ESCUELA POLITECNICA DEL EJÉRCITO

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

APLICACIÓN DE LA METODOLOGÍA MSF V4.0 A LA DEFINICIÓN E IMPLEMENTACIÓN DE ARQUITECTURAS ORIENTADAS A OBJETOS EN VISUAL STUDIO .NET 2005, CASO PRÁCTICO G5 SHARING FILES

Previa a la obtención del Título de:

INGENIERO EN SISTEMAS E INFORMATICA

POR: VILLARROEL GONZALEZ LUIS RAMIRO
MONTALVO YÉPEZ CÉSAR ALEJANDRO

SANGOLQUI, 25 de Febrero de 2008

CERTIFICACIÓN

Certifico que el siguiente trabajo fue realizado en su totalidad por el (los) Sr(s). César Alejandro Montalvo Yépez y el Señor Luis Ramiro Villarroel González como requerimiento parcial a la obtención del título de INGENIERO(S) EN SISTEMAS E INFORMÁTICA.

Sangolquí, Diciembre del 2007

_____	_____
Ing. Jenny Ruiz	Ing Geovanny Raura
Directora de Tesis	Codirector de Tesis

DEDICATORIA

César Alejandro Montalvo Yépez

El presente trabajo de tesis está dedicado a mis padres, abuelitos y amigos, que a lo largo de todos estos años de estudio han estado apoyándome y animándome a conseguir el objetivo de terminar la carrera de Ingeniería en Sistemas e Informática.

Luis Ramiro Villarroel González

Dedico el presente trabajo a mis padres, quien con su apoyo y confianza incondicional han permitido que pueda realizarme como profesional. Al resto de mi familia, en especial a mi abuela quien siempre me ha dado ánimos para continuar en busca del anhelado objetivo. Y por último a Cesar, quien ha sido como un hermano para mí en estos años.

AGRADECIMIENTO

Agradecemos a nuestros padres, familiares y amigos por contribuir en nuestro crecimiento intelectual y emocional a lo largo de estos años. A nuestros educadores que con su esfuerzo y pasión han logrado crear en nosotros amor por la investigación y autoformación, cruciales para el buen desempeño en la carrera. En especial agradecemos a la Ing. Jenny Ruiz, que con sus sabios consejos y persistencia ha influenciado en el buen desarrollo del presente trabajo de tesis. Al Ing. Edison Lazcano, quien desinteresadamente ayudó a perfeccionar el presente proyecto. Al Ing. Geovanny Raura por aportar con su vasta experiencia y conocimientos metodológicos al trabajo. Y por último Ing. Ramiro Delgado, educador y amigo que durante toda la carrera nos ha dado oportunidades y conocimientos para crecer profesionalmente.

INDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

Resumen

El presente trabajo de tesis explica de forma clara los procesos necesarios para la construcción de sistemas orientados a objetos utilizando como metodología de trabajo MSF V4.0 (Microsoft Solution Framework). MSF V4.0 perite de integrar de forma sencilla los dos modelos tradicionales de desarrollo de sistemas, es decir, el modelo en cascada y el modelo en espiral; es por eso que MSF V4.0 se convierte en una de las mejores metodologías para el desarrollo de sistemas.

Cabe resaltar que para sistemas de gran tamaño esta metodología presenta pequeños riesgos debido a que está orientada para proyectos pequeños o medianos, esto principalmente a que la documentación requerida no es muy detallada y no se emplean todos los diagramas que UML ofrece. Otro de los factores que dificulta el trabajo en proyectos grandes es la gran cantidad de retroalimentación que MSF V4.0 propone ya debido a que en cada etapa se desarrollan aplicativos funcionales, los mismos que son revisados y validados por el equipo de pruebas y finalmente por el usuario el mismo que puede sugerir cambios, o en el caso extremo, solicitar cambios en la funcionalidad. Lo cual genera costos y tempos de retraso en el cronograma del proyecto.

Para terminar esta tesis resume y expone una arquitectura orientada a objetos utilizando como metodología de desarrollo MSF V4.0, la arquitectura expuesta se basa en una arquitectura n-capas, orientada a objetos y altamente escalable. La misma que ha sido implantada en un caso práctico, el sistema G5 Sharing Files, él mismo que permite a los usuarios compartir archivos a través de un portal Web, manteniendo siempre las diferentes versiones de los archivos.

1. CAPÍULO I - INTRODUCCIÓN

La mayoría de las organizaciones reconocen la importancia que la tecnología de la información (TI) desempeña en sus objetivos de negocios. Para lo cual están desarrollando la infraestructura de TI necesaria para poder enfrentar un entorno que es cada vez más hostil; los ataques se efectúan con mayor frecuencia y exigen un tiempo de reacción más breve. Con frecuencia las organizaciones no pueden reaccionar ante las nuevas amenazas de seguridad antes de que afecten a su negocio. La administración de la seguridad de sus infraestructuras, y el valor de negocio que ofrecen, se ha convertido en una preocupación primordial para los departamentos de TI.

Microsoft Solution Framework (MSF) es una herramienta que ayuda a llevar a cabo con éxito los planes de acción creados como parte del proceso de administración de riesgos de seguridad de Microsoft. Se ha diseñado para ayudar a las organizaciones a ofrecer soluciones de tecnología de alta calidad puntuales y según lo presupuestado. MSF sintetiza un enfoque bien disciplinado orientado hacia los proyectos tecnológicos basándose en un conjunto definido de principios, modelos, disciplinas, conceptos, orientaciones y prácticas demostradas de Microsoft.

Solamente basándose en una metodología robusta como MSF y en arquitecturas que desde años han sido implementadas con éxito con esta y otras casas, se puede garantizar soluciones prácticas, seguras, escalables y desarrolladas en tiempos mínimos.

El objetivo del presente trabajo es mostrar de qué forma se puede combinar la programación orientada a objetos, con MSF, y mostrarlo en teoría y práctica, a través de G5 Sharing Files.

Gracias a la gran acogida del Internet en el mundo, hoy en día es posible compartir y acceder a información en cualquier momento y lugar. Que mejor oportunidad para desarrollar un sistema que permita la administración y control de los diferentes documentos que un ejecutivo puede llegar a necesitar en un día de trabajo normal, todo esto sin tener que regresar a su oficina a buscar el mismo; simplemente utilizando Internet.

Es decir, poseer una herramienta que permitirá organizar, compartir, y autorizar o restringir el acceso a cualquier tipo de documentos desde el Internet, de manera clara y sencilla. De esta forma será fácil y económico para una empresa tener al detalle las últimas versiones de sus documentos accesibles desde cualquier parte del mundo.

Además de lo expuesto anteriormente, la herramienta persigue un segundo objetivo claro, este es el manejo del control de cambios realizados sobre los archivos. Esto significa que cada actualización a los archivos de la empresa será guardada, y que los documentos importantes de la empresa solo podrán ser borrados por el administrador. De esta manera se reducirá de forma significativa la pérdida de información ya sea por entes maliciosos o por descuido del personal.

Combinada ambas funcionalidad, tendremos pues una herramienta integral que ayudará a la empresa a sacar el máximo provecho a la tecnología sabiendo que su información estará segura y disponible desde cualquier parte del mundo.

1.1. Justificación e Importancia

Hoy en día MSF se presenta como una solución viable como metodología para el desarrollo de proyectos de software, al igual que .NET framework V2.0 lo es como background de desarrollo.

Esto supone explorar a fondo MSF, para mostrar los pasos que se deben seguir según esta metodología para el desarrollo eficiente de un proyecto de software.

De la misma manera, es importante definir una arquitectura robusta, que permita desarrollar aplicaciones de fácil mantenimiento y escalables utilizando .Net framework V2.0

Hay dos hechos que son verdades dentro de la sociedad actual, el primero es que prácticamente toda la información importante de la empresa mediana o grande se encuentra en formato digital y lógicamente es llevada a través de un ordenador. El segundo que el Internet es ya una realidad en Ecuador y que de igual forma toda empresa mediana o grande tiene este servicio en sus oficinas.

Partiendo de estos hechos, se ve claramente justificado el desarrollo de un sistema para empresas que tengan más de una oficina, o cuyos actores encargados de modificar los documentos estén constantemente de viaje, de manera que se pueda compartir la información de la empresa de una manera rápida, segura y sencilla, de manera que los usuarios registrados al mismo puedan, descargar y actualizar los documentos.

Debido a la necesidad de compartir archivos presentados en varias empresas, la ONG Asocam ha decidido invertir en la creación de un Sistema que facilite esta tarea.

1.2 Objetivos

1.2.1 Objetivo General

- Investigar la metodología MSF (Microsoft Solution Framework) y aplicarla en el desarrollo de aplicativos con arquitecturas orientadas a objetos.

1.2.2 Objetivos Específicos

- Investigar acerca de las ventajas y desventajas de Microsoft Solution Framework.
- Utilizar las buenas técnicas de patrones y prácticas para la concepción de una arquitectura robusta orientada a objetos para .NET Framework Versión 2.0
- Adecuar el uso de Microsoft Solution Framework para el desarrollo de sistemas con arquitecturas orientadas a objetos.
- Usar los conceptos aprendidos durante los estudios universitarios para el desarrollo eficiente y eficaz de software.
- Utilizar la metodología de desarrollo Microsoft Solution Framework (MSF) en el proceso de desarrollo del sistema G5SharingFiles.

1.3 Alcance

Desarrollo del marco teórico de referencia, el mismo que investigará, analizará y adecuará, la metodología Microsoft Solution Framework para el desarrollo de aplicativos con arquitecturas orientados a objetos.

Estudio de los diferentes conceptos que intervienen en el desarrollo de arquitecturas orientadas a objetos, y desarrollo e implementación de una arquitectura robusta, eficiente y eficaz que funcione sobre .Net Framework 2.0

Desarrollo del sistema G5SharingFiles utilizando la metodología MSF y una arquitectura robusta orientada a objetos para que cumpla las siguientes funcionalidades:

- Control de usuarios: Permitirá la creación, actualización y eliminación de usuarios, los cuales van a poder pertenecer a los siguientes roles: administrador o usuario normal.
- Creación de carpetas y subcarpetas para almacenar los documentos en estructura de árbol, de tipo N padres con N hijos en N niveles.
- Administración de permisos sobre las carpetas, de forma que un usuario registrado o un grupo de usuarios puedan o no realizar las siguientes acciones:
    - Creación de archivos en la carpeta seleccionada.
    - Actualización de los archivos de la carpeta seleccionada.
    - Eliminación de los archivos de la carpeta seleccionada.
        - Papelera de reciclaje de los archivos eliminados, donde únicamente el administrador pueda eliminarlos y los demás usuarios tengan acceso a los archivos eliminados.
        - Llevar un control historial de las modificaciones que se han hecho a los archivos con las versiones de cada fecha de cada archivo.
        - Buscador para encontrar a los archivos por título o por palabras que se encuentren dentro de los archivos.

2. CAPÍTULO II - MARCO TEÓRICO DE REFERENCIA

2.1. Introducción a MSF V 4.0 ¿Qué es MSF?

2.1.1. Breve resumen del modelo

Microsoft Solution Framework (MSF), es un modelo de procesos que combina dos modelos muy comunes en proyectos de desarrollo, el modelo en cascada y el modelo en espiral.

Consta de cinco etapas, en las cuales se generan entregables concretos que ayudan a resolver los requerimientos del cliente de una manera objetiva. Las cinco etapas de MSF son:

- El "envisionamiento" del proyecto, en donde todo el equipo va a tener una idea clara, pero general de los objetivos, infraestructura, tecnología, arquitectura, roles y riesgos todos estos necesarios para el buen desenvolvimiento del proyecto.

- La planificación del proyecto, es la etapa donde se levantarán los requerimientos específicos del cliente, tomando en cuenta que MSF no es cerrado, es decir que permite cambios dentro del proyecto, incluso en la etapa de desarrollo.

- El desarrollo del proyecto, que es la etapa donde propiamente se genera el código necesario para generar un producto funcional para el cliente.

- La fase de estabilización, que es donde se prueba al proyecto en ambientes simulados, que deben ser los más parecidos a los ambientes reales, y se corrigen, en caso de haber, los errores del aplicativo.

- Y por último la implementación, que es la fase donde el proyecto será puesto en el ambiente de producción, para que cumpla el fin por el cual fue creado.

Para entender el modelo MSF, es importante entender a claridad el modelo en cascada, y el modelo en espiral, así como las ventajas y desventajas de estos, luego se podrá visualizar de qué manera MSF junta las mejores prácticas de ambos para dar como resultado un modelo altamente práctico.

2.1.2. El modelo en cascada

El modelo en cascada se representa por el gráfico de la figura 1.

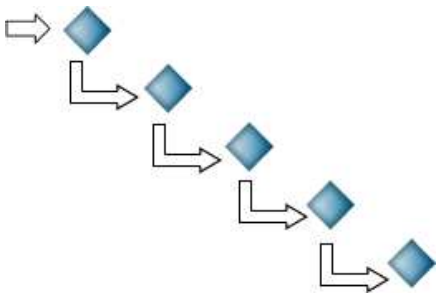


Figura 1. Representación gráfica del modelo en cascada

En la figura 1, se representa cada etapa del proyecto con un rombo. El modelo en cascada se basa en que se avanza a la siguiente fase del proyecto, siempre y cuando la etapa anterior del proyecto está finalizada por completo.

Así mismo, una vez que se ha pasado a la siguiente etapa del proyecto, no se admite un retroceso a la fase anterior.

Esto funciona bien para proyectos donde claramente se pueden delinear los requerimientos (no modificables) del proyecto en una fase inicial. Estas etapas cerradas, facilitan el planeamiento y asignación de recursos para cada etapa. Sin embargo se complica si los requerimientos tomados en un inicio se modifican en una etapa del proyecto posterior.

2.1.3. El modelo en espiral

El modelo en espiral se representa por el gráfico de la figura 2

Figura 2. Representación gráfica del modelo en espiral

En la figura 2 se representa el modelo en espiral. El modelo en espiral no define etapas claras dentro del desarrollo del proyecto, sin embargo está abierto a cambios por el cliente en cualquier momento. Es decir se trata de un

desarrollo a la par con el levantamiento de requerimientos, ya que el cliente provee retroalimentación en cualquier etapa del proyecto.

Esto puede ser muy efectivo cuando se necesita un desarrollo rápido en aplicaciones sumamente pequeñas, pero deja abierta la posibilidad de catástrofes en el proyecto por dos razones. La primera, la tarea de por sí durísima, de medir los recursos necesarios para el desarrollo, se complica aún más. La segunda, al no haber puntos de control, la tarea de desarrollo, aunque se esté trabajando con un equipo pequeño, se puede volver algo caótico.

2.1.4. El modelo propuesto por MSF

El modelo propuesto por MSF, no es un invento nuevo, ni mucho menos, es más bien tomar las ventajas de los dos modelos anteriormente expuestos con el fin de solucionar los problemas vistos en los apartados anteriores, y por otro lado asimilar las ventajas que estos mimos modelos dan. Así tenemos en la siguiente figura, la representación gráfica del modelo MSF.

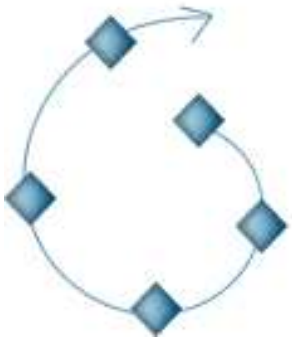


Figura 3. Representación gráfica del modelo propuesto por MSF

En la figura, cada rombo representa un entregable dado con la finalización de cada etapa, sin embargo este entregable puede ser fácilmente modificado en caso de requerirse sin que el proyecto se detenga, es decir, es un modelo abierto, tal cual el espiral, que permite volver a etapas previas del proyecto, por diferentes razones que la experiencia nos dice que casi siempre se van a dar, y a la vez existen puntos de control específicos que permiten tener control sobre el avance del proyecto, y poder crear una planificación clara acerca de los recursos estimados para el cumplimiento en plazos y costos estables.

En la figura 3, se han puesto cinco rombos, ya que estos representan la etapa cumbre de cada uno de las fases propuestas por MSF descritas anteriormente en el breve resumen del modelo, vale la pena repetirlas:

- "Envisionamiento"
- Planificación
- Desarrollo
- Estabilización
- Implementación

De la misma forma cada rombo, marca la culminación de cada etapa, y que el modelo permite crear puntos de control. De esta forma se colocan entregables específicos que indican si una etapa está terminada, (aunque puede ser abierta en caso necesario, ya que el modelo permite volver a etapas previas). La siguiente tabla muestra los entregables específicos para cada una de las etapas:

Fase	Entregable
Envisionamiento	Documento de Visión y Alcance.
Planificación	Documento del Plan del Proyecto.
Desarrollo	Alcance completado.
Estabilización	Release aprobado.
Implementación	Proyecto puesto en producción.

Tabla1 Entregables al finalizar cada fase propuesta por MSF

2.1.5.Objetivos del modelo propuesto por MSF

- Proveer de una herramienta, sobre todo práctica para la administración de proyectos de desarrollo.
- Brindar un recurso fácilmente aplicable al mundo real, que permita establecer los roles de las personas que van a trabajar en el proyecto.

- Definir claramente los recursos y metas del proyecto y manejar una guía que nos ayude a cumplir esas metas con los recursos especificados.

- Desarrollar un aplicativo escalable, que se adecue a las necesidades del cliente, sin que esto involucre retrasos en el proyecto, tomando en cuenta los riesgos del proyecto.

- En fin, MSF involucra todo un framework guía, que indica paso a paso las mejores prácticas para el desarrollo rápido y fiable de un proyecto de desarrollo de software. Recomendado, para proyectos pequeños y medianos, y que va desde el cómo levantar los requerimientos, hasta la puesta en producción del proyecto.

- En los siguientes apartados, se explicará al detalle las diferentes fases propuestas por MSF, así como las plantillas y documentación propuesta. Al finalizar este marco teórico, mostraremos las ventajas del desarrollo Ágil de aplicaciones, y cómo MSF, en su última versión 4.0 (siendo este documento escrito a finales del 2006), ayuda a este concepto.

2.2. MSF Agile versión 4.0 y Diferencias con su predecesor MSF V 3.1

2.2.1.Qué significa el desarrollo ágil de aplicaciones

El desarrollo ágil de aplicaciones, es un modelo de procesos orientado a invertir la mayor cantidad de recursos en el acto propio del desarrollo, y la menor cantidad en planificación. Sin que esto signifique por ningún motivo un desarrollo desorganizado, forzado o poco predictivo.

Principalmente se utiliza este tipo de desarrollo en proyectos pequeños o medianos, en un ambiente donde el tiempo es el recurso más valioso.

Existen dos tipos de modelos de procesos base, el adaptativo y el predictivo. El modelo predictivo intenta predecir al detalle las tareas a ser cumplidas en la planificación total del proyecto. Partiendo de esta idea, planifica al detalle todos los pasos necesarios para el desarrollo del proyecto, desde el principio hasta el final. Sin embargo, esto trae el problema, de que cambios en la dirección del proyecto pueden traer grandes complicaciones, y esto es muy usual en proyectos de desarrollo, especialmente en medianos o pequeños, donde los requerimientos y necesidades se van viendo "en el camino".

El tipo de modelo adaptativo, es por el contrario muy flexible. Predice a grosso modo la visión global del proyecto, así como los costos del mismo, pero deja abierto cambios en la dirección del mismo, debido a su función iterativa. En este tipo de modelos, se planifica de a poco, y se va sacando nuevas versiones del proyecto por cada funcionalidad nueva agregada. Se trata de segmentar el proyecto en proyectos más pequeños, que igualmente seguirán los pasos de envisionamiento, planificación, desarrollo, pruebas e implementación para concluir con la funcionalidad modular y pasar luego a la siguiente funcionalidad o iteración. La desventaja en proyectos grandes es que este tipo de segmentación e iteración puede ser difícil de controlar, la ventaja es que genera aplicaciones mucho más acordes a los requerimientos del cliente, en un tiempo mucho menor, ya que prácticamente la

planificación y el desarrollo van de la mano. Obviamente el desarrollo ágil de aplicaciones se orienta a este tipo de modelo.

2.2.2.El modelo de procesos orientado a roles para desarrollo ágil de aplicaciones propuesto por MSF Agile

Microsoft Solution Framework 4.0 para desarrollo ágil de aplicaciones, permite seguir el modelo de procesos en forma de iteraciones propuesto. Así para cada iteración tendremos las fases propuestas por su predecesor 3.1. Sin embargo, y la gran diferencia está en que se propone controlar las iteraciones a nivel de funciones que ejerce cada rol, en lugar de las fases por las que tiene que pasar la iteración. Cada fase de su predecesor tendrá uno o varios roles, con sus correspondientes actividades y entregables específicos para cada etapa. En la tabla 2, tenemos los roles que contempla MSF 4.0, y hemos dado la analogía con las fases propuestas en la versión 3.

Fase	Rol
Envisionamiento	Analista del Negocio
Planificación	Manager del Proyecto. Arquitecto.
Desarrollo	Desarrollador.
Estabilización	Tester.
Implementación	Release Manager

Tabla2 Roles asignados a cada fase en MSF

Para la conceptualización del marco teórico, se debe analizar al detalle los procesos de una iteración basándose en las fases propuestas por la versión 3.1 de MSF, a la vez, en cada fase se definirán y explicarán los roles y entregables basándose en la versión 4.0 Ágil.

2.2.3.Especificación de Roles.

Analista del Negocio

La misión principal del analista del negocio es entender y comunicar las necesidades solicitadas para el proyecto. Trabaja directamente con el cliente o cualquier ente relacionado que le ayude a comprender y visionar los requerimientos, mismos que luego se encargará en traducir en actores, escenarios y requerimientos de calidad de servicio que posteriormente serán tomados por el equipo de desarrollo para crear la solución.

Este rol también es el encargado de manejar las expectativas de funcionalidad del sistema. Dentro del proyecto el analista del negocio debe velar por los intereses del cliente, siendo este el ente que valida la experiencia del usuario y el cumplimiento de requerimientos. En fin, se encarga de ser el puente entre el equipo de desarrollo y los usuarios finales.

Administrador del Proyecto

La meta principal del administrador del proyecto es entregar valor de negocio dentro del horario y del presupuesto convenido. El administrador de proyectos debe cumplir las siguientes funciones: planeamiento y programación de actividades que incluyen el desarrollo y las diferentes

iteraciones del proyecto, monitorear y reportar el estado del proyecto e identificar y mitigar los riesgos.

También se espera que el administrador del proyecto interactúe con el analista del negocio para planificar los escenarios y la calidad del servicio que se esperan en cada una de las iteraciones, de la misma forma debe estar en contacto con los arquitectos y los desarrolladores para poder estimar el trabajo, consultar con los encargados de las pruebas para planificarlas, y facilitar la comunicación entre todo el equipo de trabajo.

Arquitecto

La meta principal del arquitecto es la de garantizar el éxito del proyecto diseñando las bases de la aplicación. Esto incluye la definición de los modelos tanto lógicos como físicos de implementación del sistema. El objetivo del arquitecto es reducir la complejidad al dividir al sistema en partes simples y claras. La arquitectura resultante es extremadamente importante ya que no solo dictamina como se va a desarrollar el sistema, sino que también establece los rasgos que harán que el proyecto llegue a ser un éxito. Esto incluye su usabilidad, mantenimiento y escalabilidad, cumpliendo con los estándares de seguridad y reacción ante cambios inesperados en los requerimientos.

Desarrollador

La meta principal del desarrollador es la de implementar la aplicación de a cuerdo a la planificación establecida. También se espera que el desarrollador ayude a especificar las características del diseño físico, la estimación de tiempos y esfuerzo para completar cada una de las tareas,

construir o supervisar la implementación de las características, preparar el producto para la implementación y proveer de experiencia en el campo tecnológico dentro del equipo.

Tester

La meta principal del tester es la de descubrir y comunicar problemas que encuentre con el producto que puedan afectar de forma considerable con su valor. El tester debe comprender el contexto del proyecto y ayudar a otros para poder tomar las decisiones dentro del contexto. La clave del éxito del tester está en encontrar y reportar las fallas del producto al realizar pruebas. El tester debe realizar informes claros de cuáles son las condiciones en las que se presento el problema y cual considera podría ser la solución al mismo. El tester trabaja en conjunto con todo el equipo para establecer los estándares de calidad del producto.

Administrador de Implementación

La meta principal del administrador de implementación es la de manejar la implementación del producto. Es el encargado de coordinar con los operarios el lanzamiento del producto. Es el encargado de crear un plan de lanzamiento y certificar los candidatos a versiones finales para distribución o desarrollo.

2.3. Estrategia y Alcance de un proyecto de desarrollo de Software

2.3.1.¿Qué significa la Estrategia y Alcance de un proyecto?

La Estrategia y Alcance de un proyecto es dar al equipo encargado una visión clara del proyecto, conjuntamente con el cliente empaparse de la lógica del negocio, del problema que se quiere solucionar, de las alternativas propuestas para solucionarlo y de la mejor manera de hacerlo.

Al final de esta etapa se quiere, que tanto el cliente como el equipo de desarrollo tengan una idea clara de los objetivos que persigue el proyecto.

2.3.2.Objetivos específicos de la Estrategia y Alcance

Mediante el envisionamiento se desea identificar los objetivos principales para los cuales la organización desea proveerse del producto a ser desarrollado. En base a esto, se persiguen los siguientes objetivos específicos:

Identificar las metas del proyecto.

- Conocer las expectativas y problemas puntuales que se quieren resolver con el proyecto.

- Formar las bases sobre las cuales el equipo desarrollará el proyecto en etapas posteriores.
- Definir el alcance del proyecto, sobre el cual se hará el planeamiento detallado en etapas posteriores.
- Estimar los recursos necesarios para el desarrollo del proyecto.

2.3.3.Quien realiza la estrategia y alcance

La estrategia y alcance del proyecto es realizada por el Analista del Negocio.

2.3.4.Tareas a ser cumplidas en la Estrategia y Alcance

Crear los escenarios

El escenario es un medio por el cual se define una interacción de un usuario del sistema con el sistema para lograr cumplir una meta específica. Se debe por tanto exponer todas las metas del sistema, para validar todas las funcionalidades del mismo y la manera en la cual se relacionan estas funcionalidades con los usuarios.

Crear los requerimientos de calidad de servicio

Este documento define las características tales como el rendimiento, la accesibilidad, la escalabilidad y la facilidad de mantenimiento, la carga de stress soportado. Estos requerimientos serán las bases sobre el como el sistema debe operar.

Crear el documento de estructura del proyecto

Este documento define la manera en la cual el equipo se organiza y manejará el proyecto. Describe la estructura organizativa, los estándares, procesos y recursos con los que cuenta el proyecto. Especifica el procedimiento para

realizar cambios en la funcionalidad del sistema. Este documento se compone de las siguientes partes:

- El propósito del documento
- Los estimados de horario, costos y entregables.
- La estructura del equipo incluyendo un resumen de los recursos, roles y las responsabilidades de cada integrante, y una lista de contactos.
- El manual de cambios del sistemas

Revisión de riesgos del proyecto

La revisión de riesgos del proyecto comienza con la enumeración de todos los posibles riesgos que pueda tener el proyecto.

Una vez enumerados, se los clasifica, se identifican las causas y las consecuencias del riesgo. Luego se debe valorar los riesgos asignando un porcentaje de probabilidad de que el riesgo ocurra y un valor sobre 10 del impacto que tendrá en el proyecto, multiplicando estas variables, se obtiene la prioridad del riesgo. A continuación se debe crear el plan de contingencia para cada riesgo.

Capturar la visión y alcance del proyecto

La visión y alcance del proyecto es el entregable principal de esta etapa, está compuesto por las siguientes partes:

- El estatuto de la visión, que es una frase que involucra la meta del sistema en un período específico con acciones específicas.
- El estatuto de oportunidades, que comprende un apartado con todas las ventajas que brindará el sistema.

- El alcance, que involucra todos los requerimientos de la empresa, separados en dos grupos, aquellos que van a ser resueltos por la aplicación y aquellos que están fuera del alcance.
- La meta del proyecto que está dividida en el concepto de solución, los perfiles de los usuarios, el resumen de uso, el resumen de requerimientos (dividido a su vez en requerimientos del negocio, usuario y operacionales), por último el criterio de éxito.

2.3.5.Visión del Proyecto

2.3.5.1.Redactar la Visión del Proyecto

La visión del proyecto de software ayuda a comprender de forma clara que es lo que el sistema podrá realizar. Se deben tener en cuenta las siguientes actividades para poder redactar la visión del proyecto de software:

1. **Resumir el ambiente del proyecto.-** En uno o dos párrafos se debe resumir el contexto del proyecto. Esos párrafos deberían describir los eventos que determinan la creación o el mejoramiento de un sistema.
2. **Explicar los Factores de Administración.-** En este punto se deben describir los requerimientos principales o los períodos de tiempo que conducirán al lanzamiento del producto. Debe dejar claro si el proyecto es en base a una fecha límite o a satisfacer una funcionalidad.
3. **Identificar los Usuarios.-** Determinar un grupo de usuarios que serán los que obtengan de ganancias de forma prioritaria del sistema.

4. **Determinar las Ventajas.-** Escriba una proposición sobre el valor del nuevo o mejorado sistema, esta proposición no debe ser mayor a veinte cinco palabras que en su conjunto expongan el valor para el grupo de usuarios seleccionado.

Como resultado de esta etapa tendremos la Visión del Proyecto.

2.3.6.Definición de Personajes

1. **Definición de Roles.-** De los usuarios determinados en la visión de proyecto, seleccione un grupo que interactué con el sistema; si no encuentra mucha información en la visión del proyecto para poder identificar a todos los tipos de usuarios debe buscar mayor información por parte de los posibles usuarios. Por lo general la visión del proyecto contiene roles como usuario o administrador.

2. **Creación de Personajes.-** Reúna información real para crear un personaje. Utilice datos obtenidos de estudios de usabilidad, visitas a los usuarios, grupos de trabajo y estudios de mercado para garantizar que el personaje es real. Agrupe el conocimiento, habilidades y capacidades con los objetivos de cada personaje.

2.3.6.1.Redefinición de los Personajes

Para realizar este punto es necesario que los usuarios revisen los personajes establecidos previamente, para de esta forma poder redefinir los actuales y de ser necesario eliminar o crear nuevos personajes.

2.3.7.Creación del Escenario

Para la correcta creación de escenarios se requiere que los roles y los personajes estén definidos; y se recomienda realizar las siguientes tareas para garantizar los mejores resultados.

2.3.7.1.Lluvia de ideas de Escenarios

De los objetivos del sistema descritos en la creación de personajes ver las diferentes opciones que el personaje tiene para resolver o realizar una tarea. A cada una de estas formas le asignaremos un nombre descriptivo y las agruparemos en un escenario. Debemos asegurarnos que cada escenario este de a cuerdo con la visión del proyecto, caso contrario tendremos que actualizar la visión.

2.3.7.2.Mejoramiento del Ciclo de Vida

1. **Creación de un Prototipo de Día de un Personaje.-** Para cada personaje identificar y describir uno o más periodos de tiempo prototipos en el ciclo de vida del personaje.
2. **Identificar Oportunidades Tecnológicas.-** Para cada día en la vida, identificar oportunidades donde el producto podría ser utilizado para ayudar al personaje a obtener mejor sus tareas. Agregar una entrada en el escenario en el que corresponda cada oportunidad. Busque la calidad de requerimientos de servicios y agréguelos en la lista.

2.3.7.3.Priorización de la Lista de Escenarios

1. **Determine una prioridad total.-** Asigne una prioridad total para cada escenario en la lista. Ordene los escenarios por rango. Coloque al más alto en el principio de la lista. Califique los requerimientos restantes entre ellos.
2. **Resalte los escenarios con mayor prioridad.-** Describa de forma breve a los escenarios de alta prioridad, esta descripción debe contener los suficientes detalles para que los desarrolladores puedan estimarla.
3. **Comunique la necesidad de estimación.-** Envié una notificación al jefe del proyecto y a los desarrolladores requiriendo la estimación general sobre los escenarios con mayor prioridad.
4. **Divida o vuelva a priorizar los requerimientos (opcional).-** Si el costo del escenario excede el presupuesto de la iteración, determine si se lo puede dividir en etapas, difiera o cancele escenarios que resulten muy costosos.

Como resultado de esa etapa tenemos la lista de escenarios priorizada y de los principales escenarios tenemos las estimaciones.

2.3.7.4.Escribir la Descripción del Escenario

1. **Escoja el personaje apropiado.-** Escoja el personaje que represente mejor al escenario, si muchos personajes utilizan el mismo escenario tome uno de forma arbitraria. Este será el personaje principal del escenario. Si el personaje requiere una funcionalidad diferente agregue un nuevo escenario que cubra esta funcionalidad.
2. **Escriba la Descripción del Escenario.-** Describa cada paso que realiza el personaje mientras se mueve por el sistema tratando de conseguir su objetivo. Retire los pasos que formen parte de escenarios descritos anteriormente y que no son parte de los pasos que diferencian al escenario de otros. Sea específico en los pasos que distinguen a cada escenario.
3. **Divida el Escenario (opcional).-** Si el detalle de estimación es muy extenso como para ser cumplido en una sola iteración. Cada uno de los sub escenario debe mantener todas las características de un escenario normal.

Como resultado de esta etapa tenemos un escenario revisado que captura todo el comportamiento exterior del sistema y las interacciones con los personajes en los diferentes factores.

Como resultado del proceso de creación de escenarios se dispone de la lista de escenarios y su descripción, de la misma forma se debe crear un calendario de desarrollo e implementación, el cual será revisado y publicado en la siguiente iteración.

2.3.8.Creación de un Requerimiento de Servicio de Calidad

2.3.8.1.Lluvia de Ideas Sobre Requerimientos de Servicios de Calidad

1. **Identificación de objetivos de servicios de Calidad.-** Defina la importancia de la calidad de servicio dentro de las siguientes categorías: plataforma, rendimiento, seguridad y capacidad.
2. **Identificación de objetivos de servicios de Calidad.-** Analice los escenarios y aplique los objetivos de cada servicio de calidad en todos los puntos en donde el sistema responda a un personaje.

2.3.8.2.Priorización de los Requerimientos de Servicio de Calidad

En esta etapa se debe asignar una prioridad a cada uno de los requerimientos de servicio para cada uno de los escenarios de la lista. Luego de esto se debe realizar una breve descripción de cada escenario.

2.3.8.3.Redacción de los Requerimientos de Servicios de Calidad

Como paso final de esta etapa se debe redactar cada uno de los requerimientos de servicio de calidad y anexar la información necesaria.

Como resultado de esta etapa tenemos el documento de Especificación de Requerimientos de Calidad.

2.3.9.Métricas utilizadas para la estimación de costos

Microsoft Solution Framework, permite la utilización de cualquier método para calcular el costo del proyecto. Frente al problema de cómo calcular el costo del proyecto se puede utilizar el método de Cálculo de Puntos de Función.

Un punto de función se define como una función del cliente. De tal forma que en un sistema con X puntos de función, el usuario final tendrá a su disposición X funcionalidades del sistema. Para realizar un mejor trabajo se debe interactuar con el usuario y este proceso requiere de dos etapas:

1. Identificación de funciones.
2. Ajustar los puntos de función de a cuerdo al ambiente.

2.3.9.1.Redacción de los Requerimientos de Servicios de Calidad

Identificar los datos de salida de forma única que es generado como resultado del sistema. Debe incluir los reportes y servicios disponibles para otras aplicaciones, los mismos que deben ser agrupados en una de las siguientes 5 categorías:

1. Salidas
2. Consultas
3. Entradas
4. Ficheros
5. Interfaces

2.3.9.1.1. Salidas

Para considerar única a una salida la misma debe cumplir con los siguientes requisitos:

1. El formato es diferente,
2. O, el resultado obtenido es producto de una lógica diferente.

Para asignar valores a cada salida se debe utilizar la siguiente tabla:

	1-5 items de datos referenciados	6-19 items de datos referenciados	20 o más items de datos referenciados
0 o 1 fichero referenciado	Simple (4)	Simple (4)	Medio (5)
2 o 3 ficheros referenciados	Simple (4)	Medio (5)	Complejo (7)
4 o más ficheros referenciados	Medio (5)	Complejo (7)	Complejo (7)

Tabla 3 Costos Valores para las salidas.

2.3.9.1.2. Consultas

Cualquier tipo de respuesta que requiera la interacción del aplicativo y la base de datos, para que una consulta sea única se debe considerar:

1. El formato de entrada o salida es diferente.
2. La lógica que interviene para obtener el resultado es diferente a la de otra consulta, aunque tenga el mismo formato de entrada o salida.

Parte Salida	1-5 items de datos referenciados	6-19 items de datos referenciados	20 o más items de datos referenciados
0 o 1 fichero Referenciado	Simple (4)	Simple (4)	Medio (5)
2 o 3 ficheros Referenciados	Simple (4)	Medio (5)	Complejo (7)
4 o más ficheros Referenciados	Medio (5)	Complejo (7)	Complejo (7)

Parte Entrada	1-4 items de datos referenciados	5-15 items de datos referenciados	16 o más items de datos referenciados
0 o 1 fichero Referenciado	Simple (3)	Simple (3)	Medio (4)
2 ficheros Referenciados	Simple (3)	Medio (4)	Complejo (6)
3 o más ficheros Referenciados	Medio (4)	Complejo (6)	Complejo (6)

Tabla 4 Costos - Consultas

2.3.9.1.3. Entradas

Se considera entradas a cada dato único que ingresa el usuario o proviene de alguna otra aplicación, el mismo que realiza actualiza la información del sistema. La entrada se considera única si:

1. Si el formato es único.
2. O, si requiere de una lógica diferente al ser registrado.

	1-4 items de datos referenciados	5-15 items de datos referenciados	16 o más items de datos referenciados
0 o 1 fichero Referenciado	Simple (3)	Simple (3)	Medio (4)
2 ficheros Referenciados	Simple (3)	Medio (4)	Complejo (6)
3 o más ficheros referenciados	Medio (4)	Complejo (6)	Complejo (6)

Tabla 5 Costos - Entradas

2.3.9.1.4. Ficheros

Los ficheros son el conjunto de datos de que administra la aplicación, los cuales pueden estar almacenados de distintas formas, ya sean archivos de texto, en bases de datos, registros, entre los principales.

	1-19 items de datos Referenciados	20-50 items de datos referenciados	51 o más items de datos referenciados
1 formato/relación de registro lógico	Simple (7)	Simple (7)	Medio (10)
2-5 formatos/relaciones de registro lógico	Simple (7)	Medio (10)	Complejo (15)
6 o más formatos/ relaciones de registro lógico	Medio (10)	Complejo (15)	Complejo (15)

Tabla 6 Costos - Ficheros

2.3.9.1.5. Interfaces

Conjuntos de datos que son enviados desde el aplicativo y luego utilizados por otros sistemas. Una interfaz es el resultado de una de las siguientes operaciones:

1. Datos o información de control se pasa del fichero A al fichero B. En A se puntúa fichero e interfase y en B sólo interfase
2. Datos o información de control se pasa del fichero B a A. En B se puntúa fichero e interfase y en A sólo interfase.
3. Datos o información de control se comparte entre A y B. A y B reciben puntos de fichero e interfase.

	1-19 items de datos referenciados	20-50 items de datos referenciados	51 o más items de datos referenciados
1 formato/relación de registro lógico	Simple (5)	Simple (5)	Medio (7)
2-5 formatos/relaciones de registro lógico	Simple (5)	Medio (7)	Complejo (10)
6 o más formatos/ relaciones de registro lógico	Medio (7)	Complejo (10)	Complejo (10)

Tabla 7 Costos - Interfaces

Luego de obtener todos los puntos de función no ajustados tenemos que calcular los puntos de función ajustados tomando en cuenta los 14 elementos que dependen del ambiente, estos son los siguientes:

1. Comunicaciones de datos
2. Datos o procesamiento distribuidos
3. Objetivos de rendimiento
4. Configuración utilizada masivamente
5. Tasa de transacción
6. Entrada de datos on-line
7. Eficiencia para el usuario
8. Actualización on-line
9. Procesamiento complejo

10. Reutilización
11. Facilidad de instalación y conversión
12. Facilidad de operación
13. Puestos múltiples
14. Facilidad de cambio.

Todos estos factores se califican de 0-5 de a cuerdo a la siguiente tabla:

Valor del Factor	Influencia en el Sistema	Porcentaje que afecta o es requerido por la aplicación
0	Ninguna	0%
1	Insignificante	1-20%
2	Moderada	21-40%
3	Media	41-60%
4	Significativa	61-80%
5	Fuerte	81-100%

Tabla 8 Costos Influencias del Sistema

Finalmente para obtener el costo del proyecto utilizamos la siguiente fórmula PF's no ajustados * (0'65 + 0.01 (influencia 14 factores)).

2.4. Diseñando un proyecto de desarrollo de software

2.4.1.¿Qué significa la planificación de un proyecto de desarrollo de software?

Una vez que se ha entendido por completo lo que se quiere del aplicativo (etapa de planeación), el siguiente paso es planificar y diagramar. La planificación de un proyecto de software se refiere a todo proceso que nos guiará hasta plasmar la especificación funcional de la solución.

Dependiendo del nivel de dificultad de la solución serán distintos los entregables a generar en esta etapa. Dado que MSF versión Agile es para aplicaciones pequeñas y medianas, en esta fase se deben cumplir los siguientes objetivos:

1. Diagramas de casos de uso
2. Definición de la arquitectura del proyecto
3. Diseño del modelo de base de datos
4. Diseño de las interfaces gráficas del sistema

2.4.2.Diagramas de Casos de Uso

Diagramas UML, que identifican la relación los diferentes actores que interactúan con la aplicación y los procesos que esta realiza de acuerdo a los requerimientos funcionales.

2.4.3.Definición de la Arquitectura del Proyecto

El proceso de definición de la arquitectura tiene los siguientes pasos a seguir:

1. Particionar al sistema, luego de conocer los requerimientos, seleccionar los mejores patrones para solucionar los problemas de seguridad, transporte, escalabilidad, entre otros. Validar las capas a ser implementadas dependiendo de la complejidad del sistema. Resolver los problemas de desarrollo tales como:

- Es mejor tener una aplicación para una base de datos específica o desarrollar la aplicación para cualquier base de datos.
- Es necesario crear una capa de acceso a datos separada de la lógica del negocio.
- Se necesita implementar interfaces entre las distintas capas.
- Otro.

2. Determinar las interfaces entre capas, en el caso en el cual el aplicativo lo necesite hay que definir las interfases que se van a desarrollar entre la capa de vista, de negocio, de acceso a datos, de entidades del negocio, y demás. Así mismo hay que definir el transporte a ser implementado en estas interfases, dependiendo de distintos factores de hardware o software se tendrá que elegir entre inproc1, com2, webservice3, remoting4 u otra tecnología.
3. Identificar las vulnerabilidades, es decir los problemas de seguridad que podría tener el aplicativo dependiendo el contexto donde va a ser implantado el sistema.

---

1 Inproc.- Consiste en la utilización de referencias a librerías locales dentro de una aplicación.
2 Com.- **Component Object Model** (**COM**) es una plataforma de Microsoft para componentes de software introducida por dicha empresa en 1993.
3 Webservice.- Un **servicio Web** (en inglés *Web service*) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.
4 Remoting.- La finalidad de remoting es proveer un marco que permita a las aplicaciones interactuar con otras más allá de sus dominios, con lo que es posible invocar métodos y pasar objetos entre ellas

4. Identificar los problemas de rendimiento que se pueden suscitar haciendo pruebas de carga dependiendo de la cantidad de usuarios estimados con aplicaciones pequeñas utilizando la arquitectura propuesta.
5. Implementar la versión final de la arquitectura para el uso por los desarrolladores.

2.4.4. Diseño del modelo de base de datos

Para esto se debe realizar el modelo lógico y luego el modelo de la base de datos, tomando en cuenta los requerimientos levantados en la fase anterior.

2.4.5. Diseño de las interfases gráficas del sistema

A partir de toda la información levantada hasta el momento, se diseñan las interfases del sistema, es decir presentar como el sistema va a terminar, se debe explicar textualmente la funcionalidad del mismo, de esta forma el usuario podrá emitir su criterio sobre el sistema en interfaces gráficas donde el coste de cambios es sumamente inferior que cambios en el sistema ya desarrollado.

2.5. Desarrollando un proyecto de desarrollo de software

2.5.1. Planificación de una Iteración

1. **Determinación del alcance de la iteración.-** Trabaje desde la fecha de entrega hasta la fecha actual, revise el tamaño del escenario, examine el costo de la iteración y revise la complejidad de la iteración.

2. **Estimación del escenario.-** Debe identificar un sub grupo de la lista de escenarios, crear las estimaciones para implementación de los mismos e identificar las necesidades especiales de los mismos.
3. **Estimación de los requerimientos de servicio.-** Identificar un grupo de requerimientos de servicios, de los cuales se deben crear las estimaciones de implementación y los factores clave de cada uno.
4. **Cronograma de trabajo para el escenario.-** Se creará un plan inicial para la iteración, y se procederá a reunirse con el equipo de desarrollo para discutir la validez del mismo.
5. **Cronograma de los requerimientos de servicio.-** De la misma forma se creará un plan inicial de iteración el cual deberá ser revisado.
6. **Cronograma para la corrección de errores.-** Se determinará un cronograma inicial para la corrección de errores, el mismo que deberá ser revisado por el grupo de desarrollo y el equipo de pruebas.
7. **División del escenario y los requerimientos de servicio en tareas.-** Se reunirá al equipo y se creará la arquitectura del sistema, se reasignara el escenario, se crearan las distintas tareas que serán desarrolladas por el equipo.

2.5.2. Guía del Proyecto

1. **Revisión de objetivos.-** Debe escoger cual es el nivel mínimo de aceptación, y reportar el nivel en el que se encuentra.
2. **Determinación del progreso.-** Mantenga el cronograma al día, y cree un estimado de viabilidad si es necesario.
3. **Evaluar los resultados de las métricas de evaluación.-** Revise los reportes de pruebas, además debe crear tareas y riesgos de pruebas.

4. **Mitigación de errores.-** Genere la lista de errores a corregir, revise los errores y asigne prioridad a la solución de los mismos.
5. **Identificación de riesgos.-** Identifique los requerimientos de servicio cuyo riesgo sea alto, proceda a examinar los escenarios y analice los problemas de la iteración y priorice los riesgos.

2.5.3. Guía de la Iteración

1. **Monitoreo de la iteración.-** Revise las prioridades de trabajo y resuelva los problemas que se presenten.
2. **Mitigación de riesgos.-** Reduzca en lo posible la probabilidad del riesgo, si se presenta el problema disminuya su impacto y siempre tenga a mano el plan de contingencia.

2.6. Estabilizando un proyecto de desarrollo de software

Durante esta etapa el grupo de pruebas debe encontrar y mitigar todos los errores que se encuentren en el sistema. De esta forma se debe reducir al mínimo los errores y se deberá comunicar a los desarrolladores para que ellos realicen los cambios necesarios.

De igual forma se recomienda colocar a uno o dos usuarios del sistema a trabajar en el mismo para que ellos ayuden en la detección de errores, para de esta forma familiarizar al usuario con el uso del sistema y verificar que el sistema provee todas las funcionalidades requeridas.

2.7. Implementando software

1. **Ejecute el plan de lanzamiento.-** Revise el material, y coordine la entrega del producto.
2. **Valide el lanzamiento.-** Ejecute completamente el sistema y revise el reporte de errores, en el caso de que encuentre nuevos errores envíelos al grupo de pruebas para que los documenten y resuelvan.
3. **Cree una nota de lanzamiento.-** Documente las limitaciones del producto y revise las notas de la versión.
4. **Implemente el producto.-** Empaquete el producto para su lanzamiento o bien instálelo.

3. CAPÍTULO III - INTRODUCCIÓN A LAS ARQUITECTURAS ORIENTADAS A OBJETOS EN .NET

3.1. Arquitecturas orientada a objetos en .NET

Una arquitectura orientada a objetos, es aquella que utiliza a los objetos en forma de Entidades del Negocio para transmitir los datos entre las distintas capas, y como fuente de datos en los componentes de la capa de vista y capa de negocio, para migrar del modelo de objetos al modelos entidad relación mediante los componentes lógicos de acceso a datos.

Para el caso de estudio de esta tesis se omitirá la implementación de interfaces de servicios internos o externos, y de los flujos de trabajo empresariales ya que estos conceptos van directamente relaciones con aplicaciones muy complejas para las cuales MSF versión 4.0 versión Agile queda relegada.

Sin embargo se presentará una pequeña descripción de estos componentes ya que es importante conocerlos al menos en concepto.

Interfaces de servicio Internos o Externos son componentes que sirven exclusivamente para juntar capas a través de un servicio de comunicación. Estas interfaces no exponen funcionalidades adicionales, sino que implementan las llamadas a la lógica del negocio generalmente, desde la vista en el caso de interfaces internas o desde otro aplicativo en el caso de interfaces externas. La principal diferencia entre las interfaces internas y externas, es que las interfaces externas deben exponer métodos que tengan

tipos de datos genéricos para ingreso y salida, esto dado que interactúan con otros aplicativos, los cuales pueden estar incluso en otro lenguaje no compatible como JAVA, esto se soluciona exponiendo datos en formatos genéricos como XML apoyados con un diccionario.

Flujos de trabajo empresariales**:** Cuando la lógica de negocio es sumamente compleja podemos ayudarnos de clases que representen a los flujos de datos empresariales, en las cuales se agrupan varios métodos de distintas clases de lógica de negocio para poder resolver el problema en cuestión.

La arquitectura que se diseñará será el pilar fundamental de la aplicación. En ella se definirán los medios para acceder a los orígenes de datos, cómo se manejará la lógica de negocio, cómo estará estructurados los componentes de presentación, entre otras.

En este capítulo se expondrá y explicará una arquitectura sencilla y práctica que ayuda a resolver la mayor cantidad de problemas de aplicaciones medianas y pequeñas.

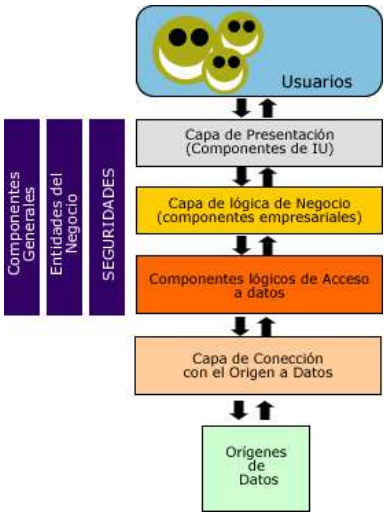


Figura 4. Arquitectura Orientada a Objetos .Net

En la Figura 4 se puede observar seis tipos de componentes indispensables en cualquier arquitectura basada en objetos, los cuales se resumen a continuación:

Capa de Presentación: Comprende el conjunto de componentes de Interfases de Usuario, que sirven como interfaz entre el aplicativo y el usuario. En el caso de .net, se dispone de algunos formatos para la presentación del aplicativo, tales como Windows Forms, Web Forms, o Dispositivos Móviles, separar la lógica de negocio, permite facilitar la creación de aplicaciones independientes de la vista.

Capa de Lógica de Negocio: Es el conjunto de componentes empresariales donde se almacenarán todas las reglas propias del negocio que maneja el aplicativo.

Capa de Componentes lógicos de Acceso a Datos: Dado que hablamos de una arquitectura basada en objetos, esta capa comprende el conjunto de componentes donde, se realiza el mapeo de los mismos desde el modelo entidad relación al modelo de objetos.

Capa de conexión con el origen de datos: En esta capa se utilizarán las mejores prácticas para conectar el origen de datos, sea este específico, es decir una aplicación que funcione exclusivamente con Sql Server por ejemplo, o genéricos.

Entidades del Negocio: Dado que se desea transportar datos desde la capa de vista hasta el almacén de datos, se necesita un medio para transmitir estos datos, dado que es una arquitectura basada en objetos, los datos se transmitirán en modo de objetos que representan a entidades del negocio, los cuales están agrupados en esta capa.

Seguridades: Son componentes de autorización y autenticación que deberán estar disponibles en cualquier parte del aplicativo.

Componentes Generales: Son componentes de distintos tipos, que pueden ir desde validaciones generales de tipos de datos, recursos matemáticos adicionales, mecanismos de administración operativa, tales como excepciones entre otros.

3.2. Objetivos de la Arquitectura

El desarrollo de una buena arquitectura persigue en general cuatro fines definidos, estos son que la aplicación sea: segura, estable, escalable y de fácil mantenimiento.

Seguridad: Una aplicación es segura siempre y cuando no se permita el acceso a los datos sensibles de la misma a usuarios ajenos al aplicativo, o en su defecto a usuarios que a pesar de tener acceso al sistema no tienen permisos suficientes para acceder a esos datos sensibles.

Estabilidad: Se puede considerar una aplicación con arquitectura estable, cuando soporta el máximo nivel de carga transaccional y número de usuarios, en el contexto de producción sin que esto interfiera en su rendimiento normal.

Escalabilidad: Una aplicación es escalable, cuando está lo suficientemente organizada para poder incrementar sus funcionalidades, sin que esto interfiera con las funcionalidades ya realizadas con la reutilización de la mayor cantidad de funcionalidad posible.

Fácil Mantenimiento: Una aplicación es de fácil mantenimiento, cuando un ente distinto a quien se encargo de la realización de la misma, puede, de una manera sencilla dar mantenimiento a la misma.

3.3. Capa de conexión a la base de datos.

La capa de conexión a la base de datos debe tener las mejores prácticas para la interacción de la aplicación con el acceso a datos. Lo primero es establecer la manera en la cual se almacenarán las sentencias que interactuarán con el origen. Por ejemplo, si se utiliza como origen de datos una base de datos, Sql Server, se utilizará al lenguaje SQL como medio de interacción con la base, ahora hay dos modos para interactuar con la base, a través de stored procedures o de non queries, ambas tienen sus ventajas y desventajas.

En el caso de los stored procedures, estos son más rápidos, dado que la base de datos no necesita compilar en tiempo de ejecución las sentencias, ya que al momento de crear los stored procedures estos quedan grabados y compilados en la base.

Sin embargo a menos que se utilice Stored Procedures 100% ANSI estos van a quedar siendo dependientes de la base de datos, lo cual es muy perjudicial en aplicaciones donde la base puede cambiar o se desee instalar en varias instituciones con distintas bases.

Los non queries por su lado, son más pesados, lo cual en aplicaciones con carga grande puede resultar un problema o requerir una mayor inversión en

hardware, sin embargo es mucho más sencillo realizar una aplicación independiente de la base de datos usando este mecanismo.

En la capa de conexión a acceso a datos, se debe tomar en cuenta otro factor que es muy importante, este es las clases que se eligen para conectarnos con la aplicación. En el caso de .net, debemos elegir el provider, en el caso de conocer que la base va a ser siempre Sql 2000 por ejemplo, el provider podría ser un SQL Provider, sin embargo si es un motor genérico, podríamos usar un OLEDB provider o un ODBC provider. Así mismo se debe saber si se va a usar el modelo de objetos ADO.Net, el cual contiene mejoras considerables en rendimiento y funcionalidad que el anterior modelo basado en recordsets, que sin embargo podría ser recomendado en un equipo de trabajo que nunca ha trabajado con ADO.net y donde no existe el suficiente tiempo para capacitación.

3.4. Capa de componentes lógicos de acceso a datos, DALC, mapeando el modelo de objetos a modelo entidad relación.

3.4.1. Función de los DALC.

En la vida real se tiene algunos formas para almacenar los datos, pudiendo ser estos almacenes bases de datos relaciones, archivos de texto, entre otros. El problema está en que sin importar el tipo de almacén que vayamos a utilizar, se requiere de un método sencillo y práctico para recuperar los datos en el aplicativo.

Dado que se desea trabajar sobre arquitecturas orientadas a objetos, se deberá recuperar los datos en Entidades del Negocio, de las cuales trata con mas detalle a lo largo de este capítulo, pero por lo pronto basta saber que son la representación mediante clases de un ente específico dentro del aplicativo, por ejemplo el ente Estudiante, o el ente Curso, que se crean a partir de datos traídos del almacén de datos.

Estos entes, se representan en forma de clases dentro de .net, las cuales pueden ser implementadas de distintas formas, tales como arreglos puros de objetos, colecciones, datasets tipificados, entre otros. De la misma manera la forma en la cual se implemente estos entes dependerá de factores como velocidad, empaquetamiento, facilidad, entre otras.

Ahora bien, sin importar la manera que se decida para conectarse a la base de datos, la capa de conexión al menos debe exponer un método para traer datos en un contenedor genérico, como por ejemplo un DataSet, y otro método para ejecutar un non-query o en su defecto un procedimiento almacenado. De esta forma se puede ejecutar las cuatro sentencias básicas de insert, update, delete y select.

Entonces, al momento de comenzar a desarrollar los DALC, se dispone por un lado un componente genérico que permite interactuar con el almacén de datos, y un conjunto de clases que representan todos los entes que van a existir en la aplicación.

La pregunta es ahora, de qué manera se van a migrar los datos desde un contenedor genérico como el dataset a un contenedor específico que representa un ente como un dataset tipificado.

Es en las clases donde se realiza esta acción, por lo general se crea un DALC para cada ente, el DALC relacionado con el ente se encargará de exponer al menos cuatro métodos, los cuales corresponden a la inserción, borrado, actualizado y selección de registros. En el caso de la selección es obligatorio que el método me devuelva un arreglo de entes en el formato específico de las entidades del negocio.

En el caso del borrado, la inserción y la actualización, dependerá de la forma que se elija para trabajar para ver si es preferible utilizar una entidad del negocio para recoger los datos para realizar esta actividad, o es suficiente con enviar por ejemplo la llave primaria en forma de entero para el caso del borrado de un registro.

Por lo general se utiliza datos simples que representan cada columna del registro a eliminar, insertar o actualizar, ya que esto evita un doble mapeo, además que los componentes de la vista en el caso del .net 2005 se adecuan mejor.

3.4.2. Acerca del acceso de los componentes lógicos de acceso a datos desde la vista.

Hay situaciones en las cuales es más fácil implementar alguna funcionalidad omitiendo la capa de negocio para ir directamente a los conectores lógicos de acceso a datos. En caso de realizar esto, se debe tomar en cuenta las siguientes consideraciones:

- La arquitectura se vuelve más compleja dado que a veces se llama a los DALC, y otras a los componentes de negocio

desde la vista, lo cual puede dificultar la escalabilidad y el fácil mantenimiento.

- Puede incrementar en el futuro lógica de negocio no contemplado, en cuyo caso tocaría crear los componentes del negocio y cambiar en la vista el direccionamiento ya no al componente de acceso a datos sino al del negocio.

3.5. Capa de lógica de negocio.

La capa de lógica de negocio es aquella que encapsulará las funcionalidades específicas de la aplicación. Es la raíz de las transacciones atómicas, expondrá métodos, sea que estos devuelvan datos simples o complejos que interactuarán directamente con la vista. Es independiente por completo del almacén de datos y de los componentes de presentación, y los métodos de esta capa deben estar agrupado en clases de forma coherente desde el punto de vista funcional.

Dado que uno de los objetivos principales es que la arquitectura sea escalable, hay que prestar la mayor atención posible a la lógica de negocio para que exponga métodos lo más útiles y genéricos posibles, de forma que la lógica que aumente, pueda usar los métodos Preexistentes para resolver problemas más complejos. Es decir, que es una buena práctica crear métodos sencillos, para agruparlos según la funcionalidad específica que se requiera.

Es importante que a lo largo del proyecto se mantenga un conjunto de datos coherente, tanto como información de entrada como de salida a los

métodos. Una buena práctica es usar un arreglo de entidades de negocio como dato de salida y un conjunto de datos básicos que representen las columnas del registro/s a ser tratado como datos de entrada.

Hay que tomar en cuenta la transaccionabilidad de la aplicación en las transacciones atómicas de la aplicación. Si una transacción, corresponde la ejecución de varios métodos atómicos, hay que implementar un mecanismo tal que si una de las transacciones no se ejecuta, todos los datos de las transacciones anteriormente ejecutadas vuelvan a sus estado inicial.

En el caso de esta arquitectura propuesta, los componentes de lógica de negocio son llamados únicamente desde otros componentes de lógica de negocio, o en su defecto desde la vista. El objetivo es que la vista se vuelva lo más simple posible, de forma que, primero sea fácil realizar cambio en los procesos del negocio, sea fácil incrementar funcionalidades, y se simplifique al máximo el cambio de interfaz de usuario o la exposición de algunas funcionalidades mediante otras interfases no definidas en un principio.

Así mismo se recomienda que toda interacción que vaya a tener la lógica de negocio con el almacén de datos, se haga a través de los Data Access Logical Connectors, ya que de esta forma se asegura que se utilicen en la aplicación el mismo mecanismo de conexión y que el código de la

aplicación esté organizado para su fácil mantenimiento y mayor escalabilidad.

3.6. Capa de entidades de negocio para transmitir datos entre capas.

Todo se inicia en el formato que se elija para devolver datos desde los componentes lógicos de acceso a datos hasta la lógica de negocio. Se puede transmitir datos principalmente de dos maneras, la primera es mediante componentes genéricos (lo cual involucra la eliminación de nuestra arquitectura toda posibilidad de ser una arquitectura orientada a objetos) o mediante objetos específicos que representen entidades del negocio (arquitecturas orientadas a objetos).

Las ventajas de las arquitecturas orientadas a objetos son las siguientes:

En la capa de Interfases de Usuario .net 2005 presenta algunas funcionalidades excelentes con el uso de Entidades del Negocio, tales como dibujo automático de formularios, creación de objetos con data binding predefinido, entre otros.

En la capa del negocio, la creación de lógica de negocio se vuelve mucho más sencilla ya que el programador está en capacidad de visualizar el problema y tiene acceso a datos importantes como el nombre de las columnas del registro en todo momento. En general tendremos una lógica de negocio mucho más clara que nos ayudará a tener una solución mucho más fácil de mantener.

La principal desventaja está en el proceso adicional que se debe realizar para el mapeo entre modelo entidad relación y modelo de objetos, el

problema se debe a que cuando recuperamos los datos al aplicativo es obligatorio traerlo en un contenedor genérico como un dataset, luego pasar los datos de este dataset a un arreglo de clases del tipo Cliente por ejemplo, o a otro dataset tipificado que representa la estructura de datos de la entidad Cliente. Sin embargo se pueden hacer funciones o clases genéricas que solucionen este problema.

De tal forma se recomienda siempre el uso de Entidades de Negocio para pasar los datos entre capas. Entre las principales tipos de datos que se pueden utilizar para tal objetivo están las colecciones, los arreglos puros de objetos y los datasets tipificados. En el caso de .net es sumamente recomendable el uso de datasets tipificados por los siguientes motivos:

Evita problemas relacionados con el empaquetamiento.

Se evita codificar las clases ya que son creadas con el proceso de arrastra y suelta desde el servidor.

Los problemas que puede generar por eficacia son ridículos, si se toma en cuenta que se utiliza arquitecturas para aplicaciones pequeñas y medianas.

Este tipo de datos es en extremo compatible con los componentes de la interfaz de Usuario.

Expone funcionalidades genéricas como búsqueda y organización sin necesidad de programarlas.

Al momento de crear las entidades empresariales hay que tomar en cuenta las siguientes consideraciones:

No se trata de crear una entidad empresarial por cada tabla de la base, sino de exponer una entidad empresarial por cada objeto que necesita

interacción. De tal forma la mayoría de tablas que se han creado para romper el problema relacional de varios a varios no deberán ser tomadas en cuenta, y de vez en cuando existirá alguna entidad empresarial que contenga los datos de más de una tabla del modelo relacional.

En el caso de aplicaciones en extremo sencillas es necesario un análisis para ver si es o no necesario la creación de entidades de creación de entidades empresariales, que serán por otro lado un coste adicional en desarrollo.

Las entidades empresariales deben validar que los datos que se les intenta ingresar son datos válidos coherentes con el modelo relacional.

Se debe tomar en cuenta que las entidades empresariales van a interactuar con la capa de vista, de negocio y de componentes lógicos de acceso a datos, por ende deben ser diseñadas de forma que sean lo más sencillo de implementar con estas capas.

3.7. Capa de vista.

La capa de vista comprende todos los componentes de interfaz de usuario que se servirán para que el usuario interactúe con la aplicación. El objetivo es dejar a esta capa lo más sencilla posible, de forma que tenga la menor cantidad de código posible, y que todos los requerimiento que se hagan, se logren simplemente con la comunicación con la capa de negocio.

Normalmente la interfaz de usuario tendrá controles que reaccionen a eventos del usuario, estos eventos no serán sino llamadas a componentes del negocio.

Cuando se necesite que los controles obtengan datos desde el almacén de datos, será la propia capa de negocio quien enviará los datos, en la vista máximo se llamará a las funciones para asociar el binding de datos. En la mayoría de casos, dado que la arquitectura orientada a objetos, este binding se realizará a partir de entidades de negocio.

La capa de vista se encarga principalmente de mostrar, validar y recuperar datos del usuario. Así mismo debe interpretar las acciones que el usuario realiza sobre el sistema para dar los resultados esperados.

En la capa de interfaz de usuario se puede llevar a cabo la transformación de datos de forma que sean comprensibles por la capa de negocio. Por ejemplo, mostrar en un combo los nombres de los clientes pero la capa de negocio necesita el ID, sin embargo, estas acciones también pueden hacerse en el negocio, de una manera más recomendable.

La paginación puede realizarse automáticamente en esta capa, ya que la mayoría de controles la realizan automáticamente, sin embargo hay que tomar en cuenta que esto puede hacer pesada a la aplicación ya que hay el flujo de todos los datos desde el negocio hasta la vista, lo cual puede afectar en el rendimiento.

Esta capa se encarga de adecuar el formato de datos al formato específico de la capa de negocio, tales como el formato de fecha, o separadores numéricos con “.”o “,”.

Las interfaces de usuario más comunes son interfaces de usuario Web, Interfaces de de Usuario Windows e interfaces de Usuario para Dispositivos Móviles.

Las interfaces de usuario Web deben ser sumamente livianas en el caso de querer ser implementadas en el Internet, son por lo general más costosas de realizar, sin embargo se tiene la ventaja que se la instala en el servidor y es muy sencillo acceder a estas desde cualquier máquina de la red local sin otro requerimiento adicional que un navegador compatible.

Las interfaces Windows son las más comúnmente usadas, brindan al usuario una mayor facilidad en su manejo, son menos costosas y por lo general más estables. Tienen el problema de la instalación en cada equipo con el consiguiente validación de versiones, de todas formas esto se puede solucionar con tecnologías tales como ClickOnce.

Las interfaces para dispositivos móviles tales como pocket pc usualmente se realizan para trabajos específicos tales como toma de inventarios, por lo general necesitan interacción adicional con un aplicativo principal que generalmente es Windows. Hay que tomar en cuenta el tamaño de la pantalla por lo cual la optimización de espacio será crucial en este tipo de interfaces de usuario.

3.8. Capa de Seguridades

La capa de seguridades servirá para brindar los mecanismos de autentificación y autorización en la aplicación. Autentificación significa

reconocer al usuario que intenta ingresar al aplicativo como un usuario válido del sistema. Autorización significa reconocer una funcionalidad específica como una función a la cual el usuario previamente reconocido como válido tiene permisos suficientes.

Generalmente si los componentes de acceso a datos y de lógica de negocio no serán expuestos a aplicaciones externas, es suficiente que las validaciones de seguridad se las hagan en la vista, caso contrario se necesitaría de mecanismos más complejos que incluyan el reconocimiento de usuarios y roles por cada método expuesto en la capa donde se desee implementar las seguridades.

Esta capa se encargará de proveer las funcionalidades básicas relacionadas con seguridades tales como validar usuarios, con la correspondiente encriptación y des encriptación en caso de ser necesario, validar roles, ingresar nuevos usuarios y asignar funcionalidades por roles, así como cualquier funcionalidad que se relacione con seguridades del sistema que se crea conveniente.

3.9. Componentes Generales

La capa de componentes generales será entonces la capa que contenga todas las funcionalidades que no estén encapsuladas en el framework que se crean necesarias dentro de la aplicación. Esto es mecanismos para validar datos especiales, fórmulas matemáticas complejas que son necesarias desde todas las capas, manejo de excepciones, manejo de

transaccionalidad, y todas las funcionalidades que se crean deben ser accedidas desde cualquier capa.

3.10. Ejemplo práctico.

En base a la teoría revisada en este capítulo, se va a crear una arquitectura práctica que ejemplifique lo expuesto. El objetivo será crear una aplicación Windows que permita recuperar, ingresar, modificar y eliminar datos desde un origen genérico tomando en cuenta todas las recomendaciones expuestas anteriormente.

El código de ejemplo de esta aplicación se encuentra en el CD adjunto al presente documento, a continuación pasaremos a explicarlo.

Revisión del almacén de datos

Para esta aplicación de Ejemplo, se dispondrá una base de datos con una tabla llamada CE_Operadora, que representa en un aplicativo de manejo de contratos de celulares las posibles operadoras. La tabla es la siguiente, y está creada es SQL Server 2000, aunque para el ejemplo esto da lo mismo porque estamos asumiendo que uno de los requerimientos del aplicativo es que sea independiente de la base de datos:

Figura 5 Entidad de Negocio

Creación de la capa de conexión con la base de datos.

Se creará un proyecto de clases al cual denominaremos LogicalConnector que se encargará de la conexión con la base de datos tal como muestra la figura

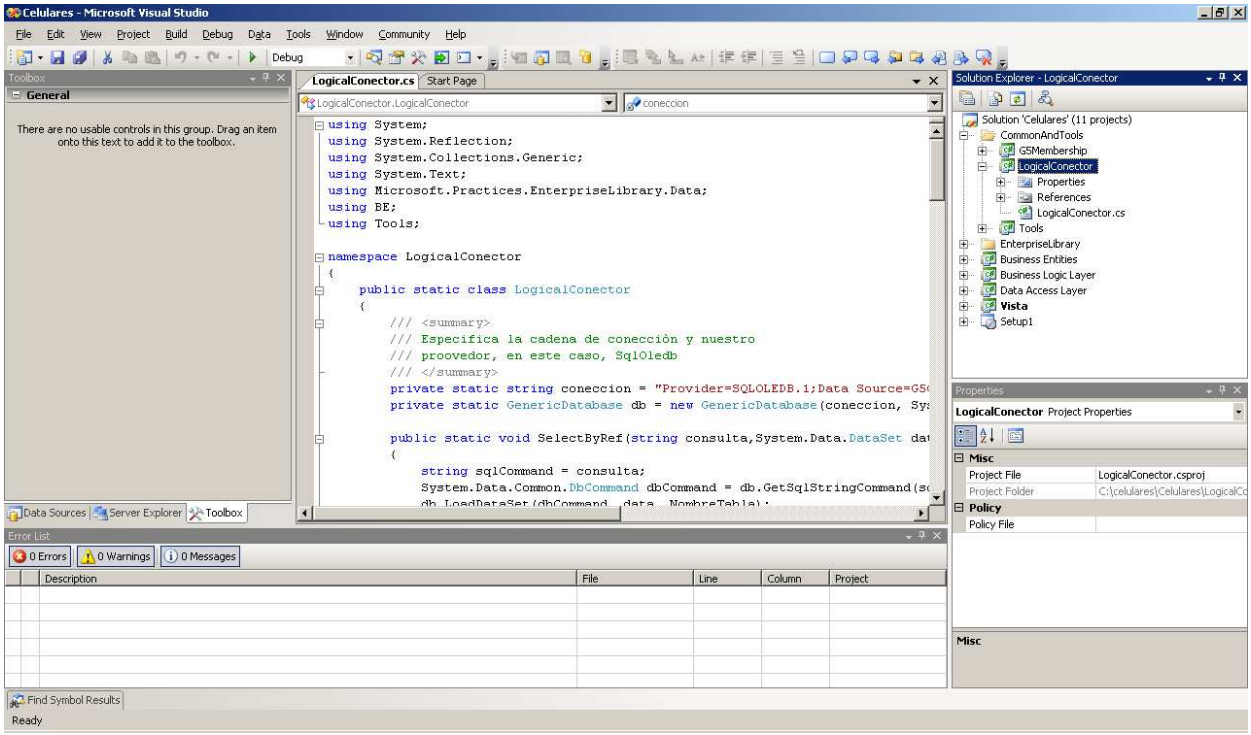


Figura 6 Print Screen del Proyecto del Caso Práctico

Esta clase tendrá las funcionalidades básicas para recuperar datos en un componente genérico, en este caso un Dataset, y la ejecución de non queries para el ingreso, borrado y actualización de datos.

Los métodos son los siguientes:

Método Select

/// <summary>

/// Retorna un dataset a partir de una consulta

/// sql de Select

/// </summary>

```
/// <returns>CursoEntityDataTable</returns>
public static System.Data.DataSet Select(string consulta)
        {
         string sqlCommand = consulta;
    System.Data.Common.DbCommand dbCommand =
db.GetSqlStringCommand(sqlCommand);
      System.Data.DataSet data = new System.Data.DataSet();
      data = db.ExecuteDataSet(dbCommand);
      return data;
                }
```

Este método recibe como parámetro un tipo de datos String que representa la consulta Select a partir de la cual se crea el DataSet que se devuelve, tanto la clase como el método son estáticos de forma que sea la misma clase que se ejecute en toda la aplicación y evitemos abrir demasiadas conexiones simultaneas.

Método Ejecutar SQL

```
    /// <summary>
      /// Ejecuta una sentencia sql
      /// </summary>
      /// <returns>CursoEntityDataTable</returns>
      public static void EjecutarSql(string consulta)
        {
         string sqlCommand = consulta;
```

```
System.Data.Common.DbCommand dbCommand =
db.GetSqlStringCommand(sqlCommand);
        db.ExecuteNonQuery(dbCommand);
      }
```

Este método recibe una consulta sql como parámetro de tipo String, y a partir de aquí la ejecuta.

Los métodos InsertWithParams y UpdateWithParams

Este par de métodos a pesar de no ser obligatorios, son de mucha ayuda en la práctica según nuestra experiencia. Se trata de métodos que ayudarán a crear de forma genérica los estatutos de inserción y actualización, pasando como argumentos el nombre de las columnas de la tabla a insertar o actualizar y los valores que están encapsulados en una clase genérica llamada ParametrosManager que contiene por adentro el arreglo de parámetros necesarios.

```
public static void InsertWithParamas(String
NombreTabla,ParametrosManager args)
      {
      string consulta = "Insert into " + NombreTabla + " (";
      foreach (Parametro val in args.regulares)
          {
          consulta += val.Nombre + ",";
          }
      consulta = consulta.Remove(consulta.Length - 1, 1);
```

```
consulta = consulta + “) values (“;

foreach (Parametro val in args.regulares)

    {

  consulta += “’” + val.Valor + “’,”;



    }

consulta = consulta.Remove(consulta.Length - 1, 1);

consulta = consulta + “)”;

///////ingrese su consulta insert//////////////////


    ////////////////////////////////////////////////


//////esta secciòn no necesita ser modificada////

EjecutarSql(consulta);

    ////////////////////////////////////////////////

 }
```

Creación de la capa de entidades del negocio

Para esto se crea un proyecto de clases, cada entidad del negocio se representará por un dataset tipificado que contiene el esquema requerido. Es importante que concuerden los nombres de las columnas y los tipos de datos con los del modelo relacional, para que la entidad esté en capacidad de validarlos como se expuso en la teoría de este capítulo.

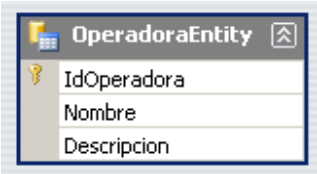


Figura 7 Dataset Tipificado

En la figura se observa el dataset tipificado que representa la entidad de negocio OperadoraEntinty

Creación de la capa de componentes lógicos de acceso a datos.

Esta capa se encargará de migrar los datos desde la el dataset genérico expuesto en la capa de conexión a la base a los datasets tipificados que representan entidades de negocio. En esta capa se expone los métodos necesarios para realizar las acciones de inserción, actualización, eliminación y selección.

Método LlenarParams

Este método recibe un conjunto de datos básicos que representan las columnas del registro a ser actualizado o insertado para retornar un objeto del tipo ParametrosManager el cual es utilizado por la capa de conexión para hacer inserciones y actualizaciones sin necesidad de escribir el sql statement.

```
private static ParametrosManager LlenarParams(int IdOperadora, string
Nombre, string Descripcion)
    {
      ParametrosManager parametro = new ParametrosManager();
    parametro.addRegulares("Nombre",Nombre);
    parametro.addRegulares("Descripcion", Descripcion);
```

```
    parametro.addKeys("IdOperadora",
System.Convert.ToString(IdOperadora));
        return parametro;
                }
```

Método GetOperadoras

Esta función retorna un objeto del tipo OperadoraEntityDataTable, que es el DataTable del Dataset Tipificado explicado en las entidades del negocio, recibe un string que representa el "Where" de la consulta de selección, y hace la migración recorriendo uno a uno todos los registros que retorna el dataset genérico que recuperamos de la capa de conexión y asignándoles al datatable del dataset específico que representa la entidad del negocio, en este datatable se guardarán todos los registros como un arreglo de datarows, cada uno representando un objeto relacionado con un registro.

```
/// <summary>
    /// Retorna todos los AreaOperadoras
    /// </summary>
    /// <returns>AreaOperadoraEntityDataTable</returns>
  public static BE.OperadorasEntity.OperadoraEntityDataTable
GetOperadoras(string where)
      {
    //////////Escribimos la consulta//////////////////
    string consulta = "SELECT * FROM CE_Operadora " + where;
```

```
            ////////////////////////////////////////////////

//////////Declaramos un dataTable llamado entidades////////
//////////que tenga EXACTAMENTE los mismos campos que//////
//////////la consulta escrita anteriormente////////////////
   OperadorasEntity.OperadoraEntityDataTable entidades = new
OperadorasEntity.OperadoraEntityDataTable();
            ////////////////////////////////////////////////////


   ////////////////////ESTE CODIGO NO NECESITA SER
MODIFICADO//////////////////////////
      System.Data.DataSet aux; aux =
      LogicalConector.LogicalConector.Select(consulta); foreach
      (System.Data.DataRow row in aux.Tables[0].Rows)
            {
         System.Data.DataRow row1 = entidades.NewRow();
   foreach (System.Data.DataColumn column in aux.Tables[0].Columns)
               {
         row1[column.ColumnName] = row[column.ColumnName];
               }
         entidades.Rows.Add(row1);
```

```
            }

            return (entidades);
            ////////////////////////////////////////////////////////////////////////////
        }
```

Métodos Insert y Update

Estos métodos se encargan de realizar inserciones y actualizaciones sobre la tabla Operadoras del modelo relacional. No todas las tablas tendrán una clase de conexión lógica de acceso a datos, no la tendrán especialmente aquellas que sirvan para romper los varios a varios, aunque esto depende mucho de la organización del equipo de desarrollo. A continuación exponemos los métodos para realizar las inserciones y actualizaciones genéricas:

Método Insert

```
/// <summary>
    /// Inserta un AreaOperadora a partir de paràmetros
    /// </summary>
    /// <returns>void</returns>
    public static void Insert(string Nombre, string Descripcion)
      {
LogicalConector.LogicalConector.InsertWithParamas("CE_Operadora",
LlenarParams(0, Nombre, Descripcion));
                    }
```

Método Update

```
/// <summary>
/// Actualiza un AreaOperadora a partir del Id
/// </summary>
/// <returns>void</returns>
   public static void Update(int IdOperadora, string Nombre, string
Descripcion)
      {
LogicalConector.LogicalConector.UpdateWithParamas("CE_Operadora",
LlenarParams(IdOperadora,Nombre,Descripcion));
                        }
```

Método Delete

El método delete ejecutará un Non Query a través de la capa de conexión a la base de datos, recibe como parámetro un entero que representa el Id del registro a eliminar y a continuación lo elimina.

```
/// <summary>
/// Borra una Operadora a partir del Id
/// </summary>
/// <returns>void</returns>
public static void Delete(int IdOperadora)
      {
       ///////ingrese su consulta delete//////////////////
```

```
string consulta = "Delete From CE_Operadora where IdOperadora = " +
IdOperadora;
        ////////////////////////////////////////////////

    //////esta secciòn no necesita ser modificada////
    LogicalConector.LogicalConector.EjecutarSql(consulta);
        ////////////////////////////////////////////////

                    }
```

Creación de la capa de lógica de negocio

En esta capa se debe programar todas las reglas del negocio, las diferentes validaciones y acciones que se presentan entre cada uno de los objetos ya creados en la capa de entidades.

La lógica del negocio interactúa con la capa de entidades y con la interfaz del usuario para de esta forma brindar seguridad y escalabilidad sin tener que alterar en lo posible la interfaz de usuario si una de las reglas del negocio cambia.

Cada uno de los métodos creados en esta capa puede retornar estructuras de datos simples o complejas dependiendo del resultado de la operación que se ha efectuado.

Además los métodos de la capa de negocio interactúan entre ellos, hay que exponer en cada método funcionalidades sencillas de forma que las funcionalidades complejas puedan reutilizar lo métodos creados

anteriormente. Aquí se expondrán todas las funciones que se necesite desde la interfaz de usuario.

En la clase de ejemplo se expondrán cinco métodos, además de los cuatro métodos expuestos en los conectores lógicos de acceso a datos, tendremos uno adicional que recupera un registro dependiendo del Id.

Método SelectAll

Llama al método GetOperadoras de la clase de conexión lógica de acceso a datos, de esta forma se retorna un arreglo de entidades de negocio en forma de un OperadoraEntityDataTable que es la tabla contenida en nuestro Dataset Tipificado.

```
/// <summary>
        /// Retorna todos los Operadoras
        /// Compatibilidad con GridView y DetailsView
        /// </summary>
        /// <returns>OperadoraEntityDataTable</returns>
[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Select, true)]
  public static BE.OperadorasEntity.OperadoraEntityDataTable SelectAll()
    {
     return DAL.OperadoraDalc.GetOperadoras("");

    }
```

Método SelectById

Realiza la misma acción que el método SelectAll, pero este envia como parámetro el "where"de la consulta Select para que se retorne únicamente los registros cuyo Id coincidan con el enviado como parámetro.

```
/// <summary>
        /// Retorna todos los Operadoras dependiendo del Id
        /// Compatibilidad con GridView y DetailsView
        /// </summary>
        /// <returns>OperadoraEntityDataTable</returns>
[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Select, true)]
  public static BE.OperadorasEntity.OperadoraEntityDataTable SelectById(int IdOperadora)
      {
  return DAL.OperadoraDalc.GetOperadoras("where IdOperadora=" + System.Convert.ToString(IdOperadora));
      }
```

Métodos Insert, Update y Delete

Son llamadas a los métodos del mismo nombre de los DALC, los cuales a su vez se encargarán de llamar a la capa de conexión a la base de datos para realizar las transaciones debidas.

```
/// <summary>
    /// Inserta un Operadora a partir de los paràmetros
    /// Compatibilidad con GridView y DetailsView
    /// </summary>
    /// <returns>void</returns>
[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Insert, true)]
    public static void Insert(string Nombre, string Descripcion)
     {

      DAL.OperadoraDalc.Insert(Nombre, Descripcion);
     }

    /// <summary>
    /// Borra un Operadora a partir del id
    /// Compatibilidad con GridView y DetailsView
    /// </summary>
    /// <returns>void</returns>
```

```
[System.ComponentModel.DataObjectMethodAttribute(System.ComponentM
odel.DataObjectMethodType.Delete, true)]
    public static void Delete(int IdOperadora)
     {
      DAL.OperadoraDalc.Delete(IdOperadora);
     }

    /// <summary>
    /// Actualiza un Operadora a partir de los paràmetros
    /// Compatibilidad con GridView y DetailsView
    /// </summary>
    /// <returns>void</returns>
[System.ComponentModel.DataObjectMethodAttribute(System.ComponentM
odel.DataObjectMethodType.Update, true)]
  public static void Update(int IdOperadora,string Nombre, string
Descripcion)
     {
      DAL.OperadoraDalc.Update(IdOperadora, Nombre, Descripcion);
     }
```

Creación de la capa de interfaz de usuario

Esta es la capa que interactúa con el usuario, en esta capa se debe tomar en cuenta todos los aspectos de navegación, facilidad de uso, seguridad de acceso y presentación del sistema.

La capa de interfaz de usuario trabaja directamente con la capa de lógica del negocio y las entidades; para, realizar todas las tareas que tiene como objetivo el sistema.

Esta capa puede ser desarrollada en varios ambientes, ya sea orientada a una aplicación de escritorio, una página Web, un servicio, o una mezcla de ellas dependiendo de los requerimientos del cliente y el de los usuarios. Gran parte de la acogida del sistema depende de la interfaz por lo que se recomienda capacitar a los usuarios en su utilización.

En el desarrollo de esta capa pueden intervenir diseñadores gráficos para mejorar el aspecto del sistema, de la misma forma los usuarios pueden apoyar con ideas de cómo facilitar el uso del mismo.

Como primer paso se debe crear un formulario genérico que contenga las funcionalidades básicas de navegabilidad, y exponga métodos virtuales. Los formularios que hagan funcionalidades comunes de administración heredarán de este, e implementarán las funciones virtuales de acuerdo a los parámetros específicos que se implemente en cada uno.

La idea es crear un formulario que tenga dos vistas, en la una se mostrará todos los registros, y se podrá eliminar a los registros, esta vista tendrá un DataGrid, y un botón para eliminar y otro para Insertar. Sin embargo en el formulario genérico no se dibujará el grid, ya que este será diferente para cada formulario específico.

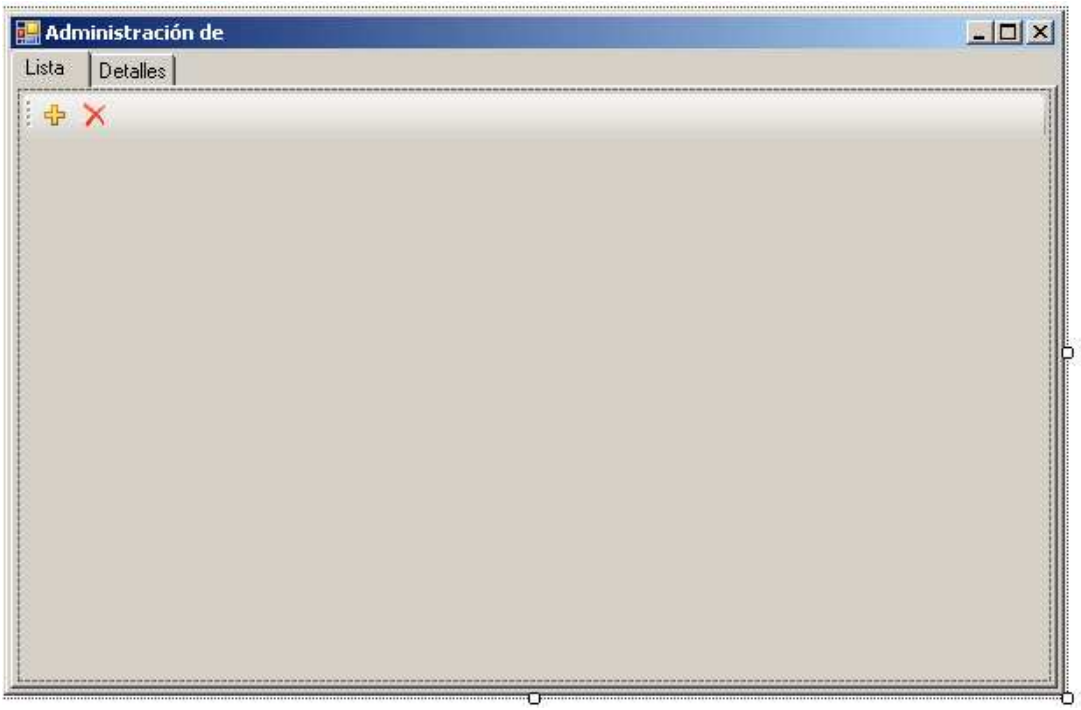


Figura 8. Caso Práctico, Formulario Genérico

Figura 9 Formulario Genérico, Detalles

El botón de insertar, y el doble clic sobre el DataGrid (cuando lo haya), nos enviará a la segunda vista del formulario, donde existirán textBox, combos, y demás controles para representar un único registro. En el caso de haber hecho clic sobre el botón para añadir, esos datos representarán el registro a ser ingresado, en el caso de haber dado doble clic en el grid, estos campos representarán los datos del registro a ser actualizado y tendrán por su valor por defecto los datos del registro a ser actualizado.

A continuación se detallan los métodos que expone este formulario genérico.

El formulario deberá mostrar u ocultar controles, llenarlos o vaciarlos dependiendo si está en estado para mostrar todos los registros en el grid, si va a actualizar un registro o si va a insertar uno nuevo. Para esto se han creado los métodos ModeView() para cuando muestra los registros en el Gris, ModeInsert() para cuando va a insertar un nuevo registro

y ModeUpdate() para cuando va a actualizar el registro. Esta práctica es recomendable seguirla, sea cual sea la vista que se use.

```
protected void ModeUpdate()
    {
    tabControl1.TabPages.Add(tabDetalle);
    tabControl1.TabPages.Remove(tabLista);
    cmdInsertar.Visible = false;
    cmdActualizar.Visible = true;
    }
protected void ModeView()
    {
    errorProvider1.Clear();
    tabControl1.TabPages.Add(tabLista);
    tabControl1.TabPages.Remove(tabDetalle);
    ActualizarGrid();
    }

protected void ModeInsert()
    {
    tabControl1.TabPages.Add(tabDetalle);
    tabControl1.TabPages.Remove(tabLista);
    cmdInsertar.Visible = true;
    cmdActualizar.Visible = false;
```

```
foreach (Control control in tabControl1.TabPages[0].Controls)
        {
      if(control.ToString().Contains("TextBox"))
      control.Text = "";
        }
      PrepararControlesInsert();
    }
```

Método Borrar

El método Borrar que verifica todo el proceso necesario antes de eliminar un registro del grid, es decir comprueba que exista un registro seleccionado en el grid para eliminarlo y verifica que el usuario verdaderamente desee eliminar el registro.

```
protected bool borrar(DataGridView grid)
    {
      if (grid.SelectedRows.Count > 0)
        {
  if (MessageBox.Show("Seguro desea eliminar el registro seleccionado",
"Alerta", MessageBoxButtons.YesNo) == DialogResult.Yes)
          {
      return true;
          }
```

```
        }
        else
        {
    MessageBox.Show("Para eliminar un registro debe seleccionarlo del grid",
"Error");
        return false;
        }

        return false;
                }
```

Método grd_DobleClick

El método grd_DobleClick pone al formulario en estado para que se pueda actualizar el registro sobre el cual se realizó el doble clic

```
protected void grd_DobleClick(DataGridView grid)
      {
        try
        {
        if (grid.SelectedRows.Count > 0)
            {
    codigo = System.Convert.ToInt32(grid.SelectedRows[0].Cells[0].Value);
        ModeUpdate();
            }
```

```
        else
            {
   MessageBox.Show("Para actualizar un registro debe seleccionar todo el
registro del grid", "Error");
            }
          }
        catch { }
                    }
```

El botón cancelar cuando el formulario está en modo de inserción o en modo de actualización funciona de manera idéntica:

```
private void cmdCancelar_Click(object sender, EventArgs e)
       {
        if (tabControl1.TabPages.Contains(tabDetalle))
        ModeView();
        else
        this.Close();
       }
```

Y además tiene la definición de los siguientes métodos virtuales a ser implementados en el formulario a heredar:

Método ActualizarGrid()

Implementa las funcionalidades para dar databinding al grid

```
        protected virtual void ActualizarGrid()
        {
         // Aqui va:
   //[nombre del bindingSource].DataSource = BLL.[nombre].SelectAll();
   // Ejemplo: aplicacionEntityBindingSource.DataSource =
BLL.AplicacionBlc.SelectAll();
        }
```

Método toolBorrar_Click

Implementa la funcionalidad para cuando el botón eliminar es ejecutado

```
    //botón eliminar
   protected virtual void toolBorrar_Click(object sender, EventArgs e)
      {
        // Aqui va:
        //borrar([nombre_del_DataGridView]);
        // Ejemplo:
        /*if (borrar(grdAplicaciones))
          {
BLL.[nombreDelBLC.Delete(System.Convert.ToInt32([nombreDelGrid].Sele
ctedRows[0].Cells[0].Value));
    ActualizarGrid();
```

```
        }*/
    }
```

Método grid_DoubleClick

Implementa la funcionalidad para cuando se hace doble click sobre el Grid.

```
protected virtual void grid_DoubleClick(object sender, EventArgs e)
    {
     // Aqui va:
     //grd_DobleClick([nombre_del_DataGridView]);
     // Ejemplo: grd_DobleClick(grdAplicaciones);
     //Asignar el evento DoubleClick del grid a esta funcion
    }
```

Implementación del Formulario Específico

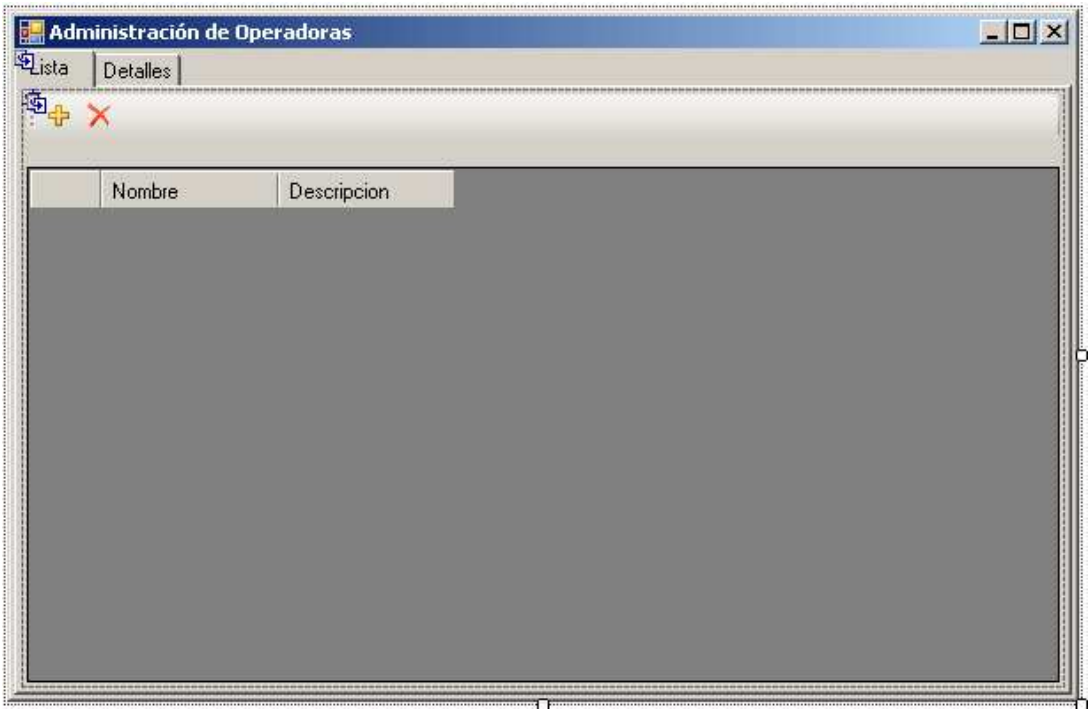


Figura 10 Ejemplo Funcionamiento Pantalla Genérica.

Se dibujan los controles de grid y específicos en los detalles, estos controles pueden ser arrastrados de la herramienta desde las entidades del negocio.

Figura 11, Ejemplo funcionamiento detalles pantalla genérica.

A continuación una explicación de los métodos que implementa el formulario específico:

El método FrmOperadora_Load

Implementa las funcionalidades para ocultar y visualizar controles cuando el formulario se carga.

```
private void FrmOperadora_Load(object sender, EventArgs e)
    {
    GrdOperadoras.MultiSelect = false;
    tabControl1.TabPages.Remove(tabLista);
    tabControl1.TabPages.Remove(tabDetalle);
    ModeView();
    ActualizarGrid();
    }
```

El Método ActualizarGrid()

Provee el origen de datos del grid.

```
protected override void ActualizarGrid()
  {
operadoraEntityBindingSource.DataSource =
BLL.OperadoraBlc.SelectAll();
  }
```

El método GrdOperadoras_DoubleClick

Llama a la función grd_DobleClick del formulario padre.

```
private void GrdOperadoras_DoubleClick(object sender, EventArgs e)
  {
   grd_DobleClick(GrdOperadoras);
  }
```

El método insertar()

Hace validaciones de datos e interactúa con el método de inserción de la lógica de negocio.

```
protected override void insertar()
  {
```

```
        bool errores;
    errores = ValidaRequerido(TxtNombre, “Debe ingresar nombre”);
        if (!errores)
          {
BLL.OperadoraBlc.Insert(TxtNombre.Text,TxtDescripcion.Text);
      ModeView();
          }
        }
```

El método cmdActualizar_Click

Realiza las validaciones necesarias e interactúa con el métdodo Update de la lógica del negocio.

```
    protected override cmdActualizar_Click(object sender, EventArgs e)
      {
        bool errores;
    errores = ValidaRequerido(TxtNombre, “Debe ingresar nombre”);
        if (!errores)
          {
    BLL.OperadoraBlc.Update(codigo, TxtNombre.Text, TxtDescripcion.Text);
        ModeView();
          }
      }
```

El método toolBorrar_Click

Interactúa con la lógica de negocio para eliminar el registro y con la función borrar del formulario padre para las validaciones.

```
protected override void toolBorrar_Click(object sender, EventArgs e)
    {
     if (borrar(GrdOperadoras))
       {
     try
          {
BLL.OperadoraBlc.Delete(System.Convert.ToInt32(GrdOperadoras.Selected
Rows[0].Cells[0].Value));
          }
   catch
          {
   MessageBox.Show("No se puede eliminar, tiene modulos relacionados",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
          }
   ActualizarGrid();
        }
      }
```

4. CAPÍTULO IV - CASO PRÁCTICO, G5SHARING FILES

4.1. Introducción al aplicativo a desarrollarse

Gracias a la gran acogida del Internet en el mundo, hoy en día es posible compartir y acceder a información en cualquier momento y lugar. Que mejor oportunidad para desarrollar un sistema que permita la administración y control de los diferentes documentos que un ejecutivo puede llegar a necesitar en un día de trabajo normal, todo esto sin tener que regresar a su oficina a buscar el mismo; simplemente utilizando Internet.

Es decir, poseer una herramienta que permitirá organizar, compartir, y autorizar o restringir el acceso a cualquier tipo de documentos desde el Internet, de manera clara y sencilla. De esta forma será fácil y económico para una empresa tener al detalle las últimas versiones de sus documentos accesibles desde cualquier parte del mundo.

Además de lo expuesto anteriormente, la herramienta persigue un segundo objetivo claro, este es el manejo del control de cambios realizados sobre los archivos. Esto significa que cada actualización a los archivos de la empresa será guardada, y que los documentos importantes de la empresa solo podrán ser borrados por el administrador. De esta manera se reducirá de forma significativa la pérdida de información ya sea por entes maliciosos o por descuido del personal.

Combinada ambas funcionalidad, tendremos pues una herramienta integral que ayudará a la empresa a sacar el máximo provecho a la tecnología sabiendo que su información estará segura y disponible desde cualquier parte del mundo.

4.2. Estrategia y Alcance del Proyecto

4.2.1. Especificación de Roles

En el documento de especificación de Roles se debe designar las actividades que cada participante del proyecto va a desarrollar.

Se incluye la descripción del rol, el nombre de las personas asignadas a dicho rol, y las actividades que deben desarrollar por persona.

El documento se encuentra en la sección Anexos: Especificación de Roles.

4.2.2. Visión del Proyecto

En este documento se proyecta una visión general del proyecto, incluyendo los objetivos básicos, la arquitectura y la estructura global a desarrollarse. Incluyendo los puntos clave que identificarán si el proyecto es exitoso o no.

El documento se encuentra en la sección Anexos: Visión Alcance.

4.2.3. Definición de Personajes

Dentro del documento de definición de personajes, se muestran los distintos personajes que van a interactuar con el sistema.

El documento se encuentra en la sección Anexos: Definición de Personajes.

4.2.4. Definición de Escenarios

En este documento se muestran los distintos escenarios donde el aplicativo podría desarrollarse.

El documento se encuentra en la sección Anexos: Definición de escenarios.

4.2.5. Priorización de escenarios

De a cuerdo a las especificaciones del cliente el escenario seleccionado será el que utiliza como plataforma ASP .Net, es decir e escenario Web. Esto debido a que el cliente requiere el acceso al sistema desde cualquier equipo que disponga acceso a Internet.

4.2.6. Requerimiento de Servicio de Calidad

Muestra el resumen del los requerimientos del software.

El documento se encuentra en la sección Anexos: Requerimiento de Servicio de Calidad.

4.2.7. Priorización de Servicio de Calidad

En esta sección se priorizan los servicios para evaluar su impacto dentro del aplicativo.

El documento se encuentra en la sección Anexos: Priorización de Servicio de Calidad.

4.2.8. Estimación de costos

Este documento calcula el coste económico del software.

El documento se encuentra en la sección Anexos: Estimación de costos.

4.3. Diseñar el proyecto de desarrollo

4.3.1. Casos de uso

Presenta los casos de uso que servirán como base para el desarrollo.

El documento se encuentra en la sección Anexos: Casos de uso.

4.3.2. Definición de arquitectura

La arquitectura del proyecto, definirá el estándar técnico a ser utilizado, las clases a ser implementadas, la tecnología a utilizarse, y las diferentes capas a implementarse.

El documento se encuentra en la sección Anexos: Definición de Arquitectura.

4.3.3. Diseño de Base de Datos

Para el diseño de la base de datos se han considerado los requerimientos del sistema y se desarrollo el siguiente modelo:

Ver anexo Modelo de Base de Datos

4.3.4. Diagrama de clases.

Presenta las clases a ser implementadas en el desarrollo.

El documento se encuentra en la sección Anexos: Diagrama de clases.

4.3.5. Diseño de Interfases

A partir de los requerimientos del sistema, se genera un anteproyecto donde consta el diseño de las interfases de usuario.

Para ver el diseño de interfases, ver Anexo: Diseño de Interfases.

4.4. Desarrollo proyecto

4.4.1. Guía del proyecto

Presenta el cronograma de actividades a ser cumplido.

El documento se encuentra en Anexos: Guía del Proyecto.

4.4.2. Guía de iteraciones

Presenta las iteraciones por las que pasará el desarrollo.

El documento se encuentra en Anexos: Guía de Iteraciones.

4.5. Estabilización y pruebas.

4.5.1. Documento de Estabilización y Pruebas

Presenta el documento con las conclusiones de las pruebas y estabilización.

El documento se encuentra en la sección Anexos: Guía de Estabilización y Pruebas.

4.6. Implementación

4.6.1. Manual de Usuario

Es el manual que sirve al usuario para aprender a usar el sistema.

El documento se encuentra en la sección Anexos: Manual de Usuario.

4.6.2. Manual de Instalación

Es el manual técnico que muestra la forma de instalación e implementación del sistema.

El documento se encuentra en la sección Anexos: Manual de Instalación.

5. CAPÍTULO V - CONCLUSIONES

- La metodología MSF V4.0 es muy útil en el desarrollo de aplicaciones orientadas a objetos ya que al ser una metodología ágil de desarrollo de sistemas permite la interacción ente todos los involucrados en el proceso de desarrollo de un sistema.
- Con la utilización de la metodología MSF V4.0 se facilito el análisis y desarrollo del sistema G5Sharing Files, debido a que esta metodología permite seguir de forma sencilla el proceso de desarrollo de sistemas.
- El desarrollo de G5Sharing Files, a partir de una arquitectura robusta y una metodología probada, simplificó en mucho al equipo técnico y dio como resultado un aplicativo útil, escalable y de fácil mantenimiento.
- La utilización de patrones y prácticas para el desarrollo de una arquitectura robusta orientada a objetos para .NET Framework Versión 2.0 es fácil y permite crear patrones genéricos de desarrollo, para ubicar de manera fácil y concreta las diferentes funcionalidades dentro del aplicativo, e implementar nuevas de una manera ordenada.
- Una arquitectura orientada a objetos ayuda a crear aplicaciones organizadas, lo que implica que son fácilmente escalables y de fácil mantenimiento.
- Una metodología ordenada simplifica en mucho los tiempos de desarrollo, ayuda a estimar de una mejor forma los costos, y disminuyen el riesgo al máximo.
- La metodología MSF tiene una desventaja, esta radica en que no propone una metodología para la estimación de costos, por lo tanto el equipo de desarrollo en conjunto con los directivos son los encargados de buscar y utilizar una metodología para la estimación de costos del proyecto.
- Hay que tomar en cuenta que la metodología MSF sirve idealmente para proyectos pequeños y medianos, como es el caso del aplicativo desarrollado, dado que toma

bases del proceso unificado de desarrollo de software (PUDS) pero no utiliza todos los modelos UML, simplemente deja a consideración del grupo de trabajo los modelos que se deseen emplear. Esto podría resultar beneficioso para proyectos pequeños, pero en cambio la documentación en proyectos de gran magnitud quedaría incompleta.

6. CAPÍTULO VI - RECOMENDACIONES

- Es importante que el uso de una metodología de la misma casa de las herramientas, si bien no es obligatoria, facilita en gran manera la implementación de la misma, por ende se recomienda, usar las metodologías probadas y recomendadas por el fabricante de cada herramienta.
- Se recomienda la creación de la arquitectura dependiendo del tipo de aplicación que se va a implementar y de las tecnologías a usarse, así por ejemplo con Visual Studio 2005 se puede evitar el uso de DALC mediante el uso de Table Adapters que encapsulan los componentes lógicos de acceso a datos y las entidades del negocio. Sin embargo es muy importante el sentido común por parte del responsable de crear la arquitectura para, dependiendo de los objetivos que se persiga sea lo más simple y práctica posible, a la vez que brinda en la mayor medida posible los conceptos de estabilidad, seguridad, escalabilidad y fácil mantenimiento.
- Es importante estudiar a fondo las necesidades del cliente, y las capacidades del equipo de desarrollo para elegir la plataforma óptima para el proyecto.
- Se recomienda entonces, el uso de esta metodología MSF, en conjunto con arquitecturas orientadas a objetos para la creación de aplicativos pequeños o medianos, por su facilidad de implantación.
- Para proyectos grandes sin embargo, se recomienda PUDS, ya que esta es una metodología más completa donde se siguen un conjunto de fases predefinidas y el ciclo se cumple a cabalidad mediante la notación UML.

7. CAPÍTULO VII – BIBLIOGRAFÍA

- MSF For Agile Software Development, © 2005, 2006 Microsoft Corporation. All rights reserved. Version 4.0.1.
- Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and The Unified Process, Second Edition.
- Microsoft Development Network, http://msdn.microsoft,com
- Programa de capacitación 5 estrellas 2005, http://www.mslatam.com/latam/msdn/comunidad/dce2005/dashboard/dashboard_view.aspx
- .NET FRAMEWORK 2.0 Application Development Foundation, Tony Northup, Shawn Wildermuth, Microsoft, 2006

ANEXOS DEL DOCUMENTO

G5Corp

Especificación de roles

G5 Sharing Files

Autor	Luis Villarroel
Cargo	Co Director del Proyecto
Date	Septiembre 1

Versión: 1.0

Analista del negocio

Descripción del Rol

La misión principal del analista del negocio es entender y comunicar las necesidades solicitadas para el proyecto. Trabaja directamente con el cliente o cualquier ente relacionado que le ayude a comprender y visionar los requerimientos, mismos que luego se encargará en traducir en actores, escenarios y requerimientos de calidad de servicio que posteriormente serán tomados por el equipo de desarrollo para crear la solución.

Este rol también es el encargado de manejar las expectativas de funcionalidad del sistema. Dentro del proyecto el analista del negocio debe velar por los intereses del cliente, siendo este el ente que valida la experiencia del usuario y el cumplimiento de requerimientos. En fin, se encarga de ser el puente entre el equipo de desarrollo y los usuarios finales.

Personas encargadas:

Luis Villarroel

Cesar Montalvo

Entregables:

Documento a entregar	Persona Encargada
Visión del Proyecto	**Luis Villarroel**
Definición de Personas	**Cesar Montalvo**
Priorización de Escenarios	**Cesar Montalvo**
Requerimientos de Servicio de Calidad	**Cesar Montalvo**
Priorización de Servicios de Calidad	**Cesar Montalvo**
Estimación de Costos	**Cesar Montalvo**
Casos de uso	**Cesar Montalvo**
Diseño de Base de Datos	
Diagrama de clases	**Cesar Montalvo**
Diseño de Interfases	**Luis Villarroel**

Administrador del Proyecto

Descripción del Rol

La meta principal del administrador del proyecto es entregar valor de negocio dentro del horario y del presupuesto convenido. El administrador de proyectos debe cumplir las siguientes funciones: planeamiento y programación de actividades que incluyen el desarrollo y las diferentes iteraciones del proyecto, monitorear y reportar el estado del proyecto e identificar y mitigar los riesgos.

También se espera que el administrador del proyecto interactúe con el analista del negocio para planificar los escenarios y la calidad del servicio que se esperan en cada una de las iteraciones, de la misma forma debe estar en contacto con los arquitectos y los desarrolladores para poder estimar el trabajo, consultar con los encargados de las pruebas para planificarlas, y facilitar la comunicación entre todo el equipo de trabajo.

Personas encargadas:

Luis Villarroel

Documento a entregar	**Persona Encargada**
Project Management	**Luis Villarroel**

Arquitecto

La meta principal del arquitecto es la de garantizar el éxito del proyecto diseñando las bases de la aplicación. Esto incluye la definición de los modelos tanto lógicos como físicos de implementación del sistema. El objetivo del arquitecto es reducir la complejidad al dividir al sistema en partes simples y claras. La arquitectura resultante es extremadamente importante ya que no solo dictamina como se va a desarrollar el sistema, sino que también establece los rasgos que harán que el proyecto llegue a ser un éxito. Esto incluye su usabilidad,

mantenimiento y escalabilidad, cumpliendo con los estándares de seguridad y reacción ante cambios inesperados en los requerimientos.

Personas encargadas:

Luis Villarroel

Documento a entregar	**Persona Encargada**
Project Management	**Luis Villarroel**

Desarrollador

La meta principal del desarrollador es la de implementar la aplicación de a cuerdo a la planificación establecida. También se espera que el desarrollador ayude a especificar las características del diseño físico, la estimación de tiempos y esfuerzo para completar cada una de las tareas, construir o supervisar la implementación de las características, preparar el producto para la implementación y proveer de experiencia en el campo tecnológico dentro del equipo.

Personas encargadas:

Luis Villarroel

Cesar Montalvo

Documento a entregar	**Persona Encargada**
Código Fuente	**Luis Villarroel** **Cesar Montalvo**

Para cualquier consulta, a continuación se detallan los datos de cada uno de los participantes en el proyecto:

Luis Villarroel

Mail: lvillarroe1@hotmail.com

Teléfono: 02-2225749

Celular: 098539764

César Montalvo

Mail: cesar_mrc@hotmail.com

Teléfono: 02-2225749

Celular: 098640134

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios
02-09-2007	Luis Villarroel	1.01	Cambios en los roles.
27-09-2007	Luis Villarroel	1.02	Cambios en los roles.

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Cesar Montalvo	1.01		03-09-2007
Luis Villarroel	1.02		28-09-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Especificación de Roles
Autor	Luis Villarroel
Fecha de Creación	1 de septiembre del 2007
Última actualización	27 de septiembre del 2007

G5Corp

Visión/Alcance

G5Sharing Files

Autor	Luis Villarroel
Cargo	Co Director del Proyecto
Date	9/16/02 3:56 PM

Versión: 1.0

Estatuto de la visión

Toda empresa que utilice G5 Sharing Files, podrá manejar el control de cambios de sus documentos y almacenarlos en Internet de forma que estén disponibles desde cualquier parte del mundo.

Estatuto de Oportunidad

Luego de implementado el sistema, la empresa podrá olvidarse de los respaldos de los archivos importantes, así como de las seguridades de acceso a los mismos, de la misma forma sin importar el lugar de las sucursales podrán acceder a la documentación actualizada de la compañía.

Alcance

El alcance del proyecto involucra:

Módulo de seguridades: Incluye control de accesos al sistema, control de accesos por carpeta, control de accesos a historiales, control de modificación y eliminación de archivos.

Fuera del Alcance

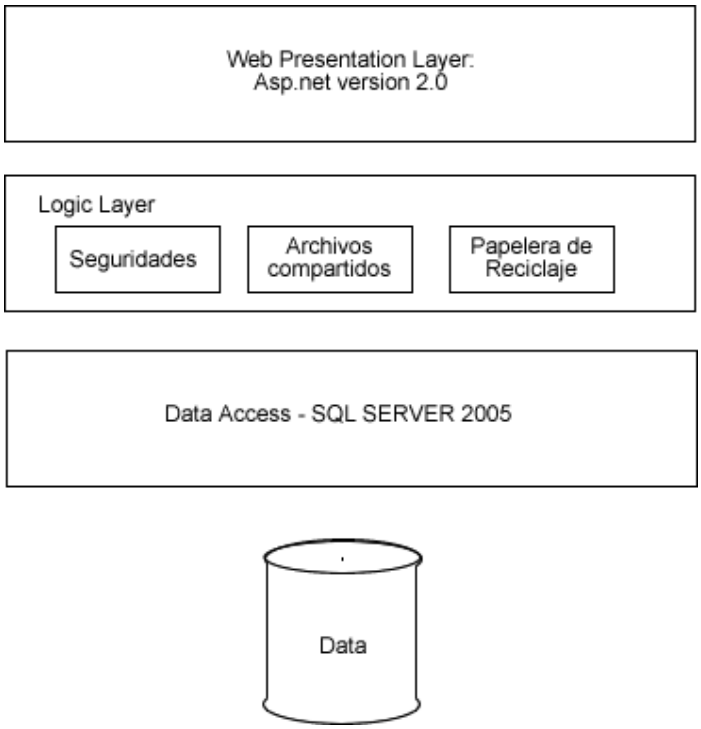
Las siguientes funciones se encuentran fuera del alcance de la versión 1.0 de la aplicación G5Sharing Files, las versiones 2.0 y 3.0 podrían tener la siguiente funcionalidad que no es factible implementar en la versión 1.0

Función	**Comentario**	**Versión donde se puede implementar**
Programación de experciencia de usuario WOW a través de AJAX	Puede ser implementada en la versión 2.0	2.0

Meta

Proporcionar una herramienta estable, rápida y segura que permita compartir archivos a través del Internet.

Concepto de Solución



Características de los usuarios

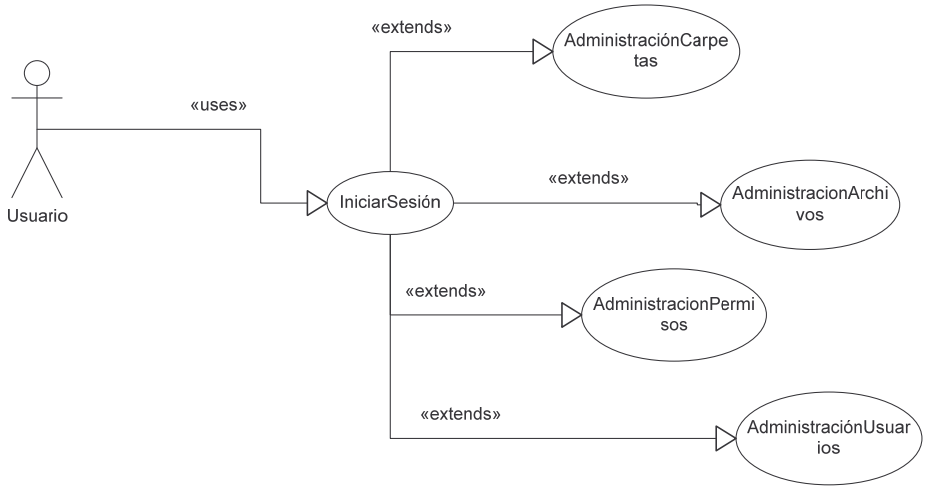
Los siguientes tipos de usuarios son esperados para el uso de la aplicación:

Usuario	**Breve descripción de las acciones del usuario**
Administrador	Será el usuario encargado de dar los permisos generales a los usuarios que usen el aplicativo.
General	Puede subir archivos al aplicativo, crear carpetas, dar permisos sobre los archivos, ver historiales, acceder a la papelera de recicleje.

Resumen de uso

EL sistema G5SHARINGFILES manejará los siguientes casos de uso. Los casos de uso serán completados durante la durante el proceso de extracción de información. Los casos de uso serán creados y priorizados. Los casos de uso seleccionados serán expandidos en los escenarios de uso. A continuación se presenta el diagrama de casos de uso general de la aplicación:

DIAGRAMA DE CASOS DE USO



Resumen de Requerimientos

La siguiente es una lista preliminar de requerimientos basada en las entrevistas iniciales y será desarrollada a fondo durante la definición de casos de usos.

- Controlar el acceso a entes externos.
- El usuario que crea la carpeta tienes permisos sobre los archivos que se encuentren en ella.

- La creación de carpetas debe ser del tipo árbol al estilo de Windows Explorer.
- Se debe guardar un historial sobre las modificaciones de los archivos. Incluyendo las copias de los archivos modificados.
- Si se elimina un archivo, este debe ir a una papelera de reciclaje.
- Los archivos puedes ser descargados desde cualquier parte del mundo.
- Los archivos pueden ser subidos desde cualquier parte del mundo

Criterios de éxito

Las personas de la institución podrán ver los archivos de la misma por Internet.

Las personas de la institución podrán subir y bajar los archivos.

Las personas de la institución podrán tener acceso al historial de cambios de sus archivos.

Las personas de la institución podrán organizar sus archivos en carpetas tipo Windows.

Las personas de la institución podrán restringir el acceso de lectura o modificación de los archivos según sea el caso.

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios
15-09-2007	Luis Villarroel	1.03	Cambios en los usuarios
20-09-2007	Luis Villarroel	1.04	Cambios en permisos
27-09-2007	Luis Villarroel	1.05	Visión a Futuro

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.03		16-09-2007
Luis Villarroel	1.04		22-09-2007
Luis Villarroel	1.05		29-09-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Documento de Visión y Alcande.
Autor	Luis Villarroel
Fecha de Creación	1 de septiembre del 2007
Última actualización	27 de septiembre del 2007

G5Corp

Definición de Personajes

G5sharing Files

Autor	Cesar Montalvo
Cargo	Co Director del Proyecto
Date	9/16/02 3:56 PM

Versión: 1.0

Los personajes definidos para la utilización de G5SharingFiles son los siguientes:

Administrador.- Este personaje se encarga de la administración de usuarios, roles y acciones principales para cada personaje del sistema. Este personaje es el único que puede resetear la clave de un personaje en caso de ser necesario.

Usuario.- El personaje usuario puede realizar acciones dependiendo de su rol y permisos asignados sobre cada carpeta. De esta forma puede crear, actualizar, descargar archivos de otras carpetas; en su carpeta propia puede crear o eliminar subcarpetas y archivos, de igual forma asignar permisos al resto de personajes.

Estos personajes utilizar el sistema de forma tal en la compartirán archivos y carpetas de acuerdo a las necesidades de cada grupo de personajes, llevando un historial de cambios realizados a los archivos por fecha y personaje.

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.04		05-09-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Definición de Personajes.
Autor	César Montalvo
Fecha de Creación	05 de septiembre del 2007
Última actualización	05 de septiembre del 2007

G5Corp

Definición de Escenarios

G5sharing Files

Autor	Cesar Montalvo
Cargo	Co Director del Proyecto
Date	9/16/02 3:56 PM

Versión: 1.0

Los escenarios propuestos para G5SharingFiles son los siguientes:

- Escenario Web Forms utilizando ASP .Net
- Escenario Windows Forms

Estos dos macro escenarios deben contener todos los escenarios que permitan interactuar a cada uno de los personajes definidos en el apartado anterior. Pera el sistema G5SharingFiles se ha definido los siguientes escenarios:

Escenario del Administrador.- En este escenario el administrador posee una interfaz que le permite la creación, actualización, eliminación y cambio de contraseña, dentro de la actualización de usuarios puede asignarle a cada usuario a un grupo de usuarios, estos grupos permiten la asignación de permisos de forma sencilla y rápida.

Escenario del Usuario.- El usuario posee un escenario en el cual puede crear, actualizar y eliminar carpetas y archivos, en los mismos que puede asignar permisos a grupos de usuarios, estos permisos permiten a otros usuario crear, actualizar, visualizar o eliminar archivos.

El escenario de usuario posee un navegador que le permite observar las diferentes carpetas a las que tiene acceso de donde podrá descargar y actualizar archivos dependiendo de los permisos otorgados.

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.04		06-09-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Definición de escenarios.
Autor	César Montalvo
Fecha de Creación	06 de septiembre del 2007
Última actualización	06 de septiembre del 2007

G5Corp

Requerimientos de Servicio de Calidad

G5sharing Files

Autor	Cesar Montalvo
Cargo	Co Director del Proyecto
Date	9/16/02 3:56 PM

Versión: 1.0

Los requerimientos de servicio son los siguientes:

- Poder Acceder a los archivos desde cualquier computador con acceso a Internet.

- Disponibilidad de acceso a los archivos de propios de cada usuario en cualquier parte del mundo a través de Internet.

- Acceso a archivos compartidos en el sistema dependiendo de los permisos asignados a cada grupo de usuarios.

- Registrar y almacenar las diferentes versiones de los archivos.

- Registrar el usuario que realizo modificaciones al archivo.

- Permitir la creación de carpetas a los usuarios.

- Permitir la asignación de permisos a cada un de las carpetas agrupando a los usuarios en grupos de trabajo.

- Permitir la eliminación de archivos de forma lógica no física para de esta forma evitar la perdida de información.

- Permitir al administrador la creación, eliminación y generación de claves para los usuarios.

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios
15-09-2007	César Montalvo	1.05	Permisos por usuarios y no por grupos de usuarios

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.05		18-09-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Requerimientos de Servicio de Calidad
Autor	César Montalvo
Fecha de Creación	10 de septiembre del 2007
Última actualización	15 de septiembre del 2007

G5Corp

Priorización de Servicio de Calidad

G5sharing Files

Autor	Cesar Montalvo
Cargo	Co Director del Proyecto
Date	9/16/02 3:56 PM

Versión: 1.0

Luego de realizar la matriz de priorización de Servicios de Calidad se ha llegado al siguiente cuadro de resumen:

	Prioridad	**Riesgo**	**Valor**	**Total**
Poder Acceder a los archivos desde cualquier computador con acceso a Internet.	1	2	5	8
Disponibilidad de acceso a los archivos de propios de cada usuario en cualquier parte del mundo a través de Internet.	1	2	5	8
Acceso a archivos compartidos en el sistema dependiendo de los permisos asignados a cada grupo de usuarios.	2	2	5	9
Registrar y almacenar las diferentes versiones de los archivos.	2	3	8	13
Registrar el usuario que realizo modificaciones al archivo.	3	3	7	13
Permitir la creación de carpetas a los usuarios.	2	1	4	7
Permitir la asignación de permisos a cada un de las carpetas agrupando a los usuarios en grupos	2	3	10	15

de trabajo.				
Permitir la eliminación de archivos de forma lógica no física para de esta forma evitar la perdida de información.	3	3	8	14
Permitir al administrador la creación, eliminación y generación de claves para los usuarios.	3	3	10	16
Nota:				
	PRIORIDAD	**VALOR**	**RIESGO**	
	1:secundario	1-10: Valor agregado	1:ordinario	
	2: importante		2:significativo	
	3: crítico		3:critico	

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.05		17-09-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Priorización de Servicio de Calidad
Autor	César Montalvo
Fecha de Creación	16 de septiembre del 2007
Última actualización	16 de septiembre del 2007

G5Corp

Estimación de costos

G5sharing Files

Autor	Cesar Montalvo
Cargo	Co Director del Proyecto
Date	9/16/02 3:56 PM

Versión: 1.0

Para la estimación de costos se utilizará el cálculo de los puntos de función ajustados. La metodología utilizada es la siguiente:

ILF (Internal Logical Files)			
	DET	**RET**	**Complejidad**
Grupos de Usuarios	3	15	Media
Usuarios	2	28	Media
Roles	3	14	Media
Carpetas	3	17	Media
Archivos	4	32	Media
Historial	3	20	Media

ELF(External Logical Files)			
	DET	**RET**	**Complejidad**

EI (External Imputs)			
	DET	**FRT**	**Complejidad**
Inicio de Sesión	6	13	Alta
Menú Principal	2	15	Media
Administración de Usuarios	4	9	Media
Edición de Grupos	4	2	Baja
Adminsitración de Carpetas	2	10	Media
Administración de Permisos de Carpeta	4	14	Media
Administración de Archivos	3	11	Media
Eliminación de Archivos	3	5	Media
Buscador	3	8	Media
Mi Cuenta	2	3	Media

EO (External Outputs)			
	DET	**FRT**	**Complejidad**
Inicio de Sesión	6	2	Media
Menú Principal	2	10	Media
Administración de Usuarios	3	9	Media
Edición de Grupos	3	6	Media
Adminsitración de Carpetas	2	8	Media
Administración de Permisos de Carpeta	5	13	Alta
Administración de Archivos	3	12	Media
Eliminación de Archivos	3	6	Media
Buscador	3	4	Media
Mi Cuenta	2	3	Baja

EQ (External Inquiries)			
	DET	RET	Complejidad

Obtenida la complejidad de los factores se debe contar cuantos de cada tipo existen para luego multiplicarlos por el factor indicado en la tabla, tal como se muestra a continuación:

	Simple	Medio	Complejo
External Imputs	1	8	1
External Outputs	1	8	1
External Inquiries	0	0	0
Internal Local Files	0	6	0
External Interfaces	0	0	0

Factor de Multiplicación

	Simple	Medio	Complejo
External Imputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6
Internal Local Files	7	10	15
External Interfaces	5	7	10

Para resumir se presenta la siguiente tabla:

Resultado de los Puntos de Función

	Simple	Medio	Complejo	Total
External Imputs	3	32	6	41
External Outputs	4	40	7	51
External Inquiries	0	0	0	0
Internal Local Files	0	60	0	60
External Interfaces	0	0	0	0
			Total	152

Luego realizamos el ajuste de los puntos de función tomando en cuenta los 14 factores de ajuste, tal como se muestra en la siguiente tabla:

	Atributos	Influencia
1	Comunicación de datos	4
2	Funciones distribuidas	0
3	Prestaciones	0
4	Gran uso de la configuración	2
5	Velocidades de las transacciones	4
6	Entrada de datos en línea	5
7	Diseño para la eficiencia del usuario final	3
8	Actualización de datos en línea	2
9	Complejidad del proceso lógico interno de la aplicación	3
10	Reusabilidad del código	3
11	Facilidad de instalación	3
12	Facilidad de operación	0
13	Localizaciones múltiples	5
14	Facilidad de cambios	3
	SUMA	37

FA = 0,65 + (0,01 * SVA)
1.02

PFA = PFNA * FA
155.04

Finalmente utilizando COCOMO se determina que el costo del proyecto es de $1.480,38 dólares americanos utilizando dos recursos en el desarrollo del sistema.

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios
19-09-2007	Cesar Montalvo	1.06	Cambios en las entradas
22-09-2007	Cesar Montalvo	1.07	Cambios en las consultas externas

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.06		20-09-2007
Luis Villarroel	1.07		24-09-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Estimación de costos
Autor	César Montalvo
Fecha de Creación	17 de septiembre del 2007
Última actualización	22 de septiembre del 2007

G5Corp

Casos de uso

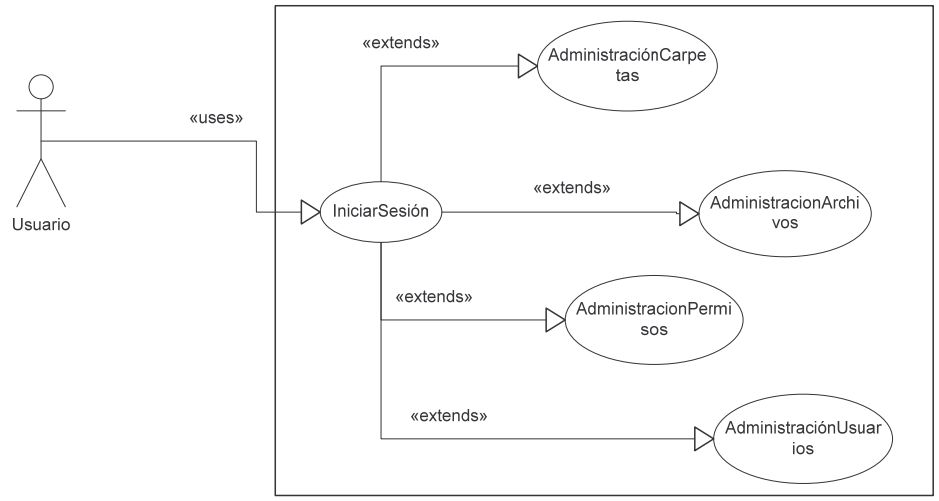
G5sharing Files

Autor	Cesar Montalvo
Cargo	Co Director del Proyecto
Date	9/16/02 3:56 PM

Versión: 1.0

El sistema G5SharingFiles atenderá los siguientes casos de uso:

DIAGRAMA DE CASOS DE USO



A continuación se describe cada uno de los casos de uso mencionados en el gráfico anterior:

Iniciar Sesión

Personajes.- Administrador del sistema, usuarios.

Descripción.- Es el primer caso de uso en el cual se verifica si la persona que desee acceder al sistema es o no un usuario registrado.

Funcionamiento Normal.- El usuario ingresa los datos de login (usuario y contraseña), una vez verificado el usuario se le presenta la interfaz correspondiente dependiendo del rol del mismo y el grupo al que pertenece.

Funcionamiento Alterno.- Si el los datos del usuario no han sido verificados se presenta un mensaje de error de inicio de sesión.

Administración de Carpetas

Personajes.- Administrador del sistema.

Descripción.- Este caso de uso permite a los personajes la creación, actualización y eliminación de carpetas.

Funcionamiento Normal.- El administrador de sistema selecciona una carpeta para luego administrar las carpetas interna, entendiéndose por administrar, crear, actualizar o eliminar carpetas.

Funcionamiento Alterno.- En el caso de presentarse algún problema el sistema mostrará los mensajes correspondientes a dicho problema.

Administración de Permisos

Personajes.- Administrador del sistema.

Descripción.- Este caso de uso permite al administrador del sistema signar permisos sobre las carpetas a cada usuario signando los siguientes permisos:

- Editar
- Eliminar
- Guardar
- Visualizar
- Crear

Funcionamiento Normal.- El administrador selecciona la carpeta contenedora de la carpeta a la que va a asignar permisos, selecciona asignar permisos de la carpeta deseada. Asigna los permisos para cada usuario o para todos los usuarios y guarda los cambios.

Funcionamiento Alterno.- De presentarse algún problema el sistema presentará mensajes de error con información sobre el problema presentado.

Administración de Archivos

Personajes.- Administrador del sistema, usuarios.

Descripción.- Dependiendo de los permisos asignados por el administrador del sistema, los usuarios pueden subir archivos, descargar archivos o eliminar archivos.

Funcionamiento Normal.-

Creación de Archivos.- El personaje cuyos permisos e facultan para crear archivos debe navegar en su computador y seleccionar el archivo que sea subir al sistema, posterior a la selección el personaje debe ingresar una descripción del archivo de considerarlo necesario.

Descarga de Archivos.- El usuario selecciona el archivo que desea descargar, luego de lo cual se le presenta el cuadro de descargas de Internet Explorer, el usuario debe seleccionar si lo quiere abrir o guardar en disco.

Actualización de Archivos.- El usuario debe descargar el archivo, guardarlo en su disco duro, y luego de editarlo subirlo al sistema, el sistema detectará el archivo por el nombre del mismo y guardará la versión anterior y la nueva versión.

Eliminación de Archivos.- El personaje seleccionará eliminará el archivo, el mismo que se colocará en los archivos eliminados. Únicamente el administrador del sistema eliminará los archivos de forma definitiva.

Funcionamiento Alterno.- En cual quier caso de presentarse algún problema el sistema presentará mensajes para poder corregir el problema.

A continuación se muestran los diagramas de casos de uso específicos para cada uno de los Personajes que utilizarán el sistema G5SharingFiles:

DIAGRAMA DE CASOS DE USO
ADMINISTRADOR DEL SISTEMA

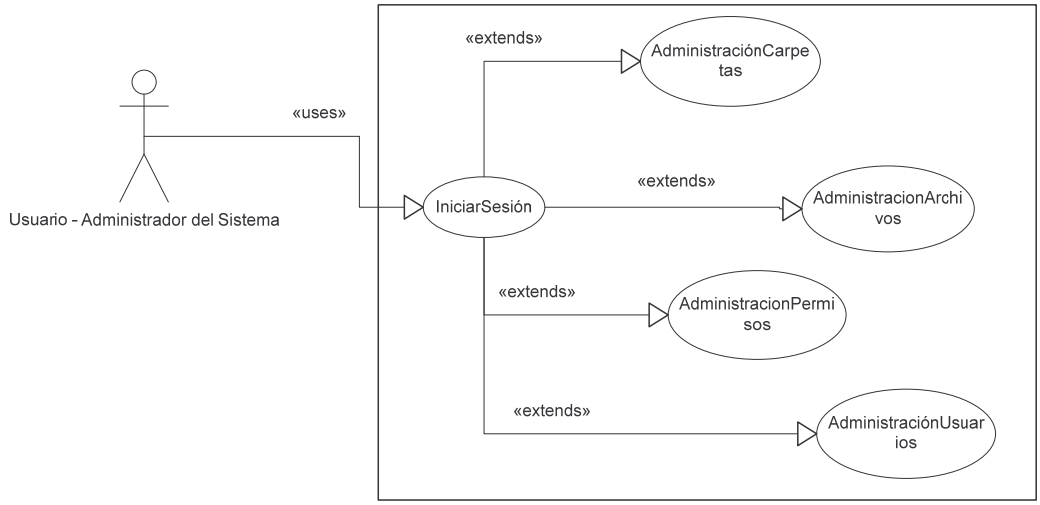
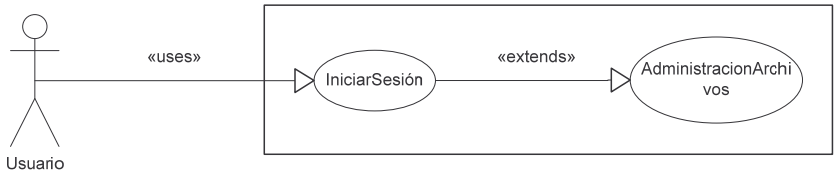


DIAGRAMA DE CASOS DE USO
USUARIOS



Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios
28-09-2007	Cesar Montalvo	1.08	Cambios en los permisos de usuarios
30-09-2007	Cesar Montalvo	1.09	Aparece el Buscador

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.08		29-10-2007
Luis Villarroel	1.09		01-10-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Casos de uso
Autor	Cesar Montalvo
Fecha de Creación	25 de septiembre del 2007
Última actualización	30 de septiembre del 2007

G5Corp

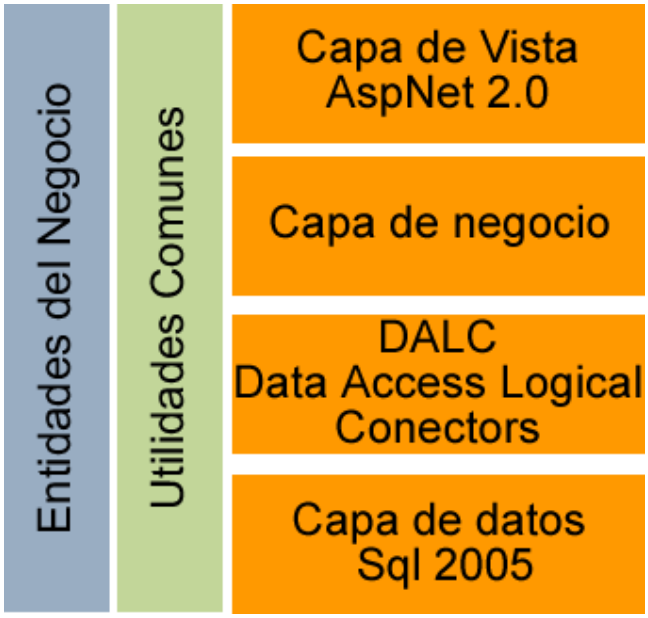
Definición de arquitectura

G5Sharing Files

Autor	Luis Villarroel
Cargo	Co Director del Proyecto
Date	1 de septiembre del 2007

Versión: 1.0

Diagrama general de la arquitectura:



Capa de entidades de Negocio

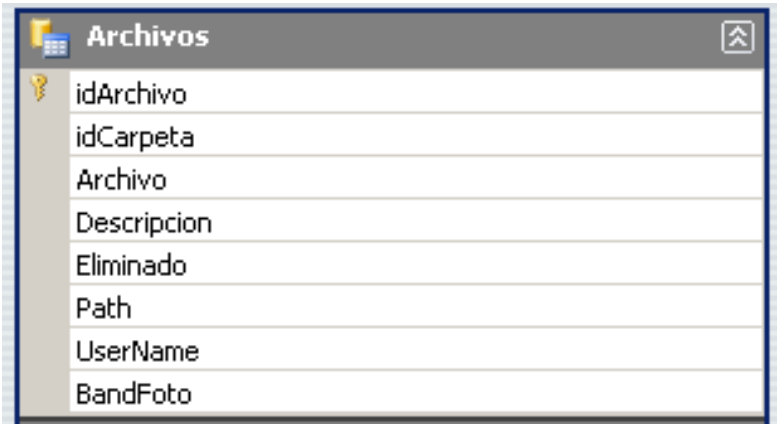
Definición:

Dado que se desea transportar datos desde la capa de vista hasta el almacén de datos, se necesita un medio para transmitir estos datos, debido a que es una arquitectura basada en objetos, los datos se transmitirán en modo de objetos que representan a entidades del negocio, los cuales están agrupados en esta capa.

Implementación

Se creará un proyecto del tipo Clase llamado Documentos.Business, que contendrá objetos del tipo TableAdapter,. La implementación práctica de las entidades del negocio se la hará a través de objetos DataTable, mismos que se encuentran en el espacio de nombre, System.Data y forman parte fundamental del objetos TableAdapter. Este objeto representa un número finito de registros, guardados independientemente en un arreglo de objetos DataRow, mismo que se encuentra en el espacio de nombre System,DataRow.. Un objeto DataRow de la misma forma está conformado por un conjunto de datos, cada dato es un valor de cada registro, y cada dato se almacena de forma independiente como un ítem de un arrlego de objetos DataColumn, mismo que se encuentra en el espacio de nombre System.DataColumn.

La siguiente imagen ilustra un DataTable desarrollado:



Capa de utilidades comunes

Definición:

Son componentes de distintos tipos, que pueden ir desde validaciones generales de tipos de datos, recursos matemáticos adicionales, mecanismos de administración operativa, tales como excepciones entre otros.

Implementación:

Se implementará un proyecto de clases llamado Documentos.Utils, mismo que contendrá todas las clases necesarias para implementar funciones comunes.

Capa de datos

Definición:

Esta capa comprende el almacenamiento de datos dentro del aplicativo.

Implementación:

Se implementará esta capa en Microsoft SQL Server 2005, dentro de una base de datos llamada Documentos.

Capa de acceso lógico a datos

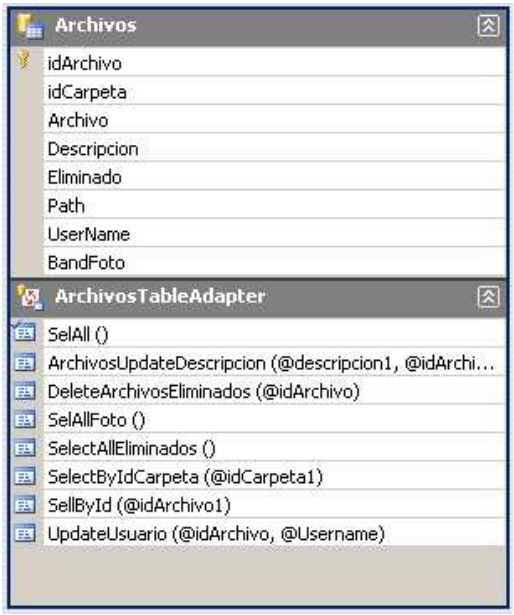
Definición:

Dado que hablamos de una arquitectura basada en objetos, esta capa comprende el conjunto de componentes donde, se realiza el mapeo de los mismos desde el modelo entidad relación al modelo de objetos.

Implementación:

Dentro del Documentos.Business, donde se implementó la capa de Entidades del Negocio, a través de los DataTables, que estaban contenidos dentro de los TableAdapters, se utilizarán los manejadores de los Table Adapters, para implementar los procesos de conexión lógica de acceso a Datos.

En la siguiente imagen se muestra cómo debe quedar implementada un TableAdapter, incluyendo a la Entidad de Negocio y al manejador de Acceso Lógico a Datos.



Las funciones implementadas en la clase (en el ejemplo ArchivosTableAdapter), interactúan directamente con la base de datos y se encargan de convertir, en el caso que el

resultado de la consulta sea un número finito de registros, a estos registros en objetos del tipo Archivos.

Capa del negocio

Definición:

Es el conjunto de componentes empresariales donde se almacenarán todas las reglas propias del negocio que maneja el aplicativo.

Implementación:

Se implementará un proyecto de clases llamado Documentos.Business, que se comunicará directamente con la vista y con la capa lógica de acceso a datos, para manipular los datos de acuerdo a los requerimientos del cliente, entre lo que se presente en la vista y las interacciones con ella y lo que se obtenga de la base de datos y los datos que en ella se manipulen.

Se deben implementar al menos cuatro funciones básicas, las cuales van a servir para insertar, actualizar, eliminar y seleccionar registros de la base de datos, a continuación se muestra un ejemplo.

```
//método de inserción, interactúa con el tableAdapter del proyecto ----
//Documentos.Business
public static void Insert(string nombre,string mail)
```

```
        {
            new
Entities.ContactosEntityTableAdapters.ContactosTableAdapter().Insert(nombre, mail);
        }

//método de actualización, interactúa con el tableAdapter del proyecto ----
//Documentos.Business

        public static void Update(int idContacto,string nombre,string mail)
        {
            new
Entities.ContactosEntityTableAdapters.ContactosTableAdapter().Update(idContacto,
nombre, mail);
        }

//método de elimincación, interactúa con el tableAdapter del proyecto ----
//Documentos.Business

        public static void Delete(int idContacto)
        {
            new
Entities.ContactosEntityTableAdapters.ContactosTableAdapter().Delete(idContacto);
        }
```

```
//método de selección interactúa con el tableAdapter del proyecto ----
//Documentos.Business
public static Entities.ContactosEntity.ContactosDataTable  SelectAll()
    {
        return new
Entities.ContactosEntityTableAdapters.ContactosTableAdapter().SelAll();
```

Capa de Vista

Definición:

Comprende el conjunto de componentes de Interfases de Usuario, que sirven como interfaz entre el aplicativo y el usuario. En el caso de .net, se dispone de algunos formatos para la presentación del aplicativo, tales como Windows Forms, Web Forms, o Dispositivos Móviles, separar la lógica de negocio, permite facilitar la creación de aplicaciones independientes de la vista.

Implementación:

La implementación se hará en un proyecto Web asp.net 2.0 llamado documentos.

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios
05-10-2007	Luis Villarroel	1.10	Optimización de la arquitectura

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Cesar Montalvo	1.10		08-10-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Definición de la arquitectura
Autor	Luis Villarroel
Fecha de Creación	01 de octubre del 2007
Última actualización	05 de octubre del 2007

G5Corp

Diagrama de Clases

G5Sharing Files

Autor	César Montalvo
Cargo	Co Director del Proyecto
Date	1 de septiembre del 2007

Versión: 1.0

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios
06-10-2007	César Montalvo	1.11	Optimización de Arquitectura
08-10-2007	César Montalvo	1.12	Utilización de Membership
09-10-2007	César Montalvo	1.13	Buscador

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.11		07-10-2007
Luis Villarroel	1.12		09-10-2007
Luis Villarroel	1.13		10-10-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Diagrama de Clases
Autor	César Montalvo
Fecha de Creación	05 de octubre del 2007
Última actualización	10 de octubre del 2007

G5Corp

Modelo de Base de Datos

G5Sharing Files

Autor	Luis Villarroel
Cargo	Co Director del Proyecto
Date	1 de septiembre del 2007

Versión: 1.0

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios
12-10-2007	Luis Villarroel	1.13	Tablas de Membership

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
César Montalvo	1.13		12-10-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Modelo de Base de Datos
Autor	Luis Villarroel
Fecha de Creación	10 de octubre del 2007
Última actualización	12 de octubre del 2007

G5Corp

Diseño de Interfaces

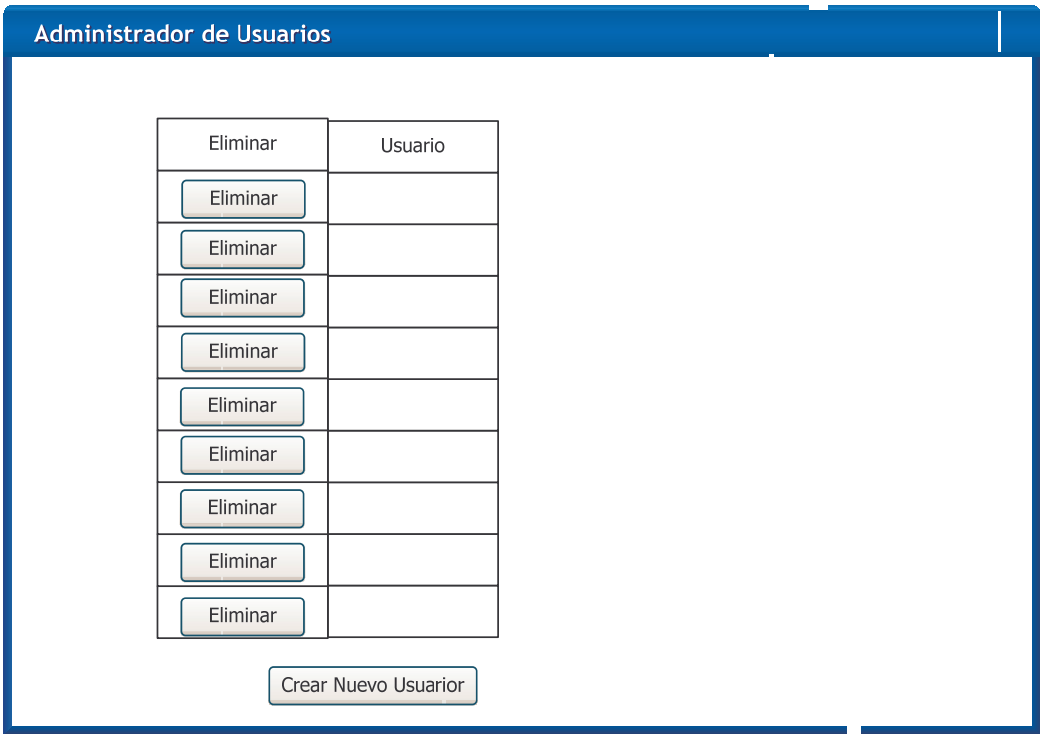
G5Sharing Files

Autor	Luis Villarroel
Cargo	Co Director del Proyecto
Date	9/16/02 3:56 PM

Versión: 1.0

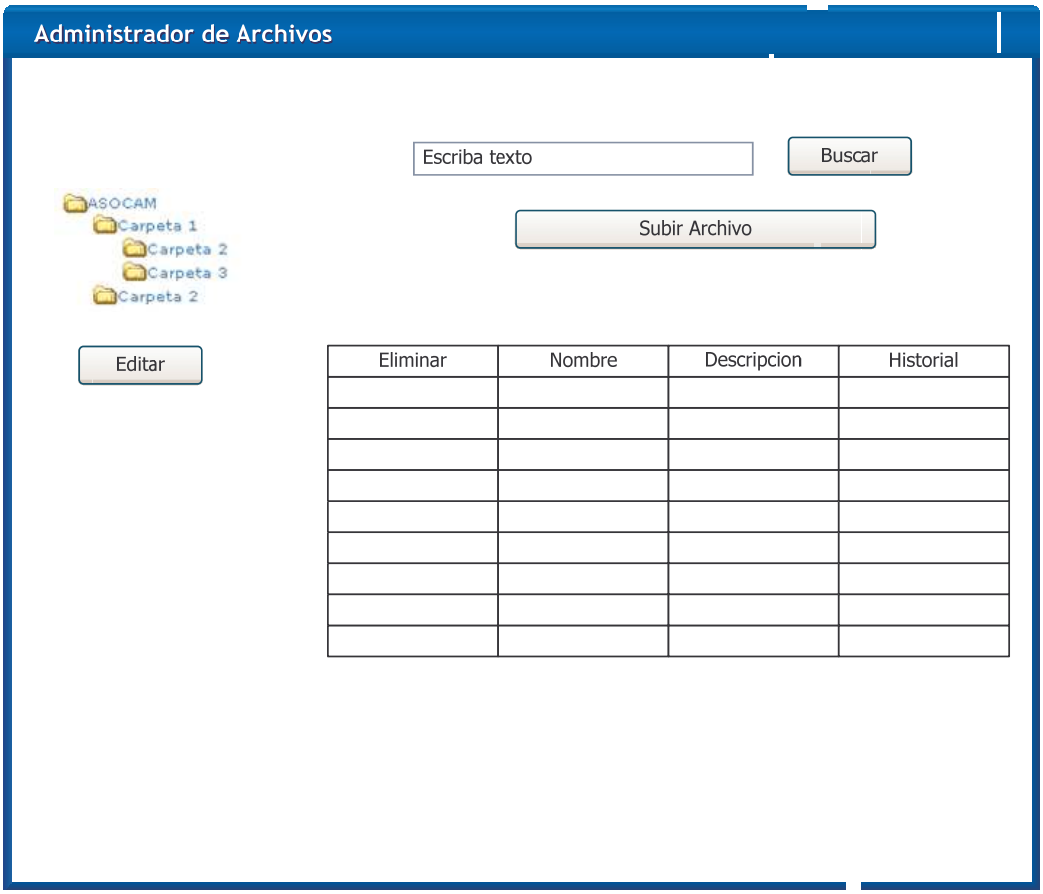
El presente documento sirve para dar al desarrollador las bases acerca de las interfases finales que el usuario desea, así como su funcionalidad.

Administrador de Usuarios:



Esta Pantalla sirve para administrar usuarios, y debe tener un botón para crear usuarios no administradores, y otro para eliminarlos.

Administración de Archivos:



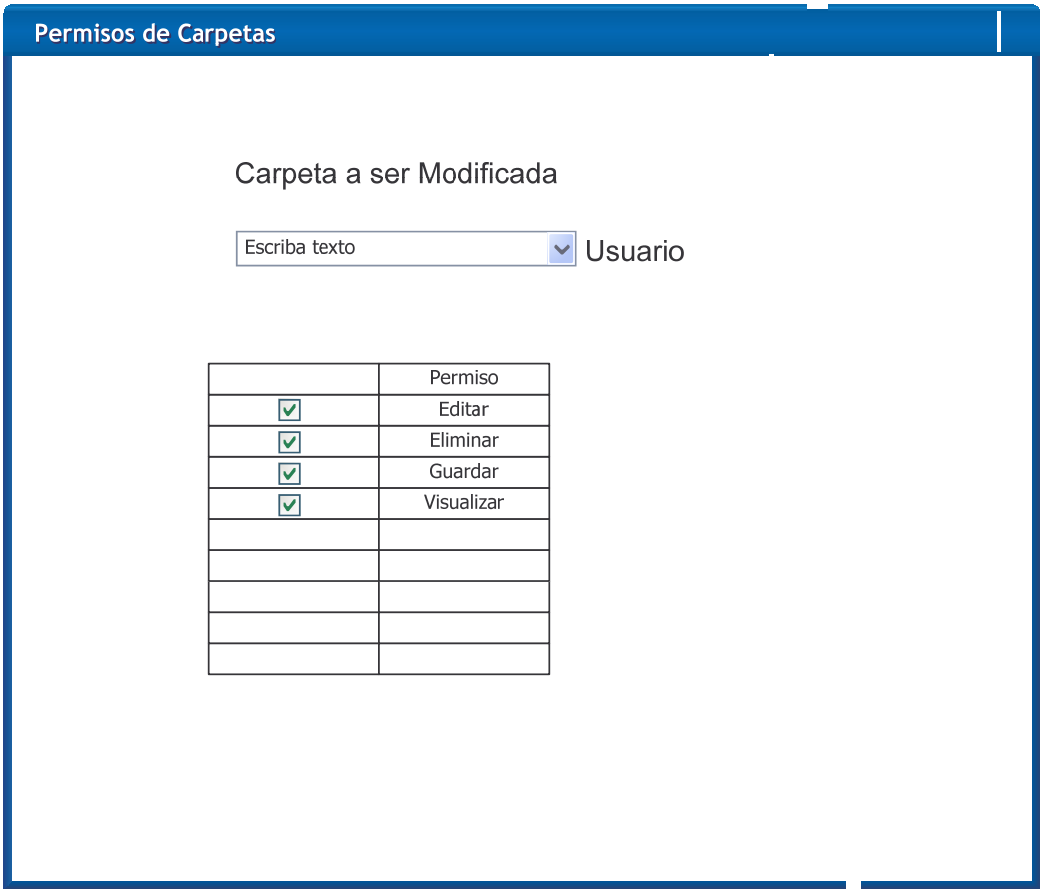
Esta es la pantalla principal del sistema, tendrá un árbol en la parte izquierda por donde podamos navegar, el usuario administrador podrá ver un botón en la parte inferior del

árbol que diga editar, con este podrá aumentar o eliminar carpetas, y dar permisos a los usuarios sobre estas.

Al lado derecho habrá un cuadro de texto y un botón buscar, que servirá para navegar y buscar el archivo a ser subido.

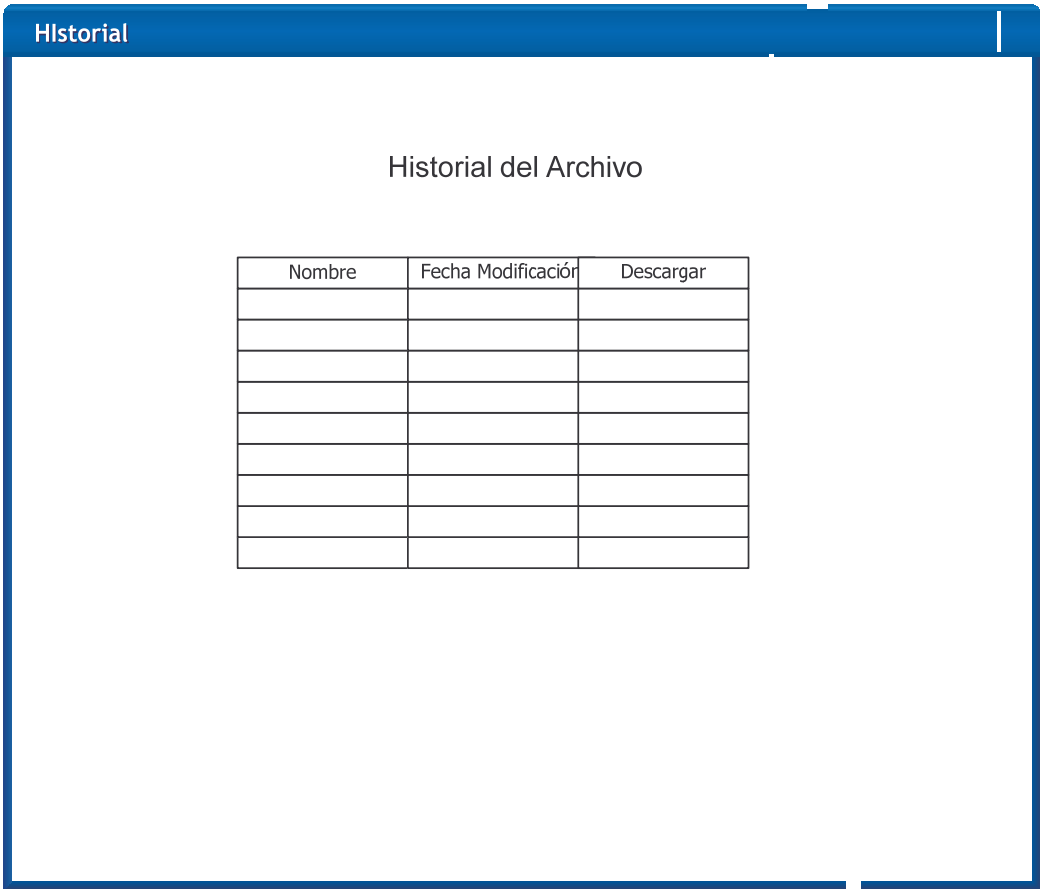
En un gris, se presentarán los archivos subidos en la carpeta, así como un hipervínculo por cada uno para eliminarlo, y otro para ver el historial de los cambios que se hagan sobre los archivos.

Permisos de Carpetas:



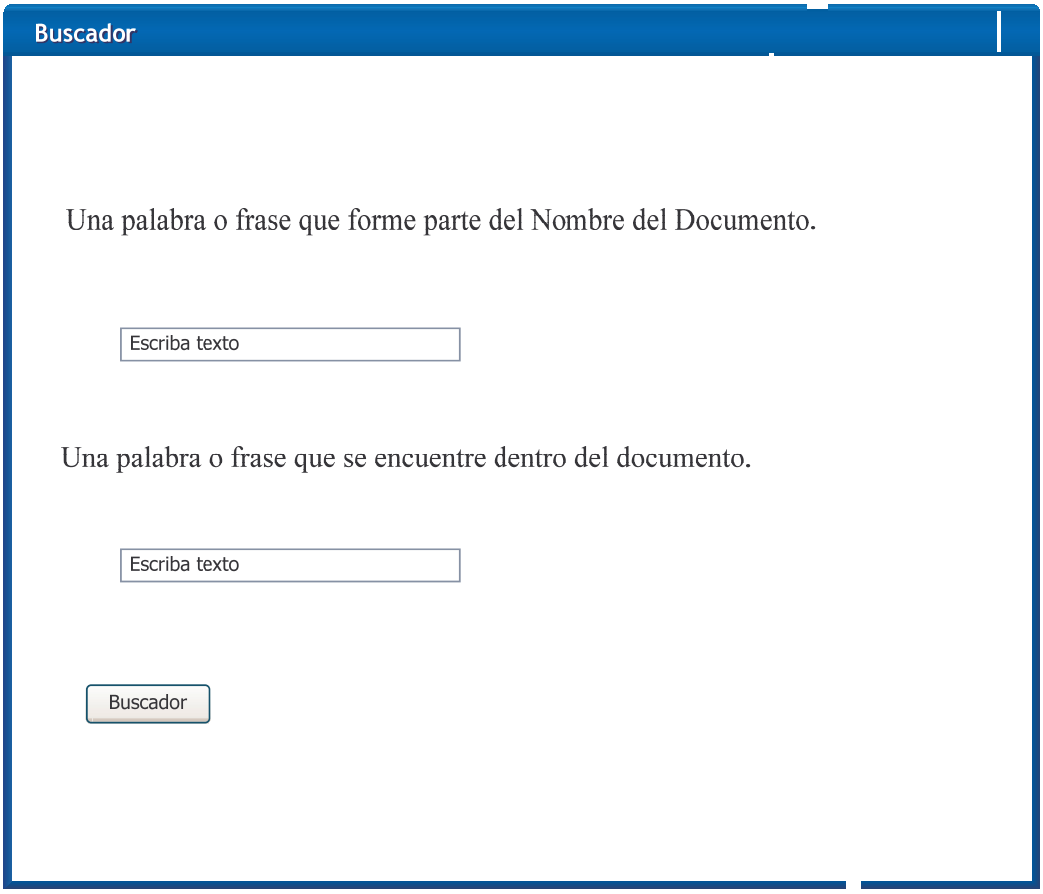
El usuario administrador, deberá poder modificar los permisos que cada usuario tenga sobre la carpeta, para esto sobre cada carpeta podrá hacer clic para modificar los permisos. En un combo se presentarán todos los usuarios, y dependiendo del usuario seleccionado, se dará uno o mas permisos de Editar, Eliminar, Guardar o Visualizar.

Historial



En esta pantalla se podrá ver el historial de todas las versiones del los archivos, incluyendo el digital listo para descargar.

Buscador:



Esta pantalla permite buscar archivos dentro del sistema, sea por una palabra dentro del documento o sea por una palabra en el nombre del archivo.

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
César Montalvo	1.13		13-10-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Diseño de Interfaces
Autor	Luis Villarroel
Fecha de Creación	13 de octubre del 2007
Última actualización	13 de octubre del 2007

G5Corp

Guía del Proyecto

G5Sharing Files

Autor	César Montalvo
Cargo	Co Director del Proyecto
Date	1 de septiembre del 2007

Versión: 1.0

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.13		13-10-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Guía del Proyecto
Autor	César Montalvo
Fecha de Creación	13 de octubre del 2007
Última actualización	13 de octubre del 2007

G5Corp

Guía de Iteraciones

G5Sharing Files

Autor	César Montalvo
Cargo	Co Director del Proyecto
Date	1 de septiembre del 2007

Versión: 1.0

Las iteraciones se realizarán de acuerdo a la Guía del Proyecto, en las cuales se destacan como puntos de control las siguientes:

- Administración de Usuarios
- Inicio de Sesión
- Administración de Carpetas
- Administración de Archivos
- Administración de Permisos

Cada una de las iteraciones incluye el desarrollo, pruebas e implementación parcial del sistema para la revisión del equipo de pruebas y la capacitación a los usuarios para de esta forma poder optimizar la utilización del sistema y garantizar la satisfacción del cliente con respecto al sistema.

En el caso de encontrarse errores o nuevos requerimientos estos deberán ser analizados y dependiendo de la complejidad se los solucionará o se deberá negociar el valor del nuevo desarrollo y los correspondientes ajustes que se deben realizar para cumplir con la planificación establecida.

Entre la finalización y el inicio de otra etapa se dispondrá de dos días para la revisión y aprobación del usuario ya que dado que el tiempo es limitado no se puede esperar mas tiempo.

Cabe recalcar que las pruebas que se realizan durante cada iteración son pruebas simples de caja negra, ya que luego de concluidas las iteraciones se llevará a cabo el proceso de Estabilización y pruebas, para finalmente llegar a la implementación.

Para dar por terminada cada una de las iteraciones se deben cumplir los siguientes requerimientos:

- Administración de Usuarios.- Creación, eliminación, actualización y asignación de grupos a usuarios.
- Inicio de Sesión.- Los usuarios pueden acceder al sistema utilizando su Nombre de Usuario y Contraseña, luego de lo cual se les presentará una interfaz en la cual puedan encontrar las carpetas y archivos a los cuales puede acceder.
- Administración de Carpetas.- Los usuarios pueden crear, editar o eliminar carpetas de la raíz.
- Administración de Archivos.- El administrador del sistema puede crear, editar o eliminar carpetas de la raíz, adicional debe existir una interfaz para realizar la búsqueda de archivos.
- Administración de Permisos.- El administrador del sistema puede asignar y remover permisos sobre las carpetas.

Para verificar las fechas de inicio de cada una de las iteraciones diríjase a la Guía del Proyecto.

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.13		14-10-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Guía de Iteraciones
Autor	César Montalvo
Fecha de Creación	14 de octubre del 2007
Última actualización	14 de octubre del 2007

G5Corp

Documento de Estabilización y Pruebas.

G5Sharing Files

Autor	Luis Villarroel
Cargo	Co Director del Proyecto
Date	9/16/02 3:56 PM

Versión: 1.0

Estabilización:

Durante la fase de estabilización se detectaron algunos errores, que luego de corregido han dado como resultado un sistema más estable y robusto.

Algunos de las conclusiones que se ha llegado de la fase de estabilización son:

Hay que tomar en cuenta que el sistema funciona con autentificación aspNet, por lo que las tablas y los SP de seguridad son preprogramados, y es importante que el desarrollador eche mano en lo menor posible a este código para evitar errores.

Es importante que si el sistema se instala en un servidor XP, se deshabilite el firewall de Windows o en su defecto, se habilite las entradas y salidas del puerto http para el buen funcionamiento del sistema.

Es importante que si Web Server y el servidor de base de datos van a estar en servidores diferentes, se debe habilitar las conexiones remotas por parte del SQL 2005.

A pesar que el sistema no tiene restricciones de peso de archivos, no se recomienda subir archivos de más de tres megas para una conexión 128-128, dado que esto puede hacer que el aplicativo colapse por el tiempo que se demora en subir.

Pruebas:

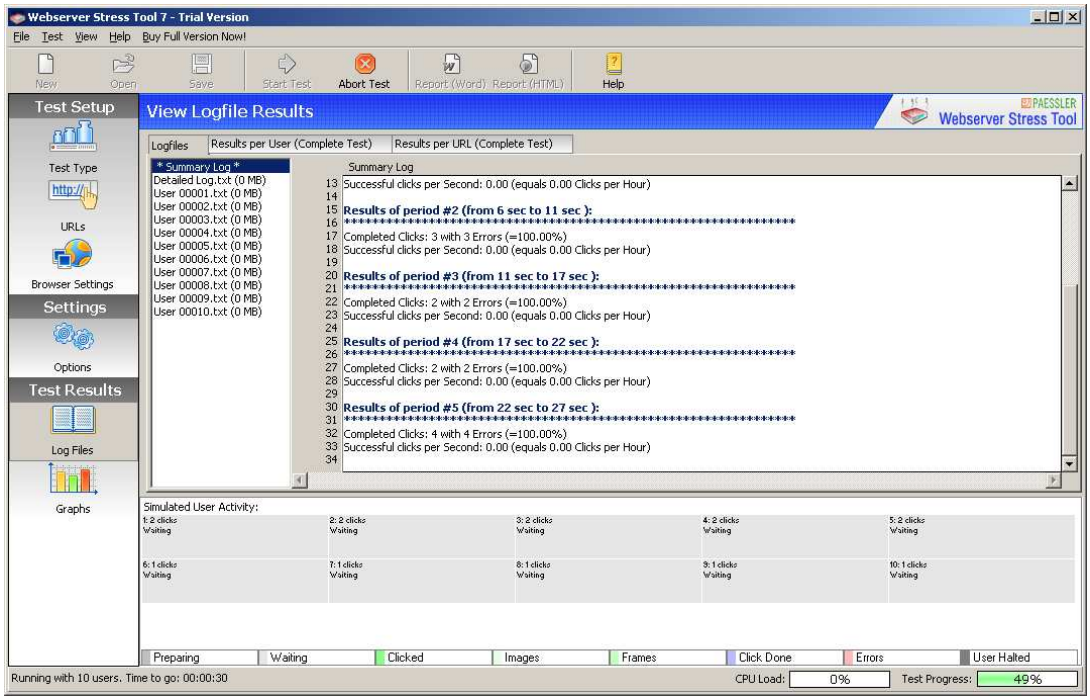
Dado que G5sharingfiles, es un sistema pequeño y no realiza ninguna operación matemática compleja, se ha descartado crear pruebas unitarias, de caja blanca y de caja negra.

La naturaleza Web del aplicativo, obliga a realizar pruebas de stress y comprobar la validez del entorno gráfico del mismo, para comprobar que cumpla con los estándares correspondientes.

Prueba de Stress

Los aplicativos Web, deben estar diseñados para soportar una alta concurrencia de usuarios y transacciones, es importante someter al aplicativo a pruebas de stress. La prueba de stress permite grabar una acción dentro del aplicativo Web, y simular ese proceso por muchos usuarios en un tiempo predefinido.

Las pruebas de stress se han realizado utilizando el Webserver Stress Tool 7, una herramienta third party, de la empresa Paessler.



La figura muestra a la herramienta mientras realiza las pruebas de stress

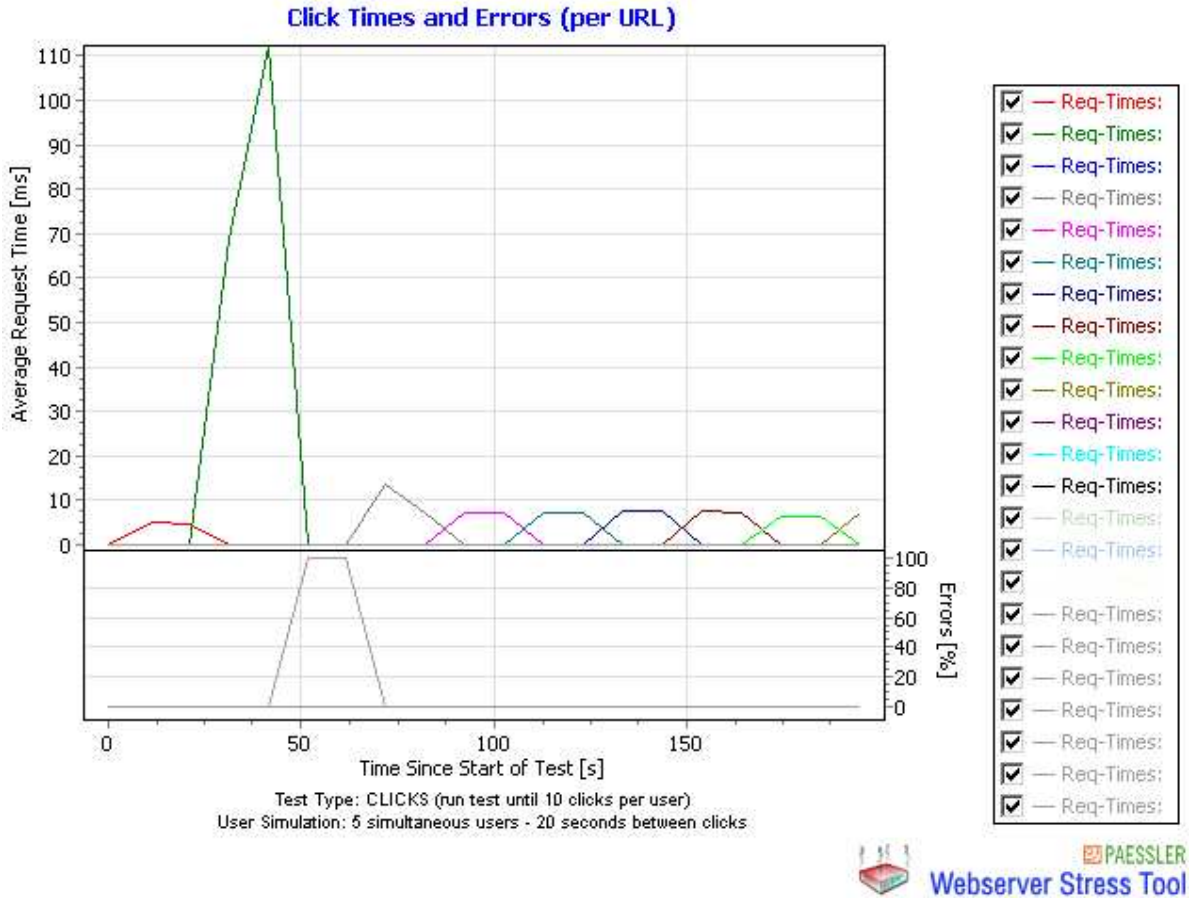
Descripción de la prueba realizada:

Se grabó en la herramienta las acciones más comunes a realizarse dentro del aplicativo, incluyendo comprobación de un usuario no válido. Luego inicio de sesión con un usuario válido. Como siguiente paso se subió un archivo de 2,5 mb. al aplicativo. A continuación se volvió a subir una versión posterior del mismo archivo. Se revisó el historial, se borró el archivo, se lo descargó desde la papelera de reciclaje de G5sharingfiles y se realizó una búsqueda por nombre de archivo.

Para la prueba se configuró a 5 usuarios concurrente, que repitieron el proceso 10 veces seguidas casa uno.

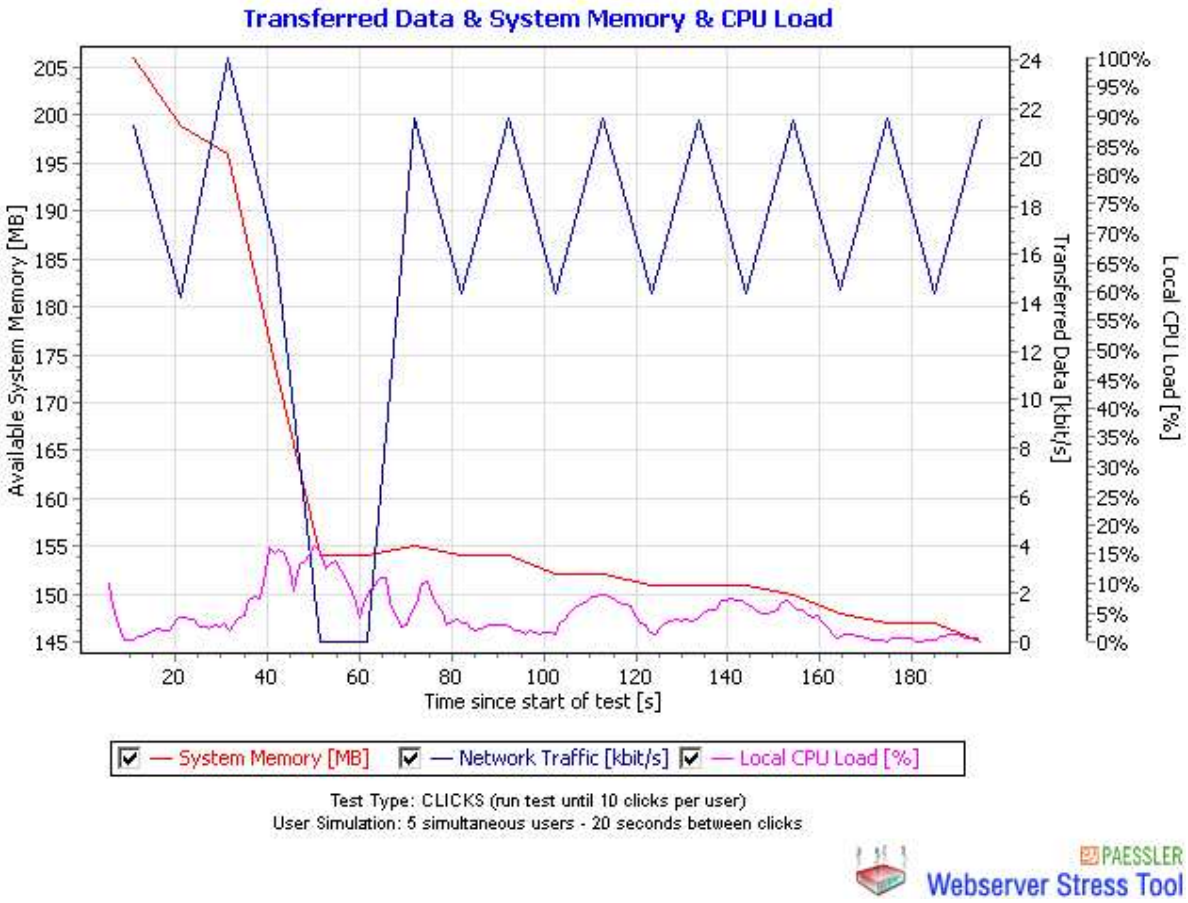
Estos son los resultados obtenidos:

La siguiente gráfica muestra la comparación entre el tiempo promedio para devolver la petición y el número de errores que generó el aplicativo.



Como se en la gráfica, el sistema presenta ningún error, y el tiempo de respuesta del servidor es bastante aceptable, teniendo como pico un punto de 110ms.

La siguiente gráfica muestra el consumo del procesador y la cantidad de información transmitida desde el servidor.

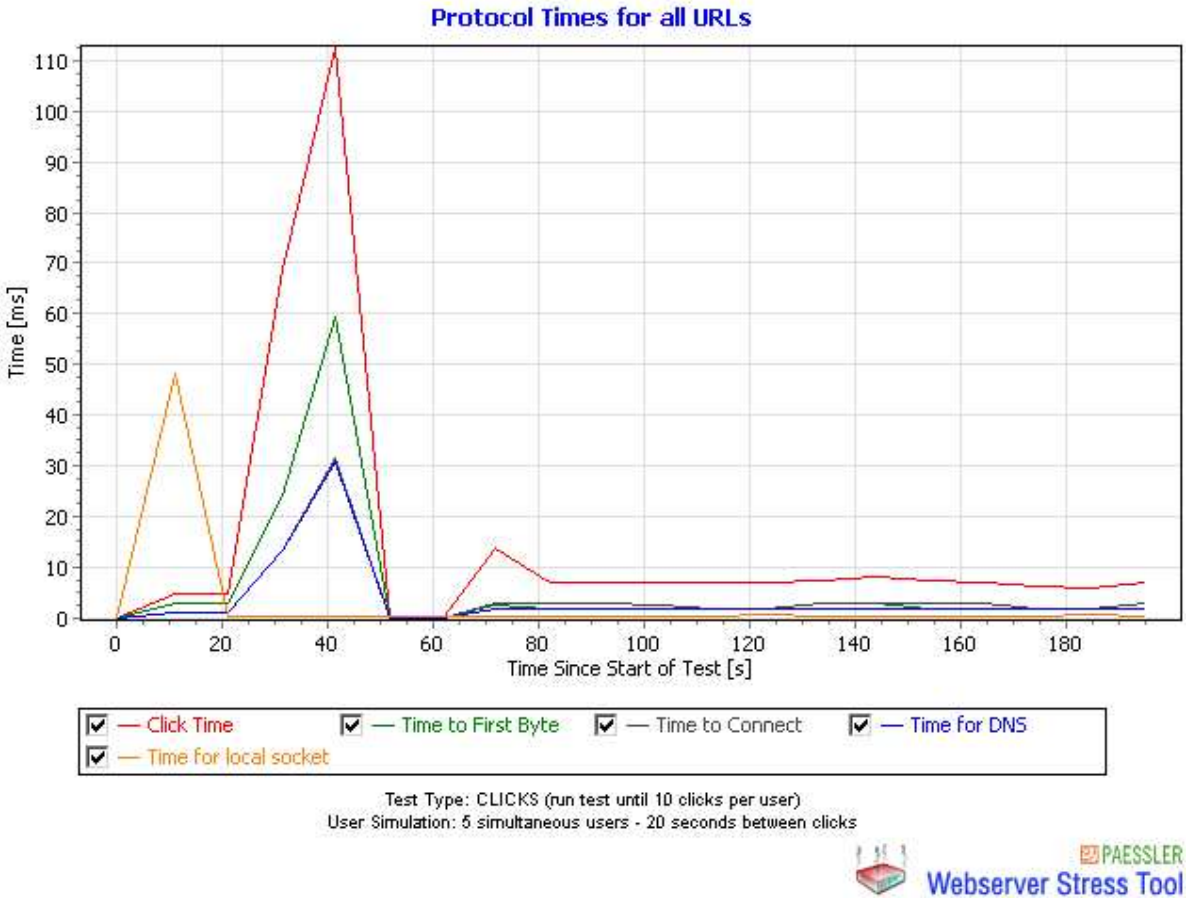


Como se ve en la gráfica la memoria usada del servidor va desde 205 megas hasta 145 megas, esto se debe a que la primera vez que se ejecuta el aplicativo necesita de memoria adicional para inicializar el mismo, luego de esta primera decreciente rápida, se ve que el promedio de uso de memoria es de 155 megas, por lo que se asume que un servidor con 1gb de ram, 8 veces más de lo necesario en la prueba, el aplicativo funcionará correctamente.

El uso del cpu tiene un tope de 20%, en este caso se trataba de un core duo de 2,2 ghz, se asume que un procesador parecido ejecutará correctamente al sistema.

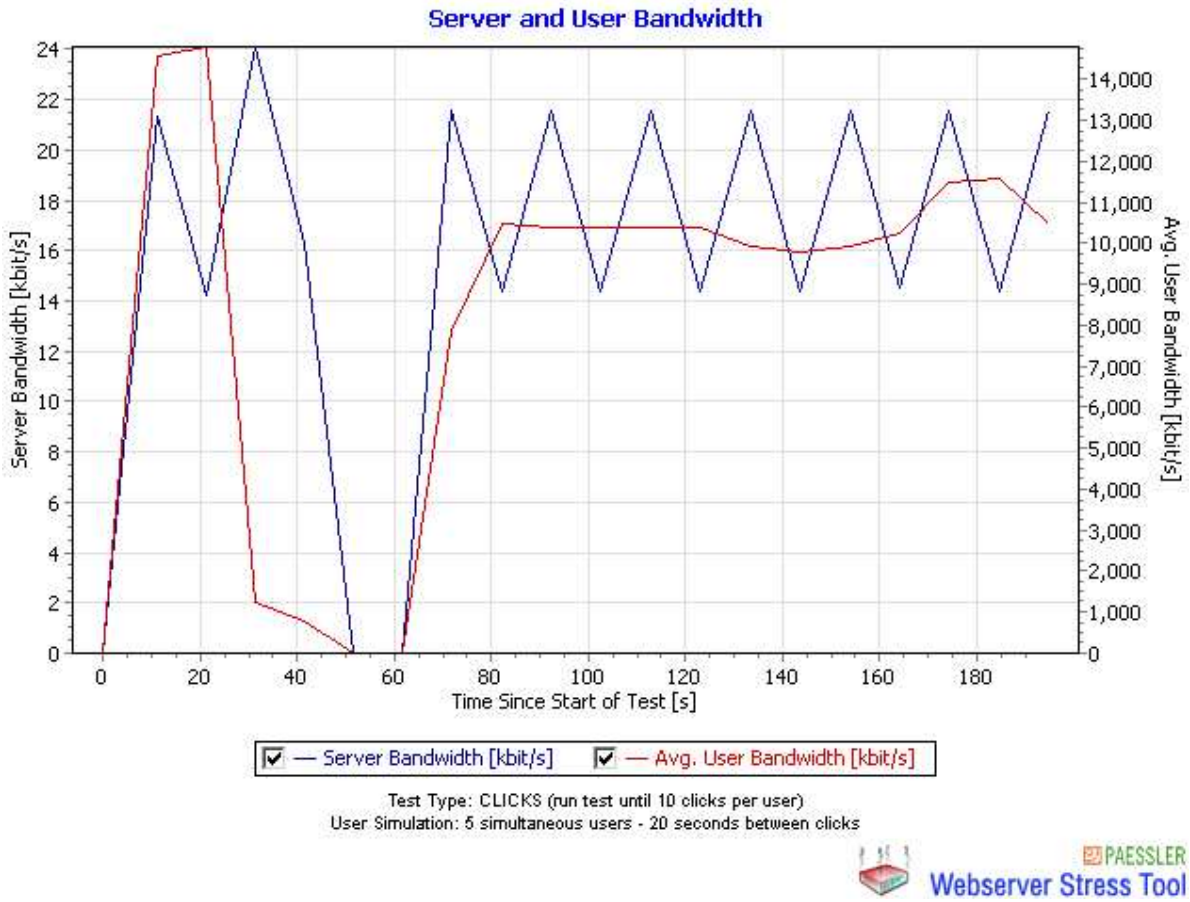
En cuanto al tráfico de la red esta se ocupó en un máximo de 22 kbits-s.

La siguiente gráfica muestra el tiempo de ejecución de los diferentes protocolos que intervienen en la tarea en el aplicativo dado.



Se ve que hay tiempos altos en la parte inicial de la prueba, tanto en la conexión, como en la respuesta del primer byte, el dns y el tiempo de respuesta al clic del usuario. Esto se debe a que el aplicativo necesita localizar al servidor, e inicializar el canal de comunicaciones por primera vez, lo cual toma tiempo, pero una vez abierto el canal, la comunicación fluye con tiempos de respuesta baja, lo cual indica la eficiencia del aplicativo.

La siguiente gráfica analiza el consumo de ancho de banda, tanto del cliente como del servidor.



Se ve un consumo casi nulo del ancho de banda del servidor, mientras el cliente tiene consumos de hasta 14000 kbits por segundo. Sin embargo esto se debe al alto consumo que significa subir un archivo de 2,5 mb por Internet. Hay que tomar en cuenta que las pruebas se hicieron en una red local, por lo que la subida fue casi instantánea, sin embargo, según las pruebas el aplicativo funcionaría, aunque no de manera óptima con conexión telefónica que posee en promedio un ancho de 33,000 kbits por segundo. Sin embargo en base a los resultados, se recomienda el uso de una conexión de banda ancha para el uso normal del aplicativo.

La siguiente tabla muestra los resultados en base a las pruebas dadas.

User No.	Clicks	Hits	Errors	Avg. Click Time [ms]	Bytes	kbit/s	Cookies
1	10	10	0	241	83,825	278.63	
2	10	10	0	237	83,825	283.05	
3	10	10	0	236	83,825	284.23	
4	10	10	0	239	83,825	280.62	
5	10	10	0	245	83,825	273.75	

Como se ve en la tabla anterior, el aplicativo no produjo errores para los 5 usuarios concurrentes.

La siguiente tabla muestra varios datos que verifican el comportamiento, rendimiento y errores del aplicativo, como se ve el mismo está funcionando de forma óptima.

URL No.	Name	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1		5	0	0.00	24	5
2		5	0	0.00	430	86
3		5	5	100.00	0	
4		5	0	0.00	55	11
5		5	0	0.00	36	7

6		5	0	0.00	36	7
7		5	0	0.00	37	7
8		5	0	0.00	37	7
9		5	0	0.00	32	6
10		3	0	0.00	21	7
11		0	0		0	
12		0	0		0	
13		0	0		0	
14		0	0		0	
15		0	0		0	
16		0	0		0	
17		0	0		0	
18		0	0		0	
19		0	0		0	
20		0	0		0	
21		0	0		0	
22		0	0		0	
23		0	0		0	

Prueba de validación de interfaz gráfica y experiencia de usuario.

Descripción del estándar	Cumple	No Cumple
El aplicativo se ajusta a una resolución de pantalla de 800x600 píxeles.	X	
El aplicativo maneja un estándar de fuentes y colores que pueden ser modificados fácilmente.	X	
El aplicativo maneja iconografía constante e intuitiva para el usuario.	X	
El aplicativo posee un sistema de navegación fácil que permite navegar entre las distintas páginas.	X	
El aplicativo posee mensajes de error en colores fuertes y con mensajes fáciles de comprender que guían al usuario en la solución de problemas	X	
El límite para llegar a la	X	

información más lejana dentro del aplicativo es de 5 clicks del ratón.		
El aplicativo no posee faltas ortográficas ni gramaticales.	X	
La interfaz de usuario es liviana y cada página no descarga al usuario más de 200 kbytes, incluyendo imágenes y scripts.	X	
El aplicativo es seguro y evita al máximo el uso de cookies.	X	

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
César Montalvo	1.13		31-10-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Documento de Estabilización y Pruebas.
Autor	Luis Villarroel
Fecha de Creación	30 de octubre del 2007
Última actualización	31 de octubre del 2007

G5Corp

Manual de Usuario

G5Sharing Files

Autor	Luis Villarroel y César Montalvo
Cargo	Co Director del Proyecto
Date	9/16/02 3:56 PM

Versión: 1.0

Inicio de Sesión:

Para iniciar sesión cada usuario debe ingresar el Nombre de Usuario y la Clave, tal como se muestra en la Figura 1.

Figura 1. Inicio de Sesión

Pantalla Principal:

En la pantalla principal el usuario puede encontrar un menú, localizado en el lado izquierdo de la pantalla, el mismo que contiene todas las acciones que el usuario puede realizar.

Para la administración de carpetas y archivos exististe un Tree View que despliega todas las carpetas a alas que el usuario tiene acceso.

El usuario también dispone de una papelera de reciclaje y de un buscador, tal como se muestra en la figura 2.

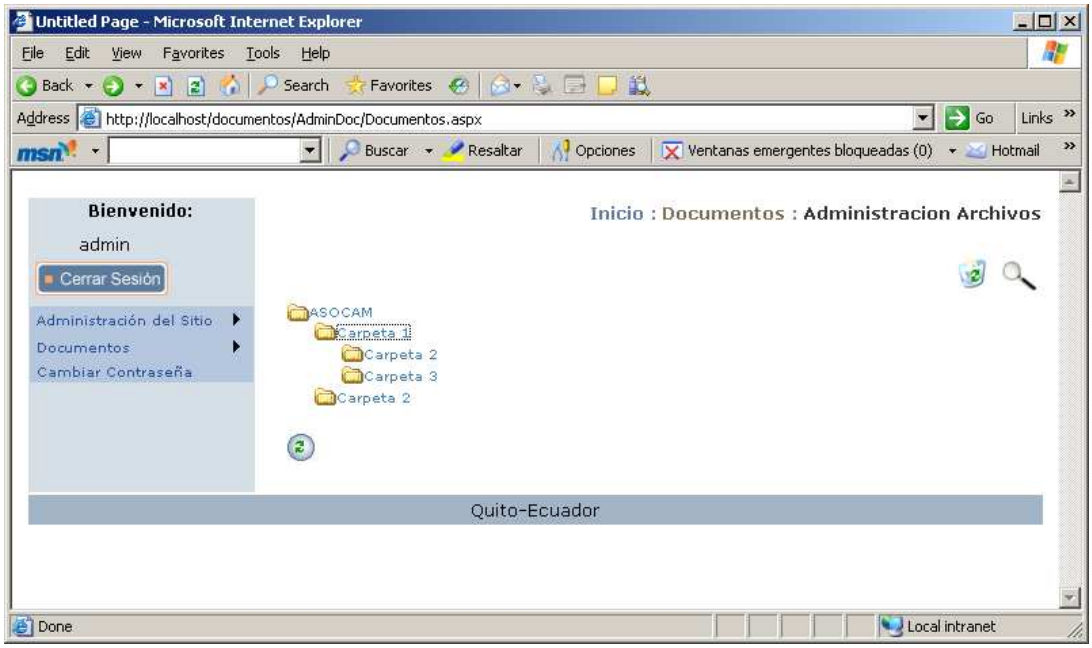


Figura 2. Pantalla Principal

Administración de Usuarios:

Esta pantalla es de uso exclusivo del administrador del sistema y permite la creación, eliminación, y asignación de grupos a usuarios, tal como se muestra en la figura 3.

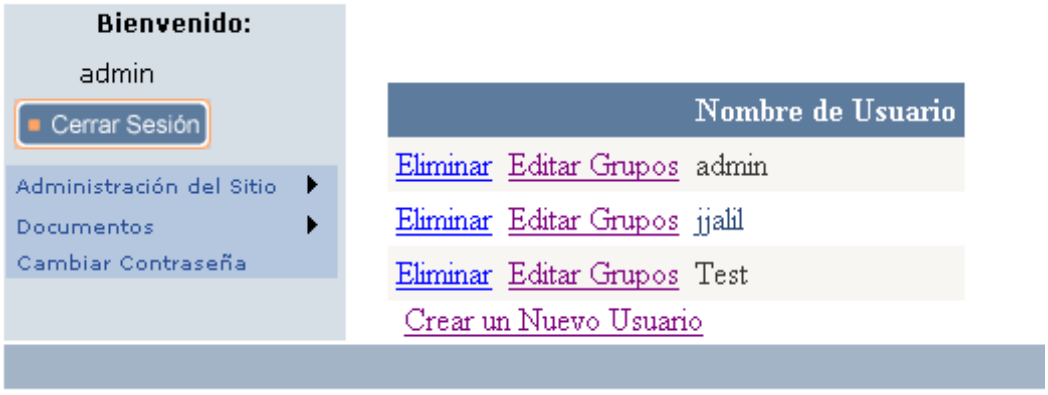


Figura 3. Administración de Usuarios

Los datos que se requieren para crear un usuario son los siguientes (véase la Figura 4):

- Usuario
- Clave
- E-Mail
- Pregunta secreta
- Respuesta secreta

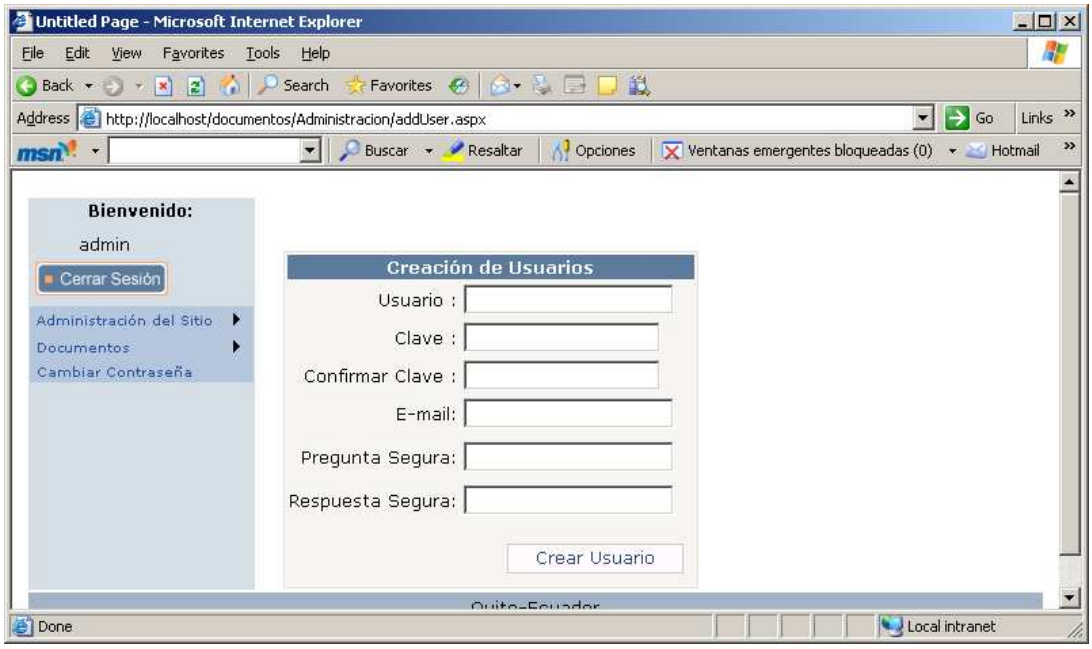


Figura 4. Datos de un Usuario

Navegación en Carpetas:

Para acceder a las carpetas el usuario debe realizar un clic sobre la carpeta deseada, lo cual despliega la siguiente pantalla (ver Figura 5).

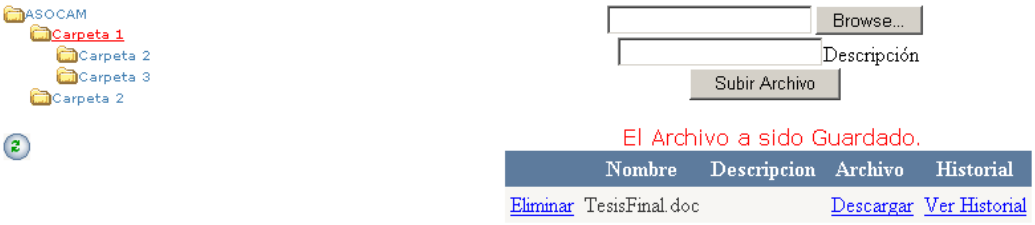


Figura 5. Navegación de Carpetas

En esta pantalla se pude descargar, eliminar o subir archivos dependiendo de los permisos que posea el usuario sobre la carpeta.

Administración de Carpetas:

En esta pantalla el administrador del sistema puede crear o eliminar carpetas, cada carpeta cuenta con un nombre y una descripción, tal como se muestra en la Figura 6.

Figura 6. Administración de Carpetas

Dentro de la administración de carpetas se encuentra la administración de permisos a grupos de usuarios, tal como se muestra en la Figura 7.

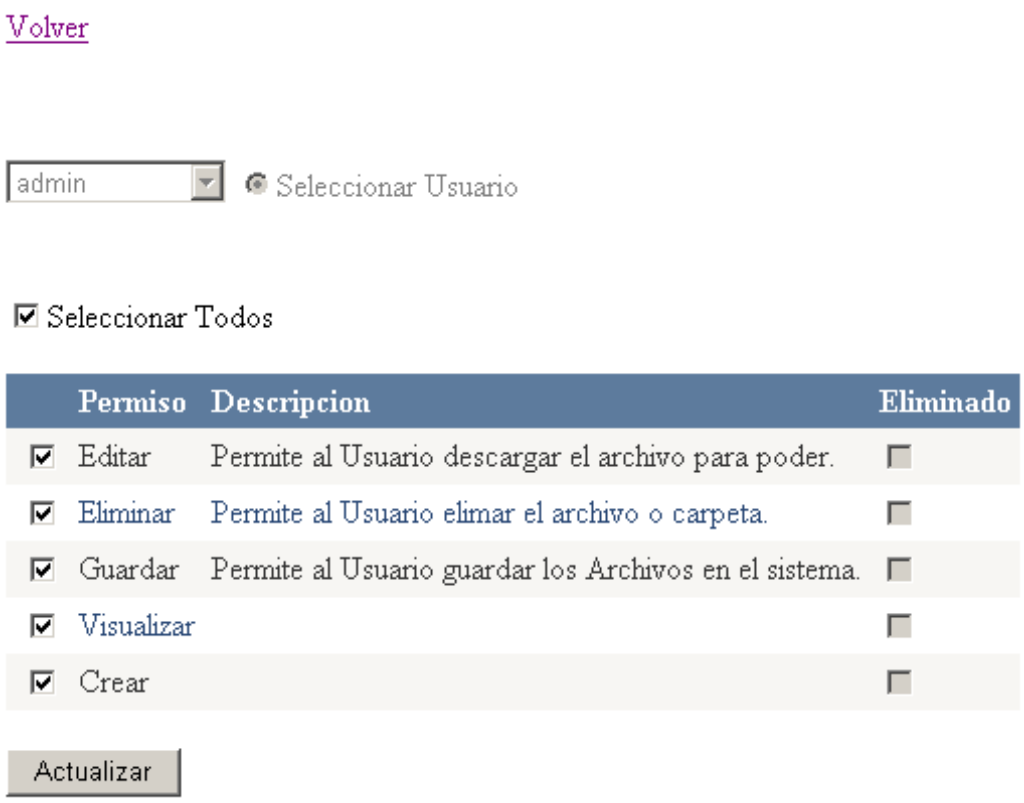


Figura 7. Administración de Permisos de Carpetas

Papelera de Reciclaje:

Permite descargar los archivos eliminados por los usuarios (Ver la Figura 8).

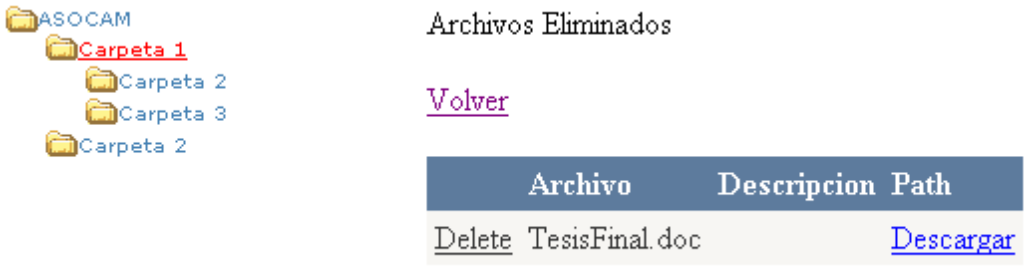


Figura 8. Papelera de Reciclaje

Buscador de Documentos:

El buscador de documentos permite localizar archivos cuyo nombre o frase se encuentre en el interior de un documento (véase la Figura 9).

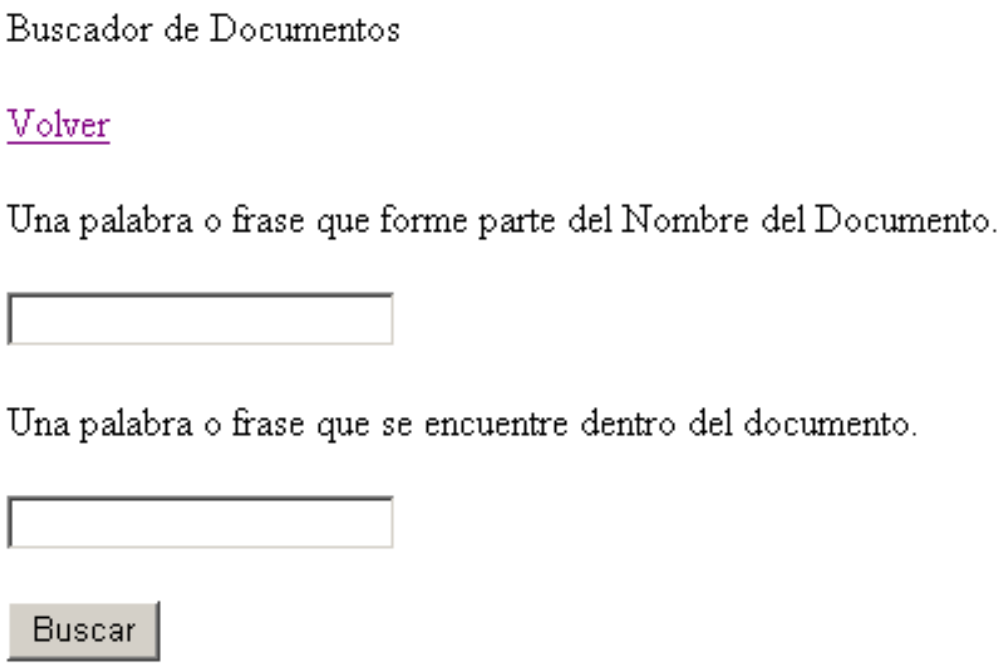


Figura 9. Buscador de Documentos

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.13		01-11-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Manual de Usuario
Autor	Luis Villarroel y César Montalvo
Fecha de Creación	1 de noviembre del 2007
Última actualización	1 de noviembre del 2007

G5Corp

Manual de Instalación

G5Sharing Files

Autor	César Montalvo
Cargo	Co Director del Proyecto
Date	1 de septiembre del 2007

Versión: 1.0

Prerrequisitos:

Para poder instalar el sistema G5Sharing Files el equipo debe poseer las siguientes características:

- Pentium III de 500 MHz o supuerior.
- 128 Mb. De memoria RAM.
- 1 Gb de memoria libre.
- Tarjeta de red.

Adicionalmente el equipo debe poseer los siguientes programas instalados:

- Microsoft Windows XP sp2 o superior
- Internet Information Server 6.0
- Microsoft Net Framework 2.0
- Microsoft SQL Server 2005

Instalación:

Para instalar G5Sharing Files se deben seguir los siguientes pasos:

1. Utilizando el administrador de IIS (Internet Information Server), se debe crear un nuevo sitio Web con el nombre de “Documentos” y configurarlo para que trabaje con Net Framework 2.0.
2. Copiar todos los archivos que se encuentran en el disco de instalación en la carpeta “G5Sahring Files Web”, al directorio creado en el paso anterior.
3. Restaurar el respaldo de la base de datos Documentos utilizando SQL Server 2005 Management Studio, el respaldo de la base de datos se encuentra en el disco de Instalación en la carpeta Base de Datos el archivo “Back Documentos”.

4. Ejecutar el Internet Explorer y acceder al sitio Web creado http://localhost/Documentos.
5. El usuario administrador del sistema deberá iniciar sesión con los siguientes datos:

   Usuario: admin.

   Clave: admin123

Hoja de revisión y firmas.

Record de Cambios

Fecha	Autor	Versión	Referencias de Cambios

Revisiones

Nombre	Versión Aprobada	Posición	Fecha
Luis Villarroel	1.13		01-11-2007

Distribución

Nombre	Posición

Propiedades del Documento

Ítem	Detalles
Título del Documento	Manual de Instalación
Autor	César Montalvo
Fecha de Creación	1 de noviembre del 2007
Última actualización	1 de noviembre del 2007

HOJA DE LEGALIZACION DE FIRMAS

ELABORADA(O) POR
VILLARROEL GONZALEZ LUIS RAMIRO
MONTALVO YEPEZ CÉSAR ALEJANDRO

_____
Villarroel González Luis Ramiro

_____
Montalvo Yépez César Alejandro

DECANO DE LA FACULTAD DE INGENIERIA

_____

Sangolquí, 25 de febrero de 2008