disminuyendo los tiempos de estimación y aumentando la productividad. |
| Referencia | http://es.calameo.com/read/0000018161cd6f6345c16 |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Pruebas, Mantenimiento y Seguridad |
| Rol | Responsable de Pruebas |
| Título | Reproducibilidad. Evaluaciones Dinámicas |
| Problema | Las evaluaciones dinámicas son difíciles de reproducir, lo que significa que la misma evaluación ejecutada en distintos momentos puede arrojar resultados diferentes. Esta variabilidad puede estar causada por cambios en los productos de los vendedores (por ejemplo, por la actualización de las firmas de virus) y también por alteraciones en el entorno de los códigos maliciosos. Algunos códigos maliciosos sólo funcionarán bien si están disponibles ciertos |

| | |
|---|---|
| | recursos externos (como servidores NTP, páginas de sitios Web bancarios, sitios liberadores de programas maliciosos). Aunque es posible imitar algunos de estos factores en un entorno de evaluación, es difícil reproducirlos con una exactitud total, y el entorno de evaluación puede resultar cada vez más artificial. Esta variabilidad significa que es difícil llegar a conclusiones certeras habiendo realizado una única evaluación. |
| Descripción | El evaluador cuenta con dos defensas contra esta variabilidad. La primera es reunir suficientes registros e información para verificar lo que ocurrió en la evaluación. Por ejemplo, para mostrar que el código malicioso “x” evaluado el día “y” sobre el producto “z” realmente llegó a actuar. La segunda es usar una cantidad suficiente de muestras y repetir las evaluaciones con el paso del tiempo (usando distintos grupos de muestras). |
| Beneficio | Que inconsistencias en el comportamiento de los códigos maliciosos tengan menos probabilidad de desviar los resultados. |
| Referencia | “Best Practices for Dynamic Testing” (version 2008-10-31) |
| Ejemplo | Por ejemplo, los códigos maliciosos recientes pueden probarse durante un mes desde el día en que se obtuvieron, comparando el rendimiento o las tasas de detección diarios. |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Pruebas, Mantenimiento y Seguridad |
| Rol | Responsable de Pruebas |
| Título | Selección del producto. Evaluaciones Dinámicas |
| Problema | A veces la protección antimalware ya está incorporada en paquetes de productos, mientras que otras veces se encuentran como productos independientes. |
| Descripción | El evaluador debe darse cuenta de estas diferencias y elegir los productos con detenimiento para realizar evaluaciones comparativas razonablemente apropiadas. Una forma de descubrir estas diferencias sería usar los anuncios de vendedores como base para elegir productos. |
| Beneficio | Tener protección contra diferentes intrusos. |
| Referencia | “Best Practices for Dynamic Testing” (version 2008-10-31) |
| Ejemplo | Si los vendedores aseguran que sus productos tratan la amenaza “x”, entonces sería razonable evaluar varios productos que traten la amenaza “x”. |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Diseño, implementación, construcción, integración y pruebas. |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Manejo de versiones |
| Problema | Por alguna razón el editor se cierra. Un gran puñetazo en el escritorio: perdimos los cambios. O tal vez fue un gran cambio que comenzamos a hacer y no funcionó. Todos hemos pasado uno que otro mal momento por no saber organizar nuestro trabajo. Muchísimos problemas surgen debido a espaguetis de código mal unidos, versiones que se creían diferentes, trabajo que se pierde. |
| Descripción | SIEMPRE hay que usar algún manejador de versiones. Ya que proporcionan: |

| | |
|---|---|
| | Backup. Son una manera muy sencilla para hacer copias de respaldo. Aunque sea una diaria.<br>Cambios. Cuando quiero hacer un cambio peligroso, me quita el miedo de pasarme llevando algo por delante. Si eso pasara, simplemente vuelvo a una versión anterior.<br>Manejo de versiones en sí. Un cliente quiere unas cosas, otro cliente quiere otras, pero la base es la misma. Puedo mantener las fuentes exactas de cada instalación, añadir nuevas cosas y luego unirlas.<br>Trabajo en equipo. El manejo de una manejador de versiones facilita enormemente el trabajo en equipo. Al final del día siempre hay una versión, igual para todos, y los cambios de cada uno los ve el otro.<br>Es sencillo. Cualquier IDE te permite importar tus fuentes a un repositorio y trabajar a partir de él sin mucho trabajo, incluso si estás comenzando a usar un manejador. |
| Beneficio | Contar con una sola versión para todos los integrantes del equipo. Hacer la vida más sencilla. |
| Autor | Luis H. Fernández |
| Referencia | Software Guisho, Obtenida el 22 de enero de 2010, de http://software.guisho.com/buenas-practicas-manejo-de-versiones |
| Ejemplo | Utilizar SVN o CVS, son sencillos, son libres, la mayoría de IDEs tiene extensiones para ellos. |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Ingeniería de Requisitos, desarrollo de software orientada por aspectos |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Separación multidimensional de intereses (MDSOC) |
| Problema | Los problemas que surgen a lo largo del proceso de desarrollo se deben a la carencia de un proceso adecuado de definición y entendimiento del problema y a la definición poco clara de las necesidades del cliente. |
| Descripción | Este enfoque es pionero en el tratamiento de los aspectos en la fase de requisitos. Se basa en el enfoque de Puntos de Vista y su fortaleza está centrada en el tratamiento de las reglas de composición de los intereses, así como en la definición y manejo de conflictos. La orientación multidimensional permite analizar el espacio de intereses del problema de forma homogénea. Para lograr esto define un interés como la agrupación de requisitos funcionales y no funcionales de cada punto de vista del sistema. |
| Beneficio | Facilitar la identificación, separación y clasificación de intereses y la composición de intereses transversales |
| Autor | Luis Fernando Londoño, Raquel Anaya y Marta Silvia Tabares. |
| Referencia | Revista EIA, ISSN 1749-1237, Número 9 p. 43-52, Julio 2008. Artículo “Análisis de la ingeniería de requisitos orientada por aspectos según la industria del software.” |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Ingeniería de Requisitos, desarrollo de software orientada por aspectos |

| | |
|---|---|
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | AOSD con casos de uso (AOSD/US) |
| Problema | Los problemas que surgen a lo largo del proceso de desarrollo se deben a la carencia de un proceso adecuado de definición y entendimiento del problema y a la definición poco clara de las necesidades del cliente. |
| Descripción | Este enfoque trata los requisitos funcionales desde los casos de uso que representan la función básica del sistema *(peer use cases).* Los requisitos no funcionales se representan en casos de uso que extienden un *caso de uso de infraestructura,* el cual se parametriza al igual que el actor que lo usa (los parámetros representan el comportamiento que se cruzará en la funcionalidad de los casos de uso base). En el análisis y el diseño los casos de uso se representan en una estructura de composición que se identifica con el estereotipo <*use case slice*> y agrupa elementos de modelo que colaboran para lograr los requisitos del sistema (funcionales y no funcionales) |
| Beneficio | Facilitar la identificación, separación y clasificación de intereses y la composición de intereses transversales |
| Autor | Luis Fernando Londoño, Raquel Anaya y Marta Silvia Tabares. |
| Referencia | Revista EIA, ISSN 1749-1237, Número 9 p. 43-52, Julio 2008. Artículo "Análisis de la ingeniería de requisitos orientada por aspectos según la industria del software." |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Ingeniería del Software basada en Componentes |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Utilizar Arquitecturas basadas en componentes |
| Problema | No se comprenden las necesidades del usuario, no se previó el impacto de los requerimientos de cambios, no se hace analizan los riesgos. |
| Descripción | La Arquitectura de Software representa el conjunto de decisiones significativas sobre la organización de un sistema de software: selección de los elementos estructurales, y las interfaces, por los cuales el sistema está compuesto; comportamiento, especificado como colaboraciones entre los elementos; composición en subsistemas de los elementos estructurales y de comportamiento, estilo de arquitectura que guíe a la organización.<br>Un componente de software puede definirse como una pieza no trivial de software, un modulo o un subsistema que contempla una función clara, tiene límites claros y puede ser integrado en una arquitectura bien definida. |
| Beneficio | Desarrollar componentes para ser reutilizados, forma la base de rehúso de la organización |
| Autor | Hanz Cocchi Guerrero |
| Referencia | http://www.slideshare.net/hanzcg/conferencia-gestin-de-proyectos-de-ti |
| Ejemplo | Componente A Aplicación<br>Componente n1 componente n2 Negocio<br>Componente m1 componente m2 Middleware<br>Componente s1 componente s2 System-software |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Seguridad en Ingeniería del Software |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Seguridad en Ingeniería del Software |
| Problema | Los sistemas al ser más complejos proporcionan más oportunidades para los ataques. |
| Descripción | Primero debe determinar los riesgos de una aplicación particular. Una vez que se identifiquen los riesgos, identificar medidas de seguridad apropiadas llega a ser manejable. En particular, al definir los requisitos, es importante considerar cómo la aplicación será utilizada. |
| Beneficio | Con ese conocimiento uno puede decidir, si o no, utilizar características complejas como contabilidad, auditoría, etc. |
| Autor | Nick Feamster |
| Referencia | http://www.acm.org/crossroads/espanol/xrds7-4/onpatrol74.html |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Procesos de Negocio e Ingeniería de Servicios |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | BPMN(Business Process Modeling Notation) |
| Problema | No existe una notación que sea fácilmente entendible por todos los usuarios de negocio. |
| Descripción | BPMN crea un “puente” estandarizado para suplir la brecha entre los procesos de negocio y la implementación de procesos. Se usa para comunicar una amplia variedad de información a diferentes audiencias. Cubre varios tipos de modelado y permite la creación tanto de segmentos de proceso como procesos de negocio de comienzo a fin, y en diferentes niveles de representatividad. Es independiente de cualquier metodología de modelado de procesos. |
| Beneficio | Se intenta reducir la fragmentación de otras notaciones y herramientas de modelado anteriores. El desarrollo de procesos de negocio por la gente de negocio ha sido técnicamente separado de la representación de procesos requeridos para el diseño de sistemas que implementan y ejecutan estos procesos. Por lo tanto, ha sido necesario traducir manualmente los modelos originales de procesos de negocio a modelos de ejecución. Tales traducciones están sujetas a errores y hacen difícil a los propietarios de los procesos entender la evolución y funcionamiento de los procesos que ellos desarrollan. |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Ingeniería web |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Nacimiento de la ingeniería web |

| | |
|---|---|
| Problema | Desarrollo de aplicaciones de una forma “tradicional” o mejor dicho “anticuada” cuando las tecnologías y tendencias han cambiado. |
| Descripción | Las aplicaciones deben estar libres de huecos de seguridad y por lo tanto es recomendable que también se incluya en el equipo de desarrollo a profesionales del área de seguridad web. Aunque en el desarrollo de aplicaciones web sea tan fácil copiar y arrastrar contenido de otras fuentes (texto, imágenes, videos, audio), los equipos de desarrollo deben conocer las implicaciones legales que esto conlleva, la mayoría de información en la red aunque es pública para su difusión tiene diferentes tipos de licenciamiento que prohíben este tipo de uso.<br>Esta nueva generación de aplicaciones que tienen a disposición todo el conocimiento de internet deben hacer uso de él, por estos los desarrolladores deben conocer su entorno y saber interactuar con él de la mejor forma, pues cada vez tienden a ser más las aplicaciones y sitios que ofrecen mecanismos para interactuar con su propia información. |
| Beneficio | Poder consultar el conocimiento que se encuentra en internet |
| Referencia | http://www.sisfo.com/blog/category/buenas-practicas/ |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Ingeniería web |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Ingeniería web, entendimiento de las necesidades del negocio |
| Problema | Hacer un trabajo con el que el usuario final no está de acuerdo. |
| Descripción | Tomar tiempo para entender las necesidades del negocio y los objetivos de l producto, incluso si los detalles de la WebApp son vagos. Si no se tienen en cuenta estos detalles, el resultado final es un buen trabajo técnico que conduce a la construcción de un sistema equivocado por razones equivocadas para el público equivocado. |
| Beneficio | Un buen trabajo técnico que conduce a la construcción de un sistema afortunado para el público idóneo. |
| Referencia | http://commons.wikimedia.org/wiki/User:MmgULE |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Ingeniería web |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Interacción de los usuarios con la WebApp (Aplicación web) |
| Problema | Utilizar métodos equivocados para el desarrollo |
| Descripción | Describir cómo interactuarán los usuarios con la WebApp aplicando un enfoque basado en escenarios. Se debe convencer a los accionistas para desarrollar casos de uso para reflejar cómo los diversos actores interactuarán con la WebApp |
| Beneficio | Se pueden aprovechar dichos escenarios: para la planeación y rastreo del proyecto, para guiar al análisis y el modelado del diseño y como una entrada |

| | |
|---|---|
| | para el diseño de pruebas |
| Referencia | http://commons.wikimedia.org/wiki/User:MmgULE |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Ingeniería web |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Plan de proyecto |
| Problema | No contar procedimiento para el desarrollo de la aplicación |
| Descripción | Desarrollar un plan del proyecto, incluso si es muy breve. La dosificación del proyectos debe ser exacta, por los cortos tiempos de plazo, muchas veces el proyecto debe planearse y rastrearse diariamente |
| Beneficio | Saber cuando y quién realizará cierta actividad. |
| Referencia | http://commons.wikimedia.org/wiki/User:MmgULE |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Ingeniería web |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Revisar la consistencia y calidad de los modelos |
| Problema | ¿Cómo descubrir errores en cualquier representación del software? |
| Descripción | Revisar la consistencia y calidad de los modelos. Las revisiones técnicas formales se deben dirigir a lo largo del proyecto IWeb. Las revisiones técnicas formales son una actividad de control de calidad del SW que debe llevar a cabo el equipo de ingeniería Web |
| Beneficio | Las revisiones técnicas sirven para: descubrir errores en la función, lógica o implementación en cualquier representación del software; verificar que el SW en revisión satisface sus requisitos; garantizar que el software se ha representado de acuerdo con los estándares predefinidos; lograr SW desarrollado de una manera uniforme y hacer proyectos más manejables. |
| Referencia | http://commons.wikimedia.org/wiki/User:MmgULE |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Ingeniería web |
| Rol | Responsable de Desarrollo y Mantenimiento de Software |
| Título | Aplicaciones probadas |
| Problema | Aplicaciones regresadas por tener errores. |
| Descripción | No apoyarse en usuarios anteriores para depurar la WebApp; diséñense pruebas amplias y ejecútense antes de liberar el sistema. Los usuarios de una WebApp con frecuencia le dan la oportunidad. Si falla en su ejecución se van a cualquier otra parte y nunca regresan. “PRUEBE PRIMER, DESPUÉS DESPLIEGUE” debe ser un sistema primordial, incluso si los plazos se deben prolongar. |
| Beneficio | Aplicaciones bien probadas y ejecutadas |

| | |
|---|---|
| Referencia | http://commons.wikimedia.org/wiki/User:MmgULE |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Diseño de páginas web |
| Rol | Diseñador de Interfaz de Usuario |
| Título | No abandone las reglas del buen diseño que ya conoce |
| Problema | Realizar una página sin reglas, no hay entendimiento de la misma. |
| Descripción | Siga las reglas básicas de un buen diseño (diseño centrado en el usuario, escribir concisamente evitando el lenguaje comercial superfluo, ofertar menos opciones, incluyendo sólo las más importantes no incluir gráficos y sonidos sólo por contar con ellos) |
| Beneficio | El usuario podrá navegar por la página sin tanto problema. |
| Referencia | http://www.rodolfoquispe.org/blog/75-buenas-practicas-de-diseno-de-paginas-web-y-intranets-basado-en-estudios-de-usabilidad.php |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Diseño de páginas web |
| Rol | Diseñador de Interfaz de Usuario |
| Título | Gráficos y multimedia |
| Problema | Dificultad para entender la página |
| Descripción | Dar a todos los gráficos nombres que sean comprensibles y que transmitan de verdad lo que el gráfico es y hace.<br>Nunca difuminar las imágenes para indicar no disponibilidad.<br>Cuando los gráficos contengan información útil, también proporcionar la información en texto.<br>Dé a los usuarios maneras y alternativas de obtener la información contenida en cualquiera de los gráficos que se encuentren.<br>No utilice una imagen en miniatura de la página de su sitio para utilizarla como gráfico en otra página.<br>Al hacer uso de gráficos, elija siempre imágenes de claras y nítidas.<br>Facilite a los usuarios la posibilidad de saltarse cualquier elemento multimedia, aplicación Java o Flash.<br>No cree automáticamente una versión sólo-texto de su sitio. |
| Beneficio | Usabilidad en la navegación de la página |
| Referencia | http://www.rodolfoquispe.org/blog/75-buenas-practicas-de-diseno-de-paginas-web-y-intranets-basado-en-estudios-de-usabilidad.php |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Diseño de páginas web |
| Rol | Diseñador de Interfaz de Usuario |
| Título | Enlaces y botones |
| Problema | Dificultad para entender la página |

| | |
|---|---|
| Descripción | Limite el número de enlaces en una página.<br>Evite pequeños botones y enlaces con texto en minúscula.<br>Deje espacio entre los enlaces y botones. Subraye todos los enlaces |
| Beneficio | Usabilidad en la navegación de la página |
| Referencia | http://www.rodolfoquispe.org/blog/75-buenas-practicas-de-diseno-de-paginas-web-y-intranets-basado-en-estudios-de-usabilidad.php |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Diseño de páginas web |
| Rol | Diseñador de Interfaz de Usuario |
| Título | Textos |
| Problema | Dificultad para entender la página |
| Descripción | Seleccione colores de texto con buen contraste.<br>No use un tamaño de fuente muy pequeño para el texto de la página.<br>No se base en una imagen de fondo para crear el contraste con el texto.<br>Pruebe los colores y fuente de su sitio con un magnificador de pantalla.<br>Asegúrese de que es posible magnificar su sitio.<br>Escriba de forma concisa y elimine el texto superfluo.<br>Replantéese la forma en la que usa los paréntesis y los asteriscos. |
| Beneficio | Usabilidad en la navegación de la página |
| Referencia | http://www.rodolfoquispe.org/blog/75-buenas-practicas-de-diseno-de-paginas-web-y-intranets-basado-en-estudios-de-usabilidad.php |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Diseño de páginas web |
| Rol | Diseñador de Interfaz de Usuario |
| Título | Páginas |
| Problema | Dificultad para entender la página |
| Descripción | Evite las “páginas portada” previas a la página de inicio de un sitio, haga que la primera página que la gente vea sea la página que mejor describe la compañía y el sitio. Incluya tan sólo los pasos y las páginas necesarias. |
| Beneficio | Usabilidad en la navegación de la página |
| Referencia | http://www.rodolfoquispe.org/blog/75-buenas-practicas-de-diseno-de-paginas-web-y-intranets-basado-en-estudios-de-usabilidad.php |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Diseño de páginas web |
| Rol | Diseñador de Interfaz de Usuario |
| Título | Campos y formularios |
| Problema | Dificultad para entender la página |
| Descripción | Limite la cantidad de información que el formulario requiere; recoja sólo el mínimo necesario. |

| | |
|---|---|
| | Ponga las etiquetas de texto de los campos muy cerca de los campos que les corresponden.<br>No confíe sólo en el asterisco para indicar que un campo es necesario.<br>Asegúrese de que el orden de tabulación es lógico.<br>Apile los campos en una columna vertical.<br>Ofrezca campos de entrada estándar para los números de teléfono. |
| Beneficio | Usabilidad en la navegación de la página |
| Referencia | http://www.rodolfoquispe.org/blog/75-buenas-practicas-de-diseno-de-paginas-web-y-intranets-basado-en-estudios-de-usabilidad.php |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Diseño de páginas web |
| Rol | Diseñador de Interfaz de Usuario |
| Título | Búsquedas |
| Problema | Dificultad para entender la página |
| Descripción | Ofrezca un motor de búsqueda que perdone los errores de ortografía.<br>No base únicamente la capacidad de búsqueda de un sitio en la interfaz de navegación.<br>Coloque la caja de búsqueda donde los usuarios la esperan encontrar, no en una zona inesperada. Describa claramente los resultados de la búsqueda.<br>Avise a los usuarios cuando no han introducido nada en la caja de búsqueda.<br>No presente el ranking de relevancia de los resultados de la búsqueda en una tabla. |
| Beneficio | Usabilidad en la navegación de la página |
| Referencia | http://www.rodolfoquispe.org/blog/75-buenas-practicas-de-diseno-de-paginas-web-y-intranets-basado-en-estudios-de-usabilidad.php |

| | |
|---|---|
| Área | Ingeniería de Software |
| Subárea | Diseño de páginas web |
| Rol | Diseñador de Interfaz de Usuario |
| Título | Comercio electrónico |
| Problema | Dificultad para entender la página |
| Descripción | Describa minuciosamente las imágenes de los productos que el sitio vende como si no hubiera imágenes.<br>Ayude a los usuarios a seguir después de haber añadido algo al carrito, dándoles una forma de llegar de nuevo a donde estaban.<br>Coloque los botones de “añadir a la cesta de compra” o “realizar pedido” cerca de los elementos a comprar.<br>Tenga en cuenta a los clientes internacionales seleccionando cuidadosamente los términos que utiliza. |
| Beneficio | Usabilidad en la navegación de la página |
| Referencia | http://www.rodolfoquispe.org/blog/75-buenas-practicas-de-diseno-de-paginas-web-y-intranets-basado-en-estudios-de-usabilidad.php |

# Anexo B

## B.1 Casos de Uso, Prototipo y Plan de Pruebas

### B.1.1 Ingeniero de Software

**Caso de uso 1:** Entrar

**Actor:** Ingeniero de Software


**Descripción:** El ingeniero de software entra al sistema

**Precondición:** El ingeniero de software debe teclear la página del sistema

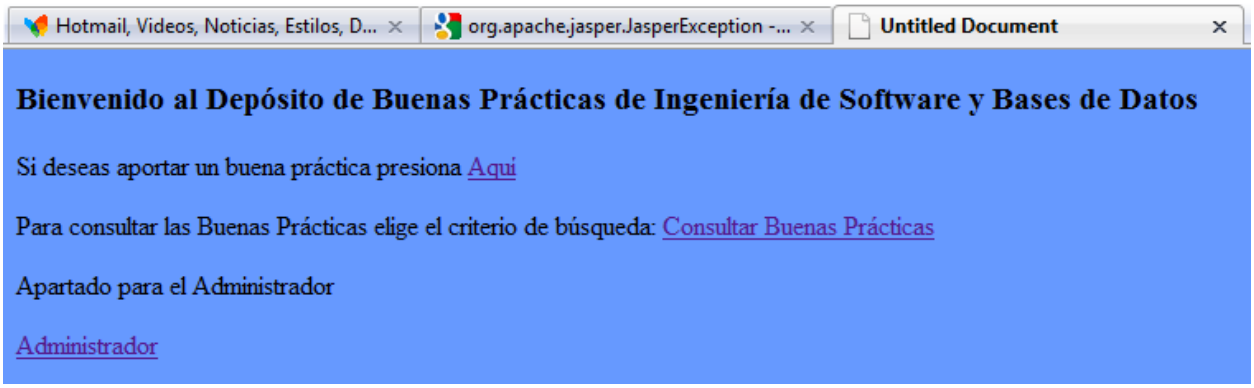
**Flujo:**

| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Entra a la página del sistema | 2 | Despliega la interfaz principal del sistema | - |

**Poscondiciones:**

- El ingeniero de software está dentro del sistema
- El sistema despliega la interfaz principal

**Prototipo**



**Caso de uso 8:** Salir

**Actor:** Ingeniero de Software


**Descripción:** El ingeniero de software sale del sistema

**Precondiciones:**
- El ingeniero de software está dentro del sistema
- El ingeniero de software desea salir del sistema

**Flujo:**

| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Cierra el navegador o teclea una nueva dirección URL. | 2 | Cierra la conexión con la máquina del usuario. | - |

**Poscondiciones:**
- El ingeniero de software salió del sistema
- El sistema ya no está en funcionamiento en esa máquina

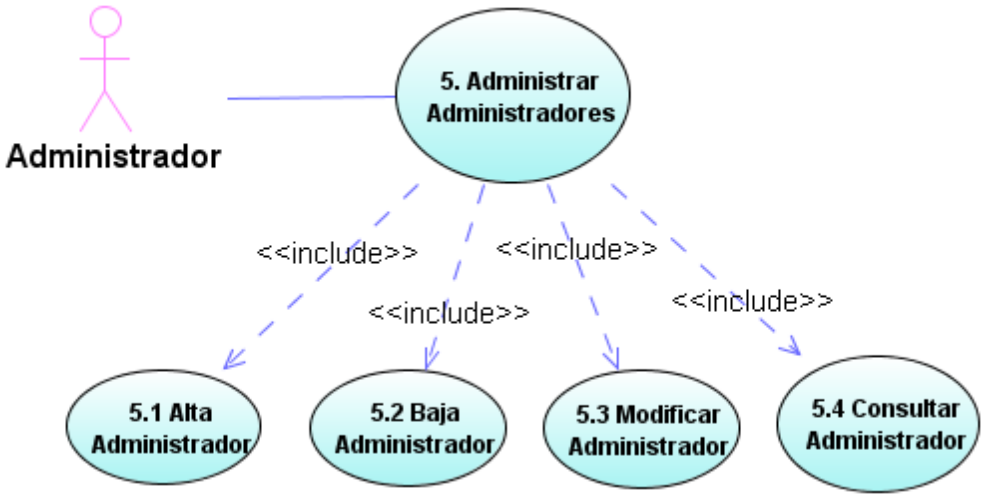
**Plan de Prueba**

| Caso de Uso | Entradas | Resultados Esperados |
|---|---|---|
| 8 | Se cierra el navegador. | El sistema cierra la sesión. |
| 8 | Se presiona cerrar sesión. | El sistema cierra la sesión |

### B.1.2 Administrador

**Caso de uso 5:** Administrar Administradores

**Actor:** Administrador




**Descripción:** El administrador desea dar de alta, baja, modificar o consultar a un administrador.

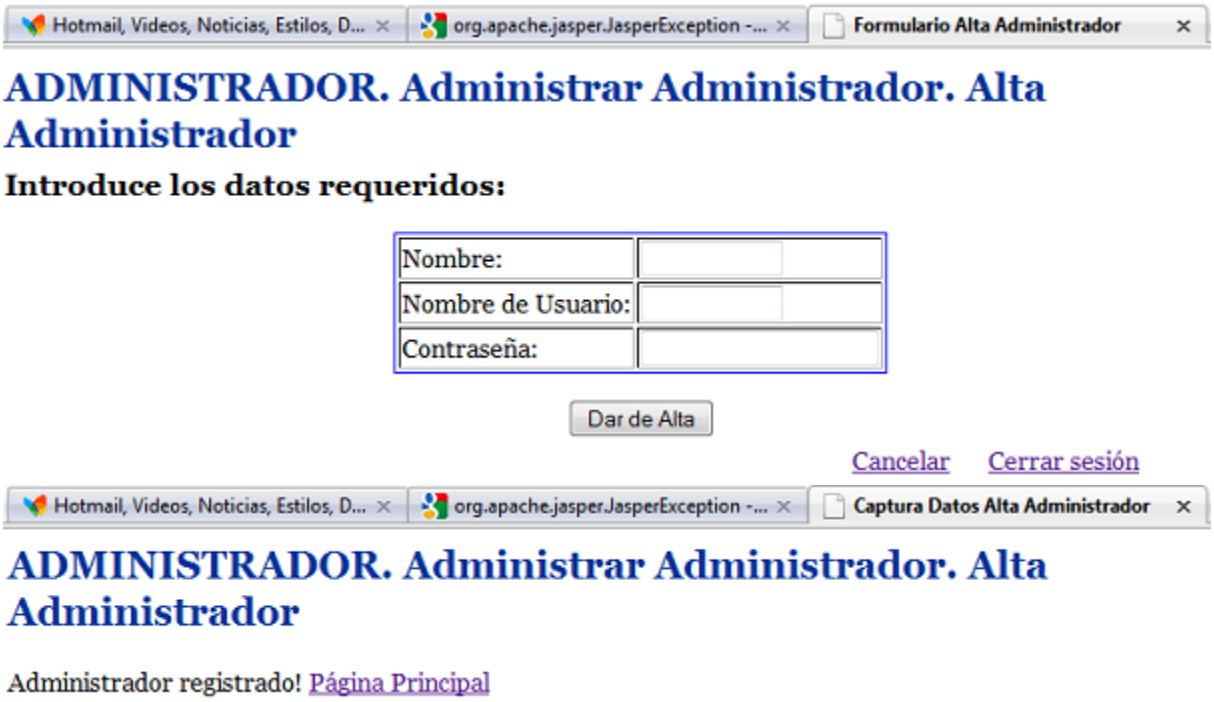
**Precondición:** El administrador se ha autentificado y desea dar de alta, baja, modificar o consultar a un administrador.

**Flujo:**

5.1 Alta Administrador

| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Elige la opción "Alta Administrador" del menú principal del Administrador | 2 | Muestra el formulario para dar de alta al administrador. | - |
| 3 | Introduce los datos requeridos en el formulario y presiona el botón "Dar de Alta". | 4 | Envía los datos a la base de datos | E1 |
| - | - | 5 | Muestra la pantalla de alta realizada. | E2 |
| 6 | Oprime el botón "Página Principal" | 7 | Muestra el menú principal del Administrador | - |

**Prototipo**

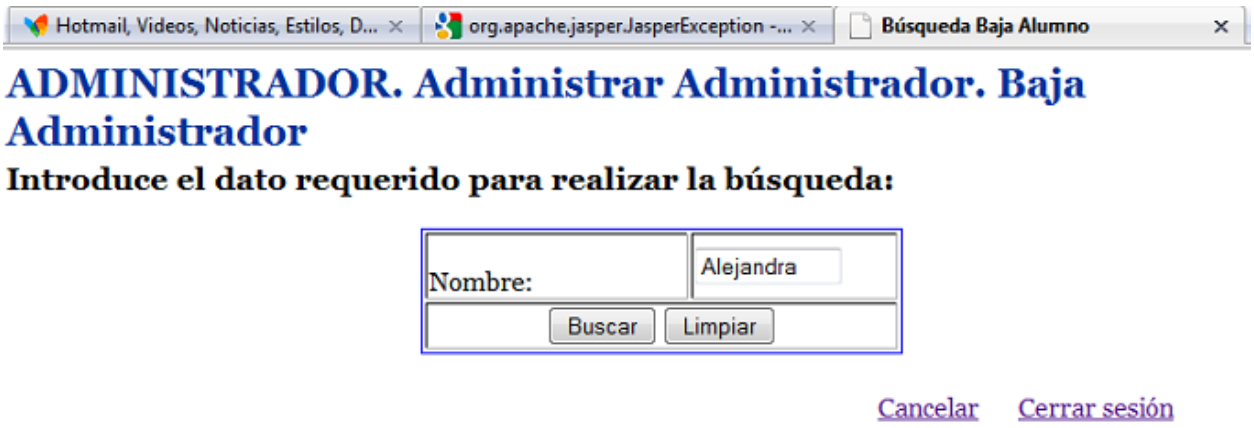



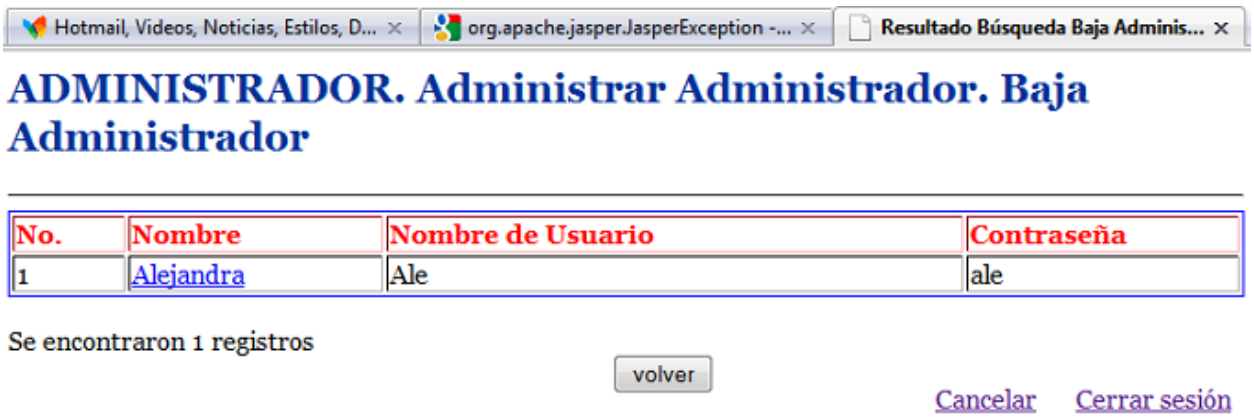

5.2 Baja Administrador

| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Elige la opción "Baja Administrador" del menú principal del Administrador | 2 | Muestra el formulario para la búsqueda del administrador a dar de baja | - |
| 3 | Elige el criterio de búsqueda, introduce los datos requeridos en el | 4 | Realiza la búsqueda en la base de datos | E1, E2 |

| | | | | |
|---|---|---|---|---|
| | formulario y presiona el botón “Buscar” | | | |
| - | - | 5 | Muestra el listado con los resultados de la búsqueda[3] | - |
| 6 | Selecciona el registro del administrador que desee dar de baja y presiona el botón “Dar de Baja”. | 7 | Muestra la pantalla de cuestionamiento de confirmación de baja [4] | E2 |
| 8 | Presiona el botón “Confirmar Baja” | 9 | Realiza la baja en la base de datos. | E2 |
| - | - | 10 | Muestra la pantalla de baja realizada | - |
| 11 | Presiona el botón “Volver” | 12 | Muestra el menú principal del Administrador | - |

**Prototipo**



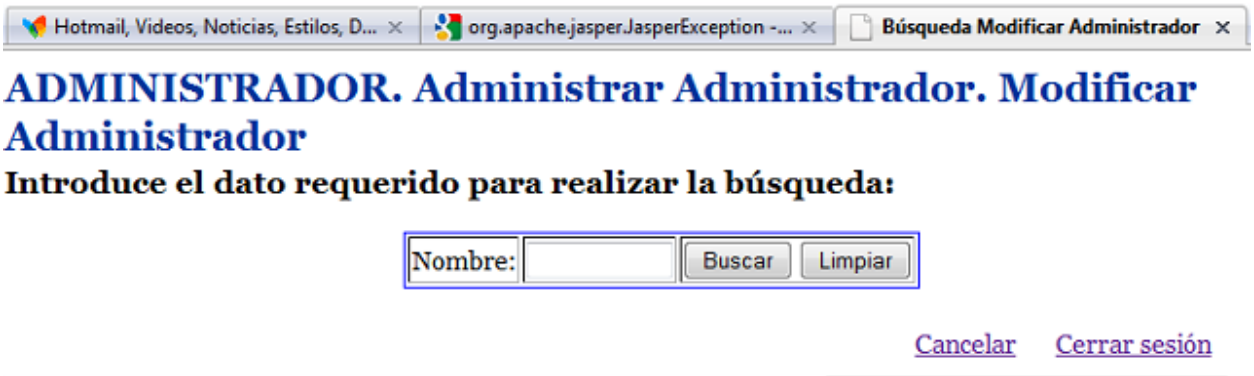


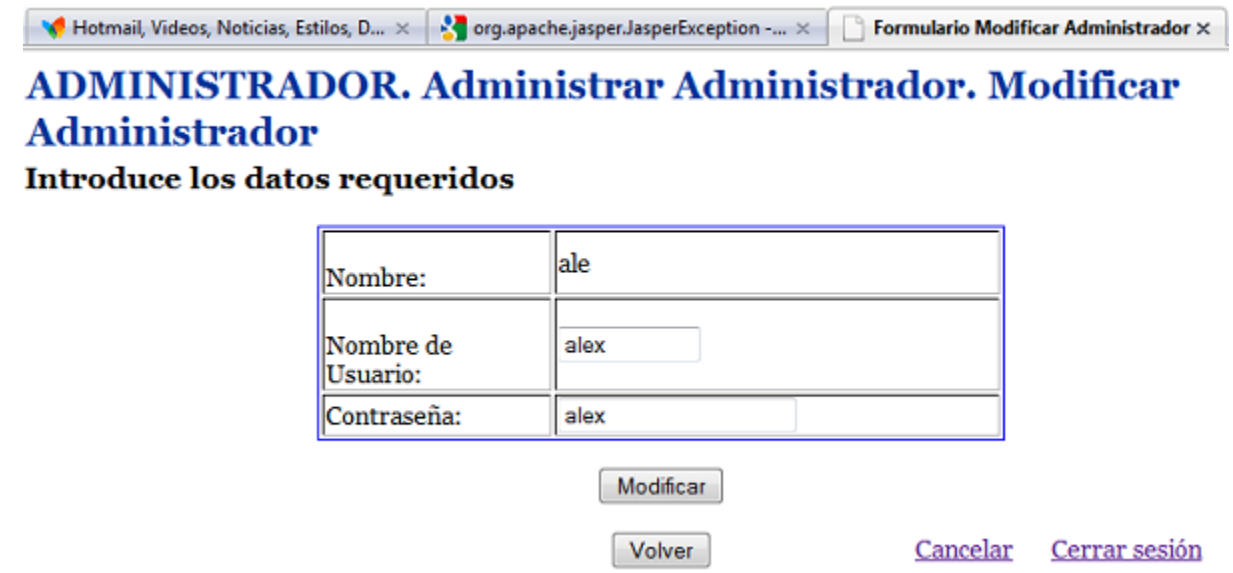

5.3 Modificar Administrador

| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Elige la opción “Modificar Administrador” del menú principal del Administrador | 2 | Muestra el formulario para la búsqueda del administrador a modificar | - |
| 3 | Escribe el nombre del administrador a modificar y presiona el botón “Buscar” | 4 | Realiza la búsqueda en la base de datos | E1, E2 |
| - | - | 5 | Muestra el listado con los resultados de la búsqueda$_3$ | - |
| 6 | Selecciona el registro del administrador que desea modificar y presiona el | 7 | Muestra la pantalla con el formulario de los datos a modificar | E2 |

| | | | | |
|---|---|---|---|---|
| | botón “Modificar”. | | | |
| 8 | Modifica los datos y presiona el botón “Modificar” | 9 | Realiza la modificación de la información en la base de datos. | E2 |
| - | | 10 | Muestra el mensaje de la modificación. | - |
| 11 | Presiona el botón “Página Principal” | 12 | Muestra el menú principal del Administrador | - |

**Prototipo**



5.4 Consultar Administrador

| ACTOR | | SISTEMA | | |
|---|---|---|---|---|
| **Paso** | **Acción** | **Paso** | **Acción** | **Excepción** |
| 1 | Elige la opción “Consultar Administrador” del menú principal del Administrador | 2 | Muestra el formulario para la búsqueda del administrador a consultar | - |
| 3 | Elige el criterio de búsqueda, introduce los datos requeridos en el formulario y presiona el botón “Buscar” | 4 | Realiza la búsqueda en la base de datos | E1, E2 |
| - | - | 5 | Muestra el listado con los resultados de la búsqueda | - |
| 6 | Selecciona el registro del administrador que desea consultar y presiona el botón “Consultar”. | 7 | Muestra la pantalla con el formulario de los datos$_4$ | E2 |
| 8 | Presiona el botón “Página Principal” | 9 | Muestra el menú principal del Administrador | - |

**Excepciones:**

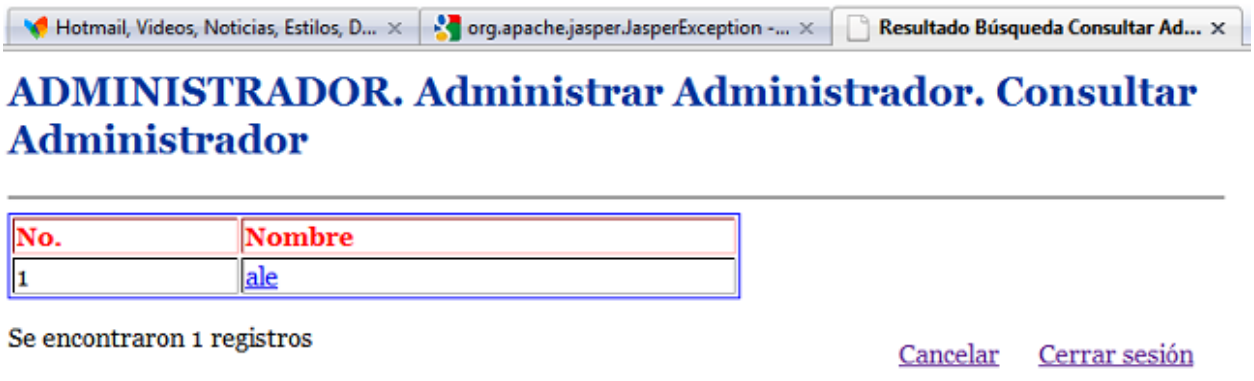
| ID | NOMBRE | ACCIÓN |
|---|---|---|
| E1 | Datos faltantes o incorrectos | El sistema presenta un mensaje de error |

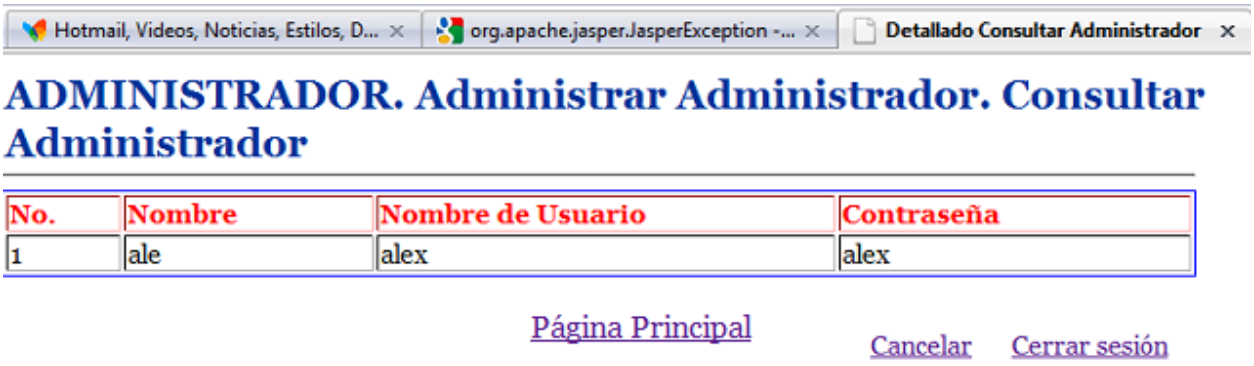
| | | |
|---|---|---|
| | | solicitando que se vuelva a introducir los datos. |
| E2 | Fallo en la conexión con la BD | El sistema presenta un mensaje de error solicitando se intente nuevamente más tarde. |

**Poscondición**: El administrador dio de alta, baja, modificó o consultó los datos de un administrador.

**Prototipo**








**Plan de Prueba**

| Caso de Uso | Entradas | Resultados Esperados |
|---|---|---|
| 5.1 | Se introducen los datos correctos y se presiona el botón “Dar de Alta Administrador”. | El sistema muestra pantalla de confirmación de alta del registro. |
| 5.1 | Se introducen datos incompletos o incorrectos y se presiona el botón “Dar de Alta Administrador”. | El sistema muestra una pantalla solicitando nuevamente los datos. |
| 5.1 | Se introducen datos y no se presiona el botón “Dar de Alta Administrador”. | El sistema no envía los datos y mantiene el formulario en su estado actual. |
| 5.2 | Se introducen criterios de búsqueda adecuados. | El sistema muestra los resultados de la búsqueda. |
| 5.2 | Se introducen criterios de búsqueda incompletos o inadecuados. | El sistema solicita la reformulación de la búsqueda. |
| 5.2 | Selecciona un registro y presiona el botón “Dar de Baja”. | El sistema muestra la pantalla de confirmación de eliminación el registro. |
| 5.2 | Selecciona un registro sin presionar ningún botón. | El sistema no cambia el estado. |
| 5.2 | Confirma la eliminación de un registro, presionando el botón “Si” | El sistema elimina el registro. |
| 5.2 | Cancela la eliminación de un registro. | El sistema no elimina el registro. |
| 5.3 | Se confirma la modificación de los datos de un registro. | El sistema modifica el registro con los datos nuevos. |
| 5.3 | Se cancela la modificación de un registro. | El sistema no realiza ninguna modificación. |
| 5.4 | Selecciona un registro para consultar. | El sistema muestra el registro. |
| 5.4 | No se selecciona ningún registro para consultar. | El sistema no cambia de estado. |

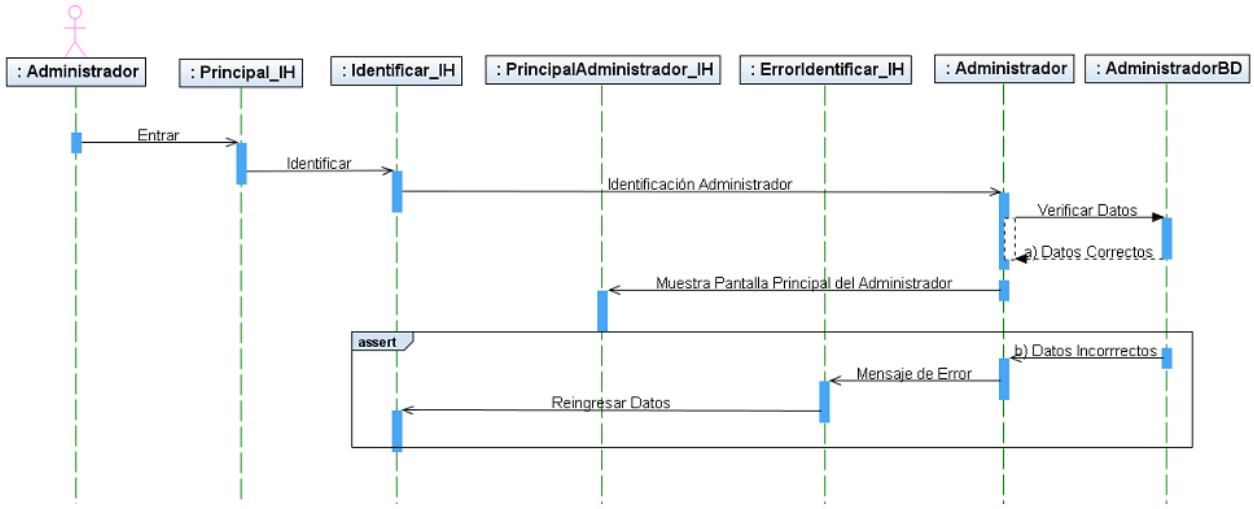
## B.2 Diagramas de Secuencia

Caso de Uso: Entrar

Este caso sólo despliega la página principal del sistema que permite a un usuario no registrado consultar o proponer buenas prácticas.

Caso de Uso: Identificar

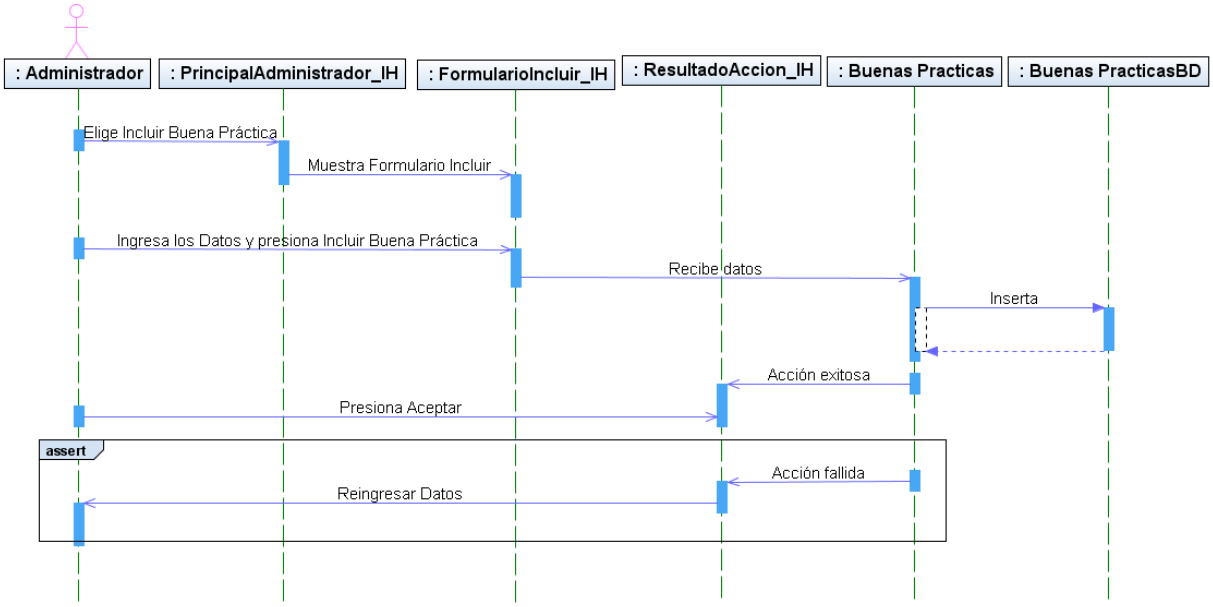
Únicamente el administrador podrá autenticarse para tener ciertas funcionalidades que otros usuarios no tendrán. La clase de control verificará que el nombre de usuario y contraseña sean correctos, para poder entrar a la página principal del administrador.




Caso de Uso: Incluir Buena Práctica

Para incluir una Buena Práctica se ingresan los datos, se validan y la clase Buenas Prácticas incluye la buena práctica.

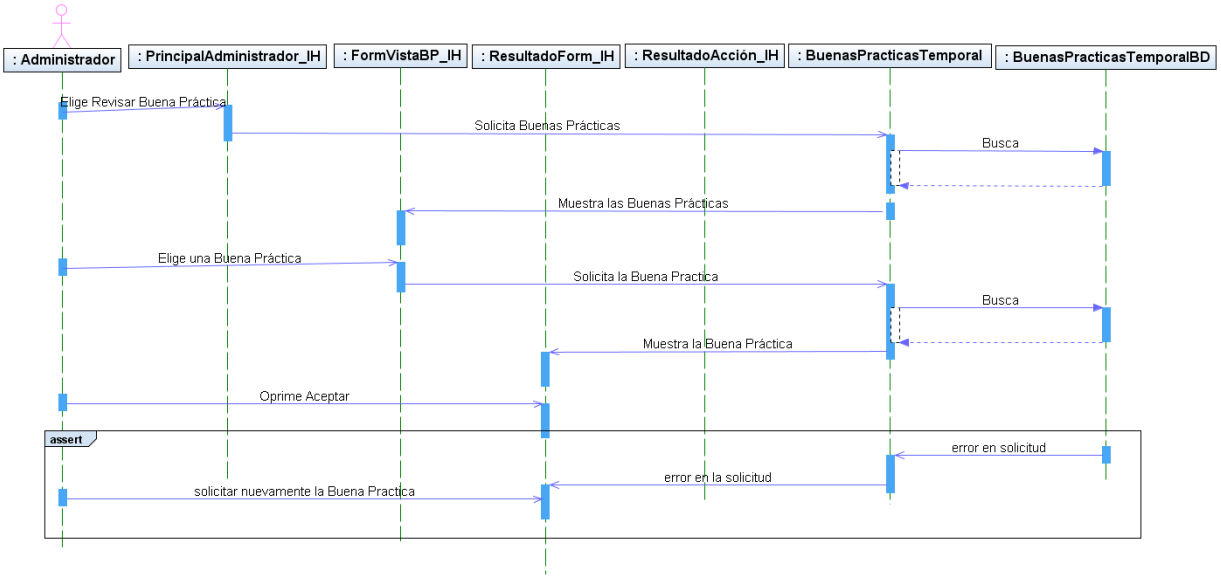
Si existe algún error con la conexión a la base de datos, se mostrará un mensaje para informar al administrador.




Caso de Uso: Recibir Buena Práctica

El administrador solicita las buenas prácticas enviadas por los ponentes, la clase Buenas Prácticas Temporal las muestras.

Si existe algún error con la conexión se muestra un mensaje.



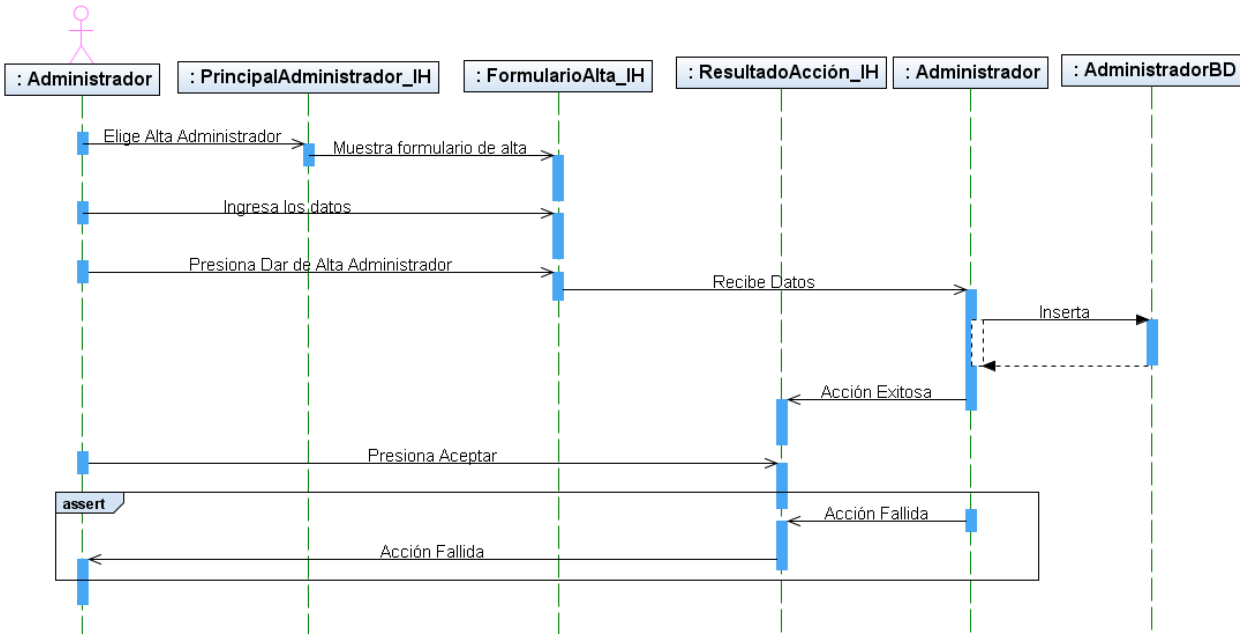
Caso de Uso: Administrar Administradores

El administrador será el encargado de administrar a los diferentes administradores, podrá, dar de alta, de baja, modificar y consultar a los administradores.

a. Alta Administrador

Para dar de alta a un administrador se ingresan los datos, se validan y la clase administrador da de alta al administrador.

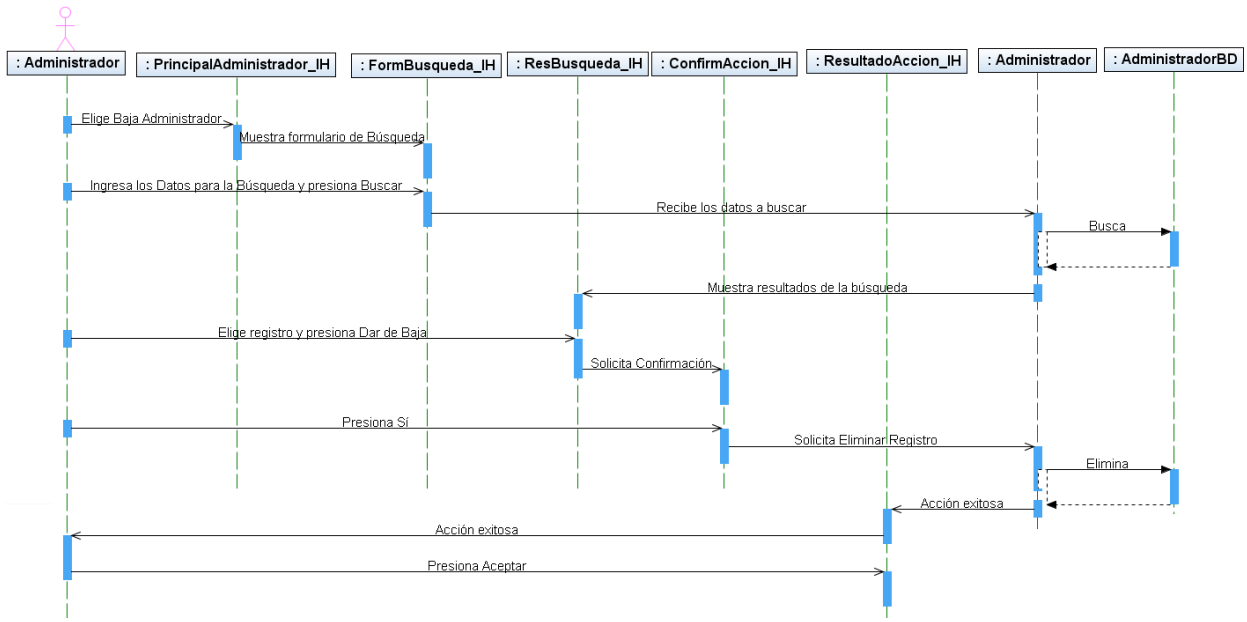
Si existe algún error con la conexión a la base de datos, se mostrará un mensaje para informar al administrador.



b. Baja Administrador

Para dar de baja a un administrador, se busca los datos, se pide una confirmación, se elimina los datos de la base de datos y sale de la sesión.

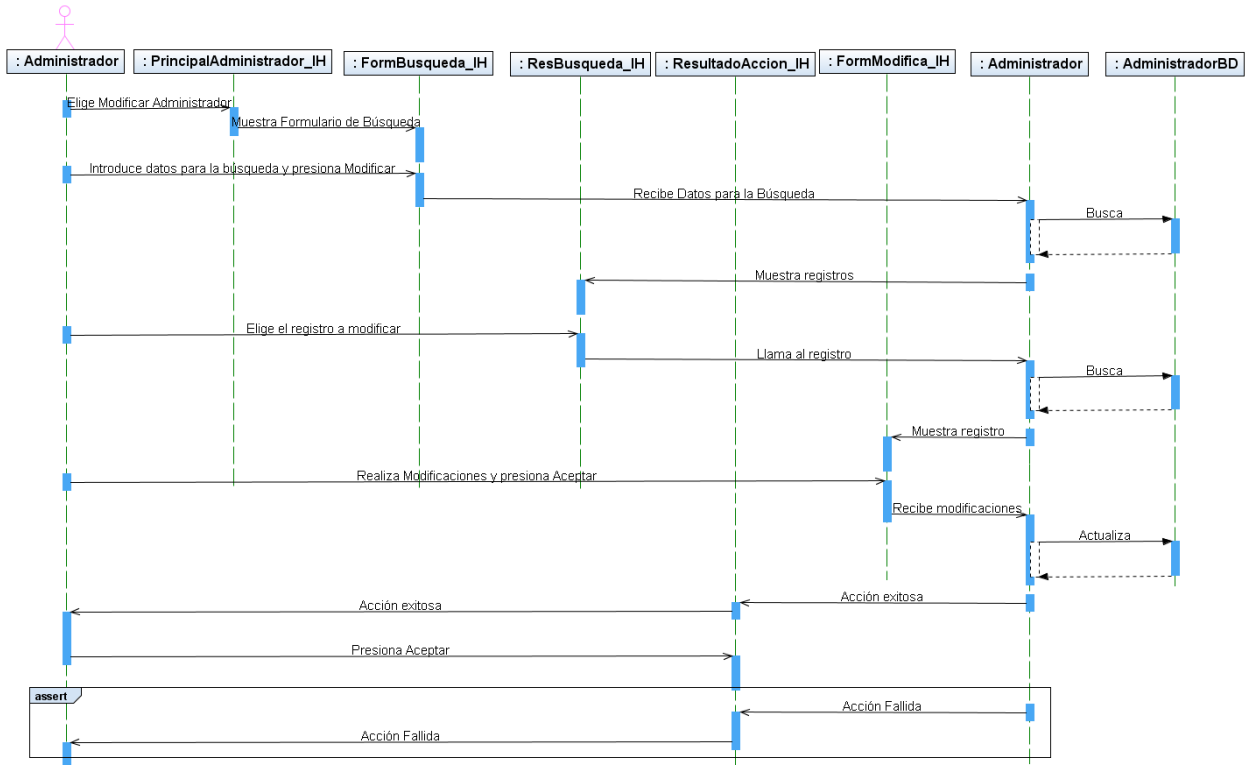
Si existe algún error con la conexión a la base de datos, se mostrará un mensaje para informar al administrador.



c. Modificar Administrador

Se despliega una interfaz con los datos que se encuentran en la base de datos, el administrador modifica los datos, se validan, finalmente se le informa al administrador que la operación se realizó exitosamente.

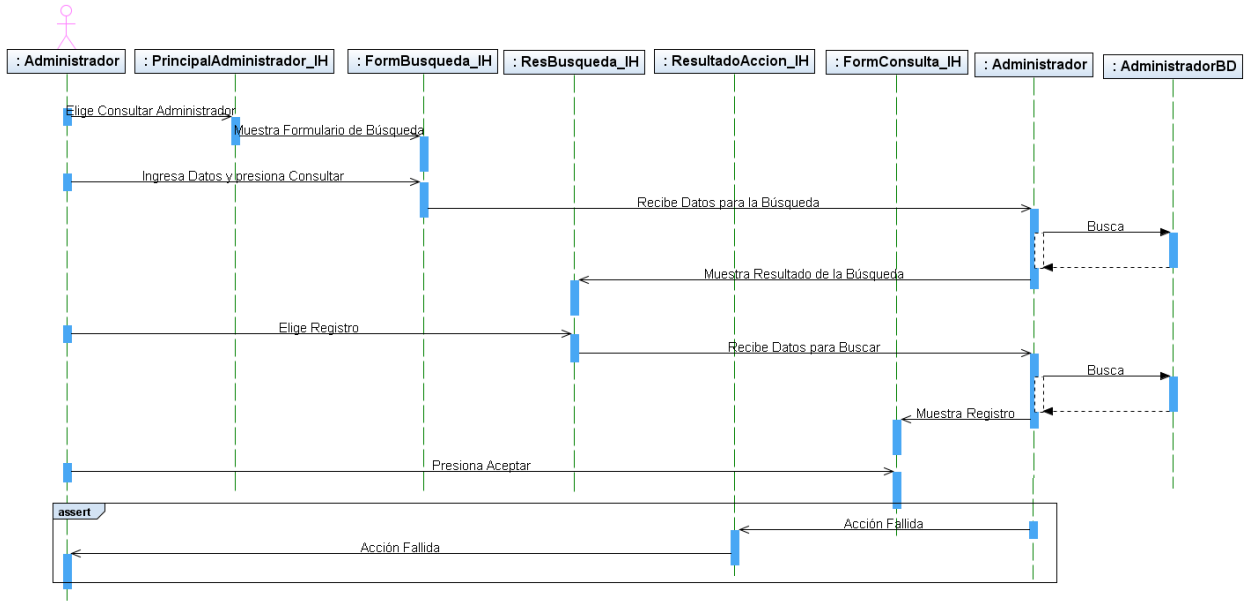
Si existe algún error con la conexión a la base de datos, se mostrará un mensaje para informar al administrador.




d. Consultar Administrador

Se despliega la información, la cual no se podrá verificar.

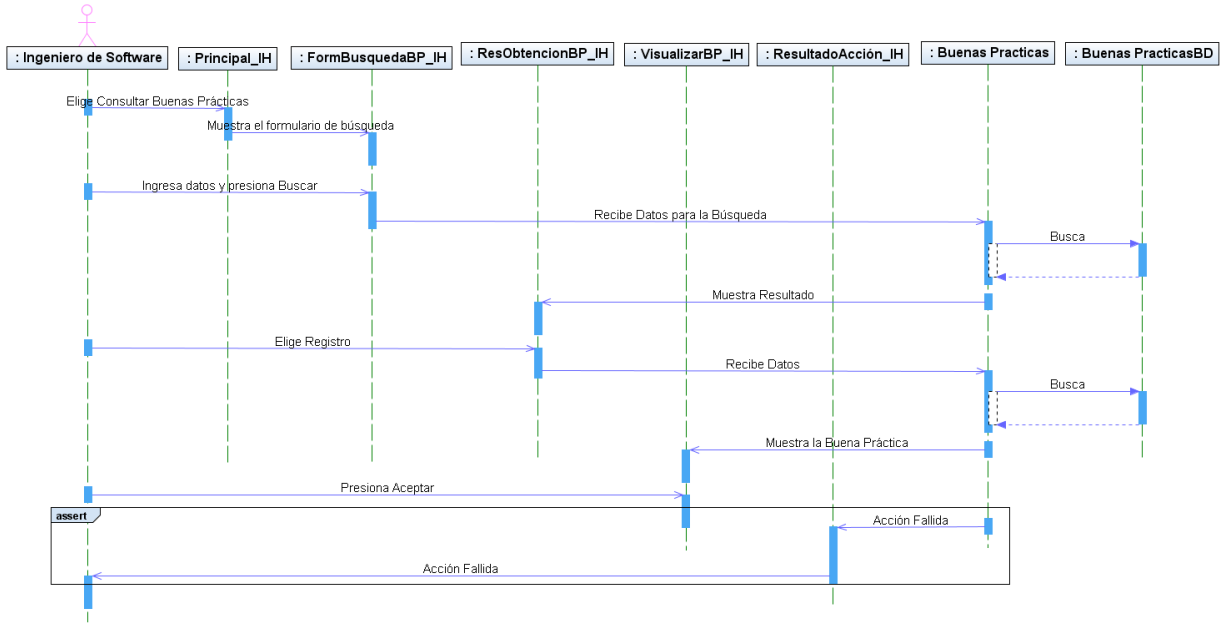
Si existe algún error con la conexión a la base de datos, se mostrará un mensaje para informar al administrador.




Caso de Uso: Consultar Buenas Prácticas

Lee y despliega los datos de todas las buenas prácticas almacenadas en la base de datos, según el criterio seleccionado: Área, Rol, Subárea, Título, Problema, Autor.

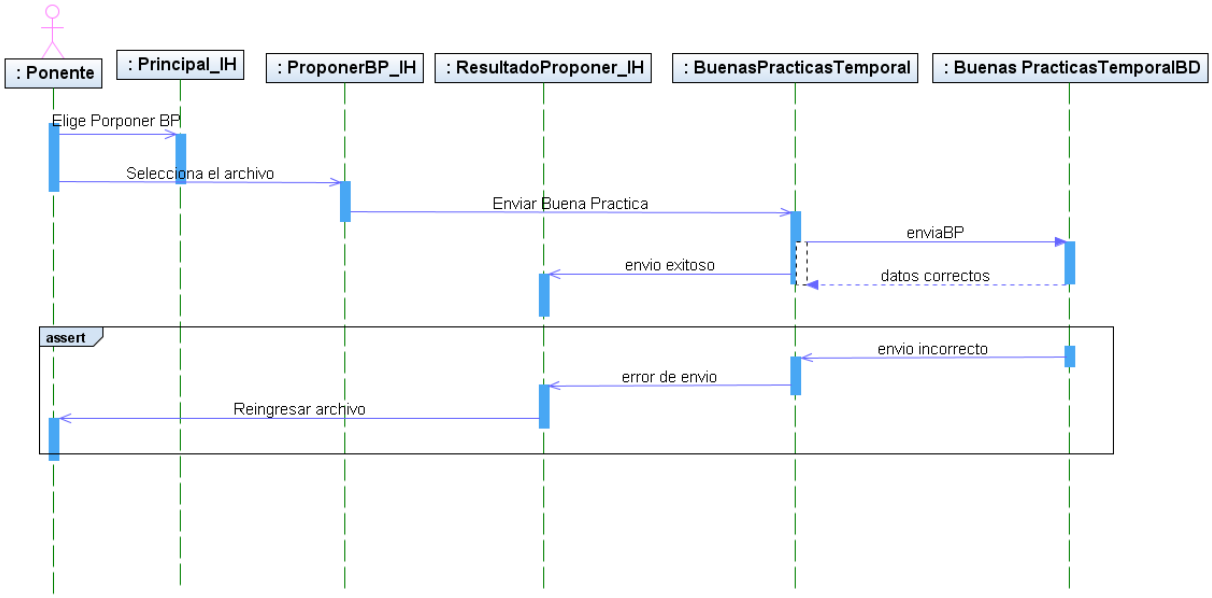
Si existe algún error con la conexión a la base de datos, se mostrará un mensaje para informar al usuario.




Caso de Uso: Proponer Buena Práctica

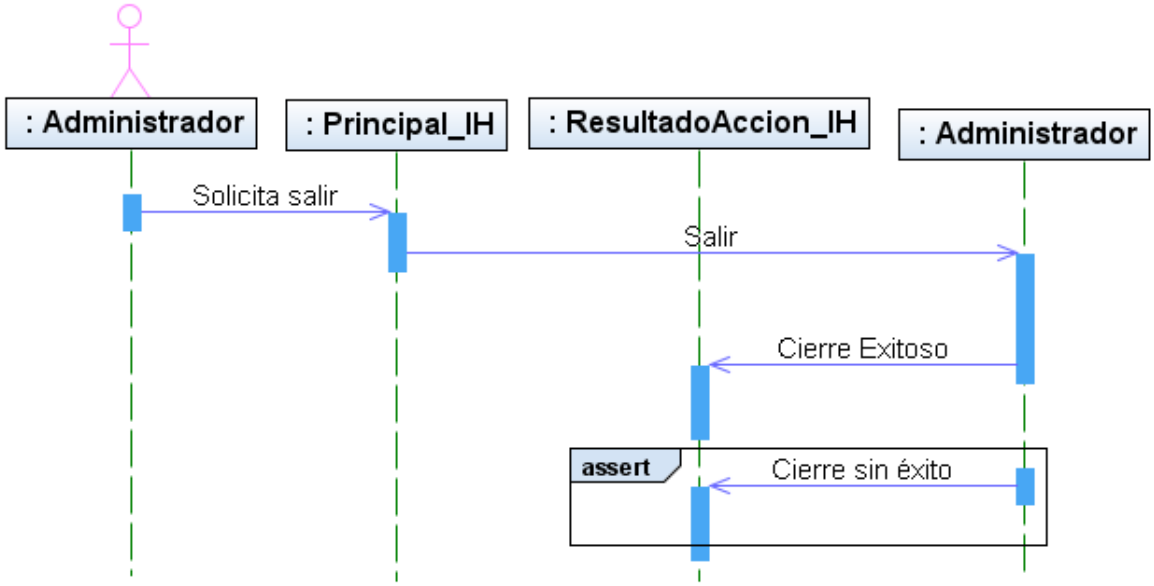
El ponente selecciona y envía su buena práctica, el sistema verifica los datos y los almacena de manera temporal.

En caso de exista algún error se mandará un mensaje al ponente, para iformarle de dicho error.




Caso de Uso: Salir

Se cierra el sistema, ya no hay manera de realizar ninguna modificación o consulta.



# REFERENCIAS

[1] Vega Lebrún, Rivera Prieto y García Santillán**:** (2008) Mejores prácticas para el establecimiento y aseguramiento de la calidad de software, Edición electrónica gratuita. Texto completo en www.eumed.net/libros/2008a/351/

[2] ITSON Instituto Tecnológico de Sonora “Educar para Trascender”. Obtenida el 27 de febrero de 2009, de http://www.itson.mx/dm/amacias/documentos/Lectura%20RUP.doc

[3] Red Universitaria de Colaboración en Ingeniería de Software y Bases de Datos. Obtenida el 17 de enero de 2009, de http://www.redisybd.unam.mx/

[4] Obtenida el 16 de septiembre de 2009, de http://www.certificacion-pmi.com.ar/certificacion_pmi_datos/project_management.htm

[5] Buenas Prácticas. Obtenida el 15 de enero de 2009, de http://www.feaps.org/calidad/buenas_practicas_que.htm

[6] Steven Feuerstein; Oracle PL/SQL Best Practices; Editorial: O'Reilly.

[7] CYTED Ciencia y Tecnología para el Desarrollo. Obtenida el 21 de septiembre de 2009, de http://*www.*alarcos.inf-cr.uclm.es/Competisoft/

[8] Jacobson Ivar, Booch Grady, Rumbaugh James. El Proceso Unificado de Desarrollo de Software. Pearson Addisson-Wesley. Año 2000.

[9] PMI Standards Committee. A guide to the Project Management Body of Knowledge (PMBOK) Project Management Institute. Version 3.

[10] Tesis. Rodríguez Hernández, Jonathan. “Componentes de Bases de Datos para MoProSoft”. 2008

[11] Christopher Alexander. A Pattern Language: Towns/Buildings/Construction. Oxford University Press, Inc. Año 1977.

[12] Gamma Erich, Helm Richard; Design Patterns: Elements of Reusable Object-Oriented Software (Hardcover); Ed. Addison Wesley Professional Computing Series. Año1995.

[13] Artículo. Sánchez, Jorge. “Java 2. Incluye Swing, Threads, programación en red, JavaBeans, JDBC y JSP/Servlets; Año 2004”

[14] Scribd. Obtenida el 1 de octubre del 2009, de http://www.scribd.com/doc/7411856/Caracteristicas-de-C

[15] Diccionario Online. Obtenida el 4 de octubre de 2009, de http://www.zonagratuita.com/diccionario/p/php.htm

[16] Bases de Datos. Obtenida el 17 de febrero de 2009, de http://132.248.181.234/sbd/notas.htm

[17] Aula Clic. Curso de SQL. Obtenida el 3 de marzo de 2009, de http://www.aulaclic.es/sql/b_8_1_1.htm

[18] Tesis. García Velasco, Rosa Elba. “Utilización de XML en el desarrollo de un sistema con datos semiestructurados”. 2006.

[19] Bases de Datos XML. Obtenida el 10 de marzo de 2009, de http://www.info-ab.uclm.es/asignaturas/300219/pdf/BBDDXML.pdf

[20] Introducción a Postgresql. Obtenida el 20 de mayo de 2009, de http://www.eqsoft.net/presentas/introduccion_a_postgresql.pdf

[21] Aplicaciones Empresariales. Obtenida el 22 de mayo de 2009, de http://www.aplicacionesempresariales.com/version-83-de-postgresql.html

[22] Wikipedia. Obtenida el 22 de mayo de 2009, de http://es.wikipedia.org/wiki/Microsoft_SQL_Server

[23] Gabillaud, Jérome. SQL Server 2005, SQL, Transact SQL. Ed. Software SL. Año 2006.

[24] Zonaoracle, Obtenida el 20 de agosto de 2009, de http://www.zonaoracle.com/manuales-tutoriales-oracle/?id=185

[25] Paper. Obtenida el 21 de mayo de 2010, de http://www.iessanvicente.com/colaboraciones/oracle.pdf

[26] eXist. Obtenida el 27 de mayo de 2009, de http://exist.sourceforge.net/

[27] Xíndice. Obtenida el 29 de mayo de 2009, de http://xml.apache.org/xindice/

[28] Carey Michael, Florescu Daniela; XPERANTO: Publishing Object-Relational Data as XML; IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120