
# ELEVATIONSYS

Memòria del projecte

d'Enginyeria Tècnica en

Informàtica de Sistemes

realitzat per

*Guillermo Federico Prestigiacomo*

i dirigit per

*Rafael Fernández Gonzáles*

**Escola d'Enginyeria**

Sabadell, *Febrer* de *2011*

El sotasignat, ***Rafael Fernández González***,

professor de l'Escola d'Enginyeria de la UAB,

**CERTIFICA**:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva direcció per

***Guillermo Federico Prestigiacomo***

I per a que consti firma la present.
Sabadell, ***febrero*** de ***2011***

---

Signat: ***Rafael Fernandez***

## FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

**Títol del projecte: ElevationSys**

**Autor[a]: Guillermo Federico Prestigiacomo**

**Data:** ***Febrero 2011***

**Tutor[a]/s[es]: Rafael Fernandez**

**Titulació: Ingeniería Técnica en Informatica de Sistemas**

**Paraules clau** (mínim 3)


- Català: BackOffice, Sistema de Gestió Web, ASP.NET, SQL Server
- Castellano: BackOffice, Sistema de Gestión Web, ASP.NET, SQL Server
- Anglès: BackOffice, Web Managment System, ASP.NET, SQL Server


**Resum del projecte** (extensió màxima 100 paraules)


- Català: ElevationSys és un sistema de gestió web per a una empresa d'ascensors. El sistema conté diferents tipus d'usuaris i aquesta administrat per uns fluxos en relació de dependència que van marcant el ritme de la feina.
- Castellà: ElevationSys es un sistema de gestión web para una empresa de ascensores. El sistema contiene diferentes tipos de usuarios y esta administrado por unos flujos con relación de dependencia que van marcando el ritmo del trabajo.
- Anglès: ElevationSys is a web management system for an elevator company. The system contains different types of users and is administered by a relation of dependence flows that mark the rhythm of work.

# Resumen del proyecto

El proyecto de Gestión Web, ElevationSys, es un Backoffice para un sistema de empresa con una línea de producción. En este caso es una empresa que se dedica a la fabricación de ascensores. Esta herramienta permitirá la comunicación entre los diferentes departamentos de la empresa.

Comenzando por el departamento Comercial donde se crearán los clientes y los diferentes estudios para hacer las ofertas hasta que una es aceptada. Cuando una oferta de un ascensor es aceptada el departamento de planificación iniciará el proceso de fabricación del ascensor planificando los tiempos y creando las tareas que correspondan a cada uno. Las oficinas técnicas serán las responsables de la parte mecánica y eléctrica eligiendo las piezas necesarias para su producción. El proceso de ensamblado y producción de la máquina corresponde al departamento de Taller, donde todas sus tareas fueron creadas previamente por el departamento de planificación. Paralelamente a toda la fabricación se encuentra el departamento de contabilidad que es el encargado de la gestión de los cobros y pagos.

Para unir todos los departamentos el sistema estará basado en unos flujos donde reflejarán el estado de la máquina y donde el sistema dirá a cada persona perteneciente a cada departamento las tareas pendientes que tienen cada uno. Cada usuario es responsable de cumplir con su trabajo y una vez terminado avanzar el flujo a su siguiente estado. De esta forma el trabajo pasa a los siguientes responsables hasta terminar con la producción del ascensor.

# Índice de contenido

# 1. INTRODUCCIÓN

## *1.1. Objetivos.*

Uno de los problemas que sufre una empresa de varios departamentos es la comunicación. Mientras mas grande es la empresa mas difícil se vuelve comunicarse correctamente y esto hace que cada departamento en vez de trabajar juntos, trabajen por separado echándose la culpa los unos a los otros por cualquier inconveniente que pueda surgir en el medio de una producción.

Los sistemas informáticos pueden ayudar mucho en la colaboración entre departamentos. Pueden aislar el trabajo de cada uno y registrar los trabajos realizados por ellos. Esto ayuda a la comunicación y a la agilización entre departamentos.

La solución que ElevationSys quiere ofrecer es la de separar la función de cada departamento en diferentes nodos y unirlos a través de un flujo. Se pueden definir diferentes flujos que reflejen diferentes situaciones de la empresa. Cada nodo representa una tarea en un departamento y cuando este termine su labor, se pasará al siguiente nodo dando aviso al departamento responsable.

Esto simplifica la labor de cada departamento de hacer saber que ya termino con su trabajo. Al finalizar tiene que terminar con su nodo para avanzar al siguiente estado del flujo. El sistema se encargará de avisar a los responsables. El estado del flujo reflejara el estado de la máquina y se podrá saber si se están cumpliendo con las previsiones o si hará falta un cambio. Al finalizar los flujos se finalizará la producción y estará la máquina lista para entrega.

La empresa ElevationSys es una fabrica de Ascensores. El proceso de un ascensor se inicia con su venta a través del departamento comercial. Una vez un cliente solicita un nuevo ascensor el departamento de planificación se encarga de planificar el proceso de fabricación, creando las tareas, gestión de las compras y poniendo una fecha de entrega de la maquina. Luego pasaría todo a los departamentos de oficinas técnicas donde se encargan el diseño mecánico y eléctrico de la máquina. Una vez todo esta definido pasaría a ser el momento de ensamblado en el departamento de Taller. Paralelamente el departamento de contabilidad estará a cargo de la administración de los pagos y el manejo de las facturas.

Básicamente esto es lo que se quiere mostrar en el sistema. La interacción entre un departamento y otro a través de flujos y estados. Para ello hay creado una estructura de base de datos relacional donde se guarda toda la información. Fundamentalmente la estructura básica de la empresa nace en el cliente. Este tiene diferentes estudios que marca el proyecto de venta. Y dentro de cada estudio hay operaciones que definen de una forma mas técnica la maquina a vender. Dentro de un estudio se pueden hacer varias operaciones hasta que haya una que le interese al cliente y es cuando se pone en marcha el proceso de fabricación en esa operación aceptando el proyecto de estudio. Cuando se acepta la operación es cuando se inicia el flujo, antes de eso es un tema puramente comercial.

Cada usuario pertenecerá a un departamento y dependiendo de esto tendrá acceso a

diferentes partes del sistema. La idea de esto es centralizar mejor el trabajo de cada uno y no mostrar cosas que no corresponden. Esto se otorgar mediante la utilización de llaves. Cada pantalla tendrá una o mas llaves. Si el usuario tiene esta llave entonces tendrá el permiso de utilizarla. Sino no aparecerá. Existen tantos grupos como departamentos, además del administrador. Cada uno tendrá una función en el sistema. Las llaves se otorgar a grupos y a usuarios. Cada usuario también pertenece a un grupo. Esto hace que un usuario pueda tener mas llaves por mas que pertenezca a un grupo en particular.

Los inconvenientes que se puedan llegar a generar es básicamente a la adaptación de los usuarios al sistema. Para ello simplemente se mostrarán las ventajas que ofrece el sistema y de la fácil implementación que conlleva utilizarlo. No deja de ser una herramienta para gestionar la empresa y liberar a los usuarios de las tareas monótonas del día a día.

Hay muchas mejoras para hacer ademas de lo ya implementado. El sistema esta hecho para ser escalable e incorporar mejoras en cualquier momento.

## *1.2. Estructura de la memoria*

La memoria esta dividida en 8 capítulos

1. Introducción: en este capitulo se da una breve reseña de los objetivos y la descripción del sistema a realizar.

2. Estudio de viabilidad: Se mostrará el estudio de viabilidad que indica las posibilidades de llevar a cabo el proyecto. Mostrando detalles económicos, legales y técnicos.

3. Modelo de desarrollo y herramientas a utilizar: Se explica acerca del tipo de metodología que se ha utilizado para el desarrollo del proyecto.

4. Análisis de aplicaciones: en este apartado se explican los requerimientos funcionales y no funcionales.

5. Diseño: en este apartado se mostrarán todos los diseños que representan el sistema. Se utilizarán diagramas UML , detalles de la interfaz y requerimientos para el correcto funcionamiento.

6. Pruebas: Aquí se explicarán las pruebas realizadas a lo largo del desarrollo.

7. Conclusiones: se muestran las conclusiones generales referentes al proyecto. Haciendo referencia a los objetivos conseguidos y los que no, posibles ampliaciones y la opinión personal.

8. Bibliografía: se mostrarán las fuentes que ayudaron a confeccionar este proyecto.

9. Agradecimientos: apartado de agradecimientos

# 2. Estudio de viabilidad

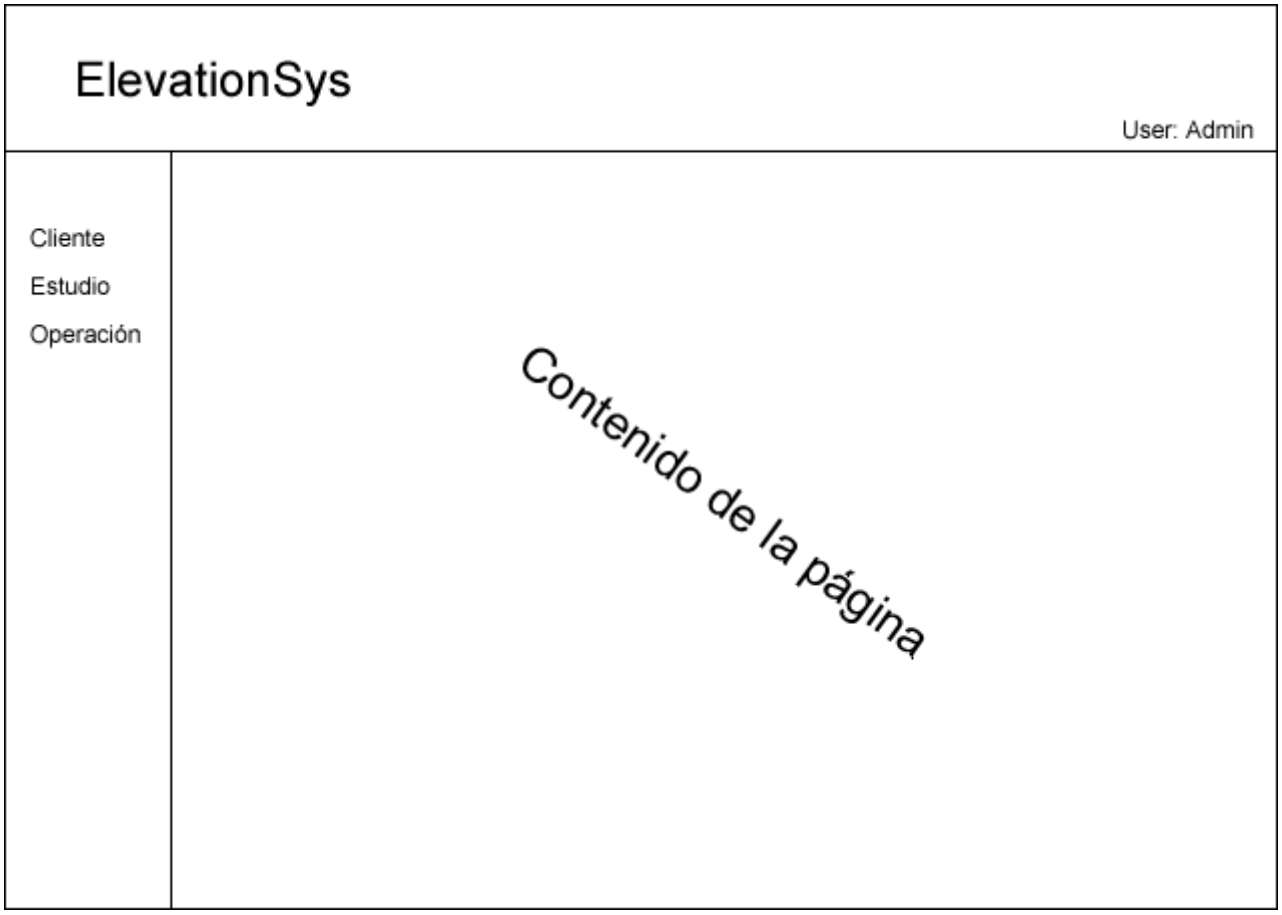
## *2.1. Descripción del sistema*

Será un BackOffice constituido de una barra de menú en la parte izquierda de la aplicación y mediante estos menús se podrá ir navegando por la web. Además las pantallas que estén relacionadas tendrán un acceso rápido en forma de link, para manejarse de una forma mas dinámica.

En el lateral izquierda el menú contendrá en algunas secciones para una mejor interfaz de usuario.

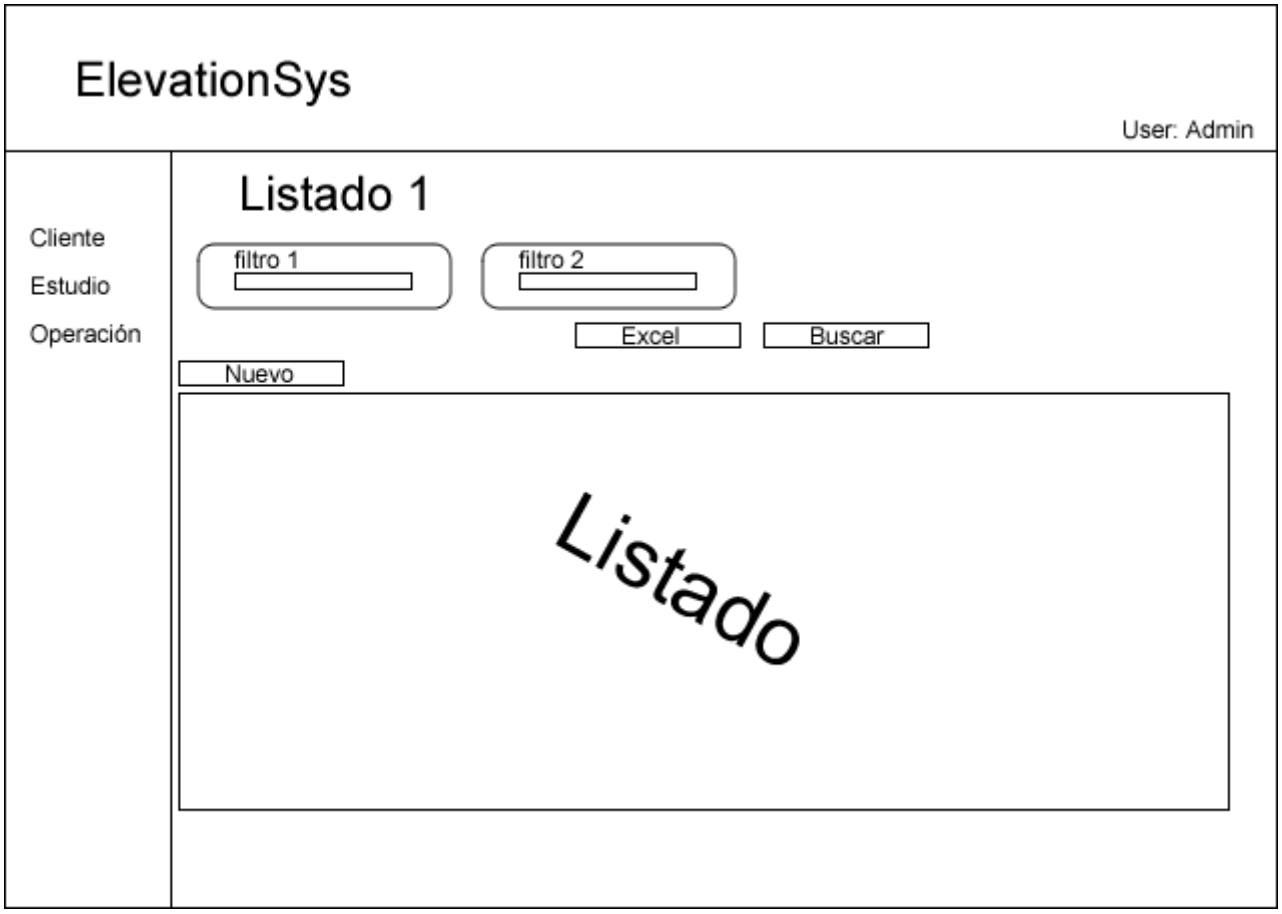
En la parte superior a la derecha se mostrará el nombre de usuario que esta registrado en el sistema. En el centro de la pantalla estarán las diferentes pantallas cambiando únicamente este apartado en todo el sistema.

Es obligatorio registrarse en el sistema para acceder a él. De toda la administración de usuarios se encargará el administrador del sistema



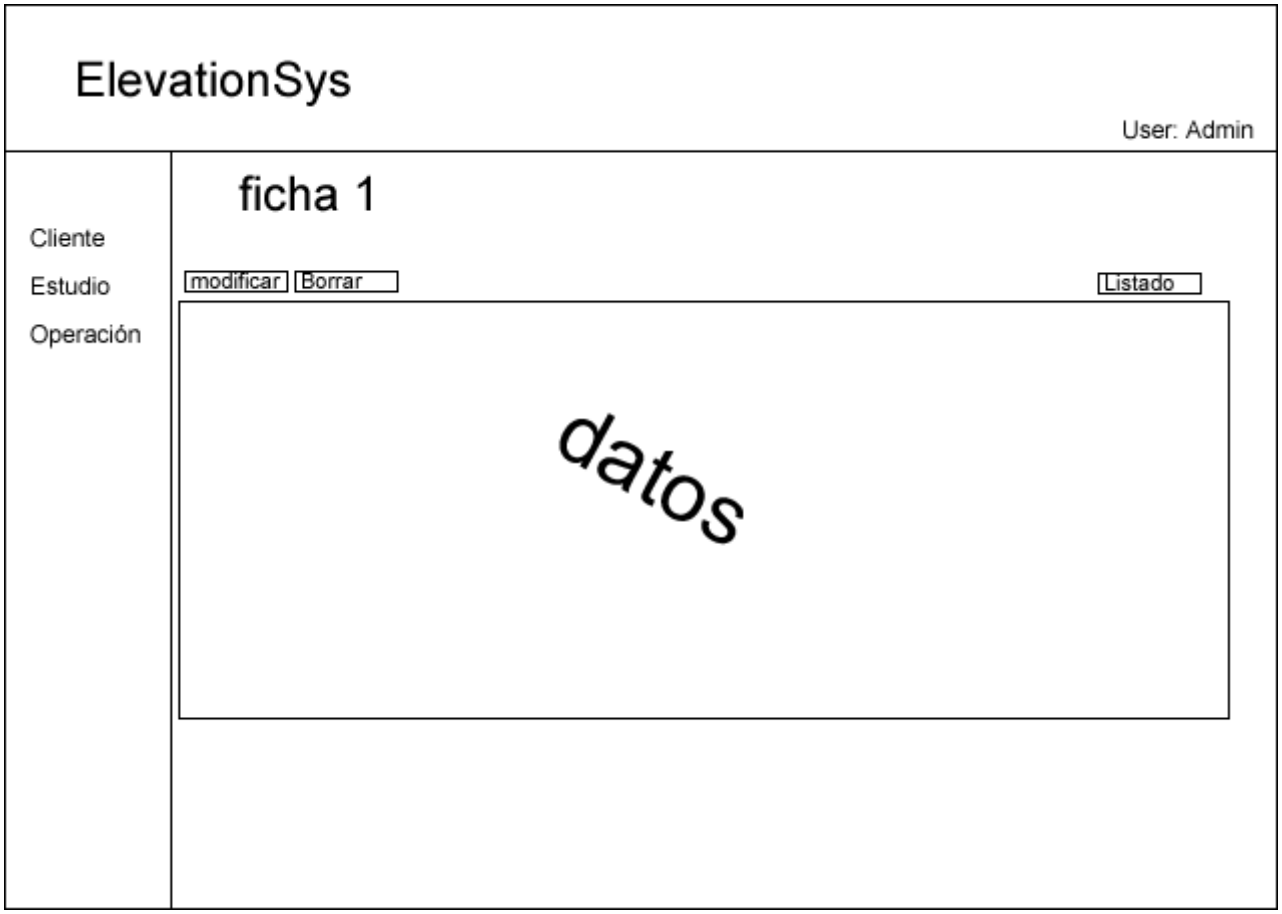

Las pantallas generalmente se dividirán en dos. En informes y en fichas.

Los informes contendrán un filtro y mostrarán los resultados permitiendo entrar en la ficha de cada uno. También cada informe podrá ser exportado a Excel. Si el usuario tiene los permisos necesarios, este podrá ver un botón que le permitirá crear una ficha nueva.



Las fichas tendrán los datos bloqueados. Si se quiere modificar estos datos habrá que hacer click en el botón de modificar. También se permitirá eliminar los datos si no hay restricciones.

Tanto la modificación como la eliminación se habilitarán para los usuarios que tengas los permisos correspondientes.



## *2.2. Recursos de Hardware y Software*

A continuación se detallan los recursos necesarios para poder usar el sistema backoffice y también para poder desarrollarlo.

**PC Cliente**

- Cualquier ordenador que sea capaz de correr el Interner Explorer 7

**Servidor**

- IIS
- SQL Server

**Entorno de Programación**

- Visual Studio .NET (ASP.NET)
- SQL Server
- Open office, para documentación
- MS Visio

**Hardware**

- Cliente (Req. mínimos)
  - Procesador P IV
  - 512 Mb de Ram
  - Disco duro 20 Gb
  - Teclado, Mouse
  - conexión a Internet

- Servidor (Req. Mínimos)
  - Procesador P IV
  - 2 GB RAM
  - Disco Duro 1 TeraByte
  - Acceso a Internet
  - Mouser, teclado

**Recursos Humanos**

- Director de Proyecto
- Programador Web (yo)

## *2.3. Análisis Costo Beneficio*

**Coste Material**

| | |
|---|---|
| Hosting | 90 € / Año |
| Conexión Internet | 50 € / Mes |
| Licencias MS Windows XP | 50 € / Ordenador |
| Visual Studio Profecional | 670 € / Licencia |
| MS Sql Server | 49 € / Procesador |

**Coste Personal**

| | |
|---|---|
| Jefe Proyecto | 50 € / Hora |
| Programador | 30 € / Hora |

**Tareas**

| | | |
|---|---|---|
| Est. Viabilidad | 20 Hs | 1.000,00 € |
| Definición Proyecto | 35 Hs | 1.750,00 € |
| Desarrollo Web | 50 Hs | 1.500,00 € |
| Testing | 20 Hs | 600,00 € |
| Documentación | 25 Hs | 1.250,00 € |
| **TOTAL** | **150 hs** | **6.100,00 €** |

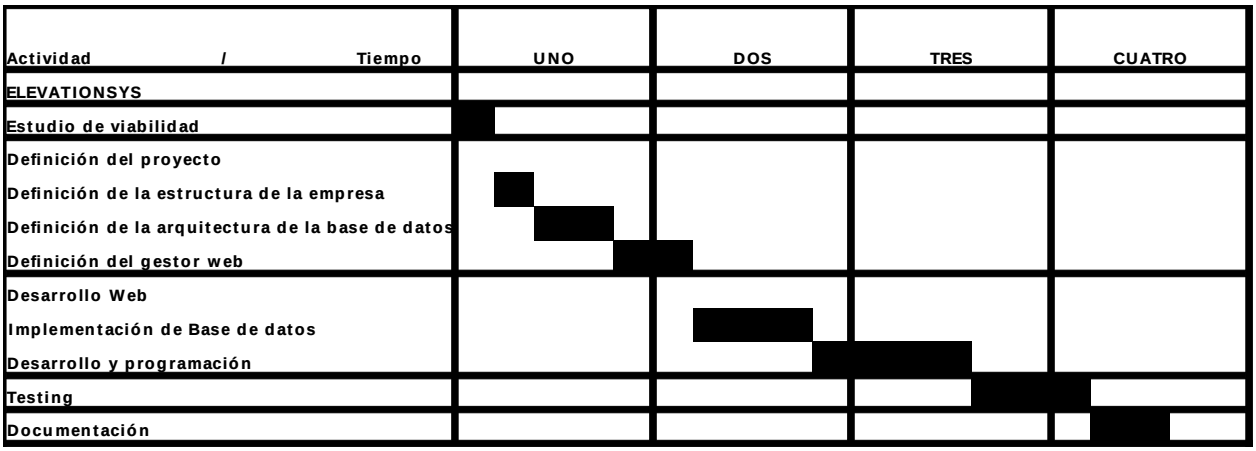
**Presupuesto Final**

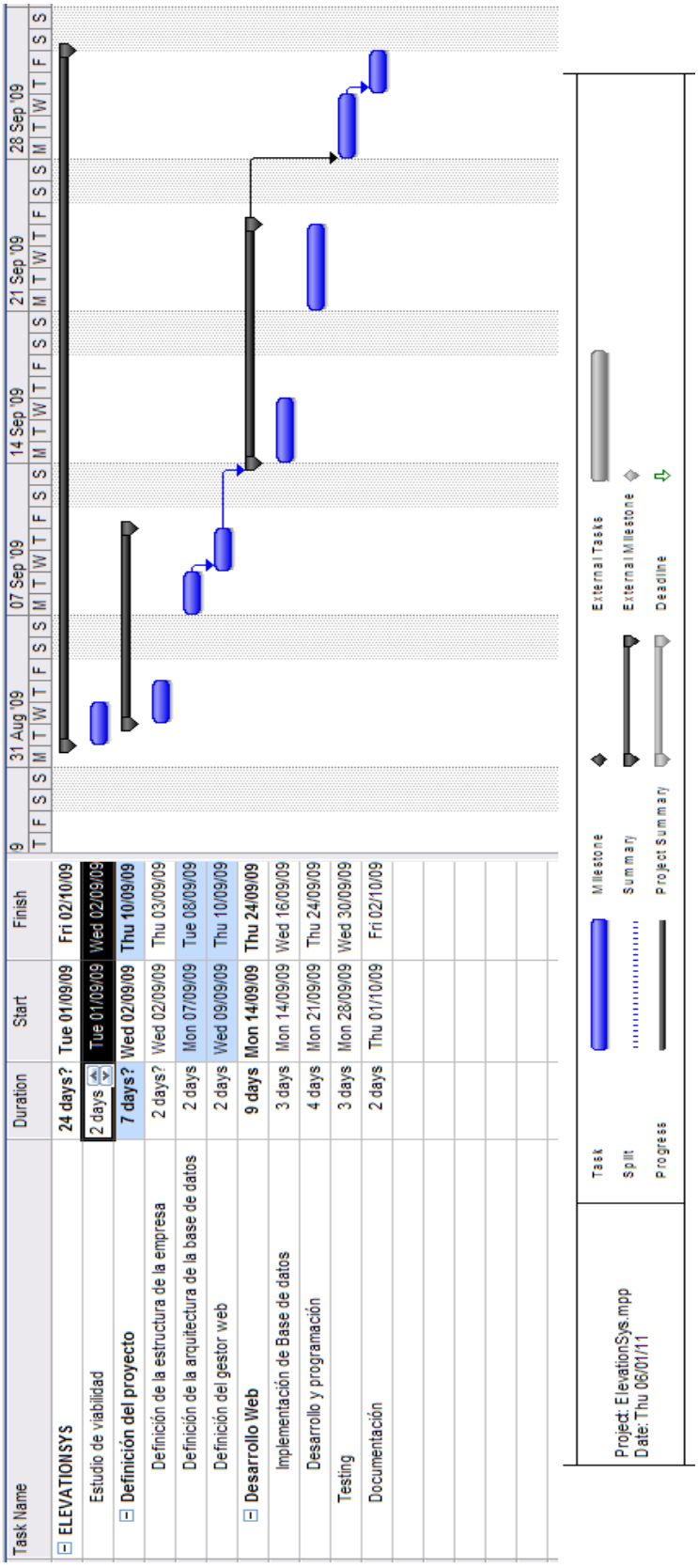
| | |
|---|---|
| Coste Material | 917,00 € (Licencias) |
| Desarrollo Software | 6.100,00 € |
| Total | 7.178,00 € |

- Sin Iva
- Presupuesto Válido hasta el 01/03/2011

## *2.4. Planificación*

Tiempo necesario para llevar acabo el proyecto.




Task Name
Duration
Start
Finish
ELEVATIONSYS 24 days? Tue 01/09/09 Fri 02/10/09
Estudio de viabilidad 2 days Tue 01/09/09 Wed 02/09/09
Definición del proyecto 7 days? Wed 02/09/09 Thu 10/09/09
Definición de la estructura de la empresa 2 days? Wed 02/09/09 Thu 03/09/09
Definición de la arquitectura de la base de datos 2 days Mon 07/09/09 Tue 08/09/09
Definición del gestor web 2 days Wed 09/09/09 Thu 10/09/09
Desarrollo Web 9 days Mon 14/09/09 Thu 24/09/09
Implementación de Base de datos 3 days Mon 14/09/09 Wed 16/09/09
Desarrollo y programación 4 days Mon 21/09/09 Thu 24/09/09
Testing 3 days Mon 28/09/09 Wed 30/09/09
Documentación 2 days Thu 01/10/09 Fri 02/10/09
31 Aug '09
07 Sep '09
14 Sep '09
21 Sep '09
28 Sep '09
Project: ElevationSys.mpp
Date: Thu 06/01/11
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

## *2.5. Diagrama MS Project*

## *2.6. Evaluación de Riesgos*

El BackOffice esta pensado para abarcar todo el procedimiento de la fábrica, desde que se hace el pedido desde el cliente hasta que se entrega la máquina.

1. **Incumplimiento de las funcionalidades de la empresa**

El sistema no cumple con las necesidades de la empresa y no refleja el estado de la empresa a través del sistema.

- **Solución**:
  El análisis funcional del sistema sera basado en cada usuario cumplimentando todas las funciones que hace este usuario e implementandolas en el sistema

2. **Inadaptación de los usuarios**
   Una vez implementado el sistema pude que los usuarios no estén familiarizados a él y no hagan uso del mismo.
   - **Solución:**

El sistema reemplazará el papel de la empresa y se utilizaran cada vez menos el papeleo haciendo imprescindible usar el sistema para ver el trabajo que hay.

3. **Mala introducción de datos**
   Al crear nuevos datos puede darse el caso que los datos no estén bien introducidos y eso cree incongruencia y un mal reflejo de los datos en el sistema.
   - **Solución**:

A cada usuario se le enseñara de manera previa como funciona el sistema y se hará uso de un manual de usuario para poder consultarlo en caso de dudas

4. **Poca Flexibilidad del Sistema**
   Al tener los usuarios tan definidos y permisos específicos a cada uno de ellos puede darse el caso que un usuario tenga poca libertad a la hora de usar el sistema
   - **Solución**:
     En el caso de que un usuario necesite usar el sistema de una forma distinta a la cual fue pensado y tenga la necesidad de utilizar una funcionalidad que no podía utilizar, simplemente se le agregará el permiso correspondiente para que lo pueda utilizar.

También existen problemas externos al Software

1. **Conexión a Internet**

En el caso de que no haya acceso a Internet no se podrá utilizar el sistema.

- **Solución**

Hay que asegurarse de que el proveedor de internet tenga un buen compromiso con la empresa y se pueda garantizar de la mejor manera posible la conexión.

2. **Problemas de saturación del servidor**

Puede darse el caso que hayan consultas al servidor muy pesadas o que se conecten demasiados usuarios al mismo tiempo y saturen al servidor.

- **Solución**

Las consultas pesadas, que gasten muchos recursos en el servidor, se intentaran ejecutar en horarios que no se utilice mucho el sistema. En el caso de que hayan muchos usuarios, el sistema es escalable por lo tanto se puede agregar mas memoria RAM o bien agregar otro servidor para que trabajen en paralelo.

3. **Perdida de información**

   En caso de corte de luz y que un disco duro falle, se puede perder información

   - **Solución**

     Se harán copias de seguridad diarias de toda la información del sistema, para que la perdida sea la menor posible.

## *2.7. Aspectos Legales*

Debido a que se estará trabajando con los datos personales de clientes y proveedores se tiene que tener muy en cuenta el ámbito legal de confidencialidad de datos, por ello se ha implantado un sistema de seguridad para evitar posibles problemas futuros en este aspecto.

Lo que se ha pensado para cumplir este aspecto de seguridad es implementar sistemas de accesos, utilizando backups, a parte de sistemas de identificación y autentificación que ya se tiene.

Por este motivo el contenido de este portal se regirá por el LOPD, ley orgánica de protección de datos, ley 15/1999 del 13 de Diciembre que tiene por objetivo proteger los datos de carácter personal registrados en cualquier soporte que los haga susceptibles.

Con ello se intentará mantener tanto la integridad de los datos como su disponibilidad y como su confidencialidad.

# 3. Modelo de desarrollo y Herramientas utilizadas.

## *3.1. Metodología utilizada.*

La metodología que se utilizará para llevar a cabo este proyecto será la metodología de ciclo estándar de análisis y diseño estructurado, una metodología de ciclo de vida lineal.

El motivo de escoger esta metodología es porque se adapta al sistema ya que se puede ir entregando por etapas o fases, las cuales van sucediendo de manera lineal. Una vez superada una etapa se podrá seguir con la siguiente aunque hayan etapas que puede ser independientes, su implementación puede realizarse por separado.

Las fases que se especifican a continuación son las que se llevara a cabo.

1. Estudio de viabilidad
2. Análisis de requerimientos
3. Diseño de la empresa
4. Desarrollo
5. Pruebas, integración
6. Instalación y evaluación del sistema
7. Documentación

Estas son las fases que se respetarán en el desarrollo.

Definición de fases

Fase1: **Estudio de viabilidad**

En esta fase se llevará a cabo el análisis que nos permitirá evaluar las garantías dellevar a cabo con éxito el proyecto, aquí se evaluarán los costes y beneficios y la planificación del mismo.

Fase 2: **Análisis de requerimientos**

a. **Requerimientos funcionales:** En esta fase de describirán todos los requerimientos del sistema a realizar.
b. **Requerimientos no funcionales**: En esta fase se describirán todos los requerimientos no funcionales a realizar

Fase 3: **Diseño de la empresa**

a. **Diseño de la estructura de la Base de Datos**:aquí se llevara a cabo el desarrollo de la base de datos del sistema. Creando las tablas y relaciones necesarias para representar la empresa.
b. **Diseño de la aplicación Web**: en este apartado se crearán los módulos y objetos que formarán parte del sistema

Fase 4: **Desarrollo**

En esta fase se implementará todo lo que se diseño en la etapa anterior tanto en la base de datos como en la parte de programación del sistema.

Fase 5: **Pruebas e integración**

Se mirarán que todas las pantallas funcionen correctamente y también que el ciclo de los flujos cierren perfectamente. También se probarán distintos navegadores.

Fase 6: **Instalación y evaluación del sistema**

Se realizará la publicación del backoffice y se mirarán si los objetivos planteados se cumplieron.

Fase 7: **Documentación**

Se llevará acabo el informe final que contenga todo lo necesario para representar este proyecto.

## ***3.2. Especificación de las herramientas a utilizar.***

A continuación se da una pequeña explicación sobre las herramientas y lenguajes utilizados en el proyecto.

El lenguaje ASP.NET se eligió porque es el lenguaje que utilizo cada día. Si bien no es gratuito existen unos kit de desarrollos, los express edition, que si lo son. Las aplicaciones de desarrollo Microsoft son muy completas y el MSDN es de gran ayuda a la hora de sacarse dudas sobre cualquier cosa. Además ASP.NET tiene unas herramientas muy fáciles de implementar que se amoldaban muy bien a la arquitectura de la empresa y cualquier representación Web de un BackOffice.

Bajo ASP.NET se implemento el kit de desarrollo AJAX que implementado con las aplicaciones ASP de microsoft le dan una fluidez especial al sistema, haciéndose ver mas como un formulario de windows y no como un formulario web.

Para la interfaz del sistema se han utilizado tanto hojas de estilo (CSS), como las “Style Sheet” de microsoft (SKIN) que se implementan mejor con los componentes que ellos ofrecen. Entre estas dos interfaces se puede definir todo el aspecto de la aplicación y en cualquier momento cambiando cualquiera de estos valores podremos cambiar la apariencia de la aplicación sin que deje de ser la misma.

Como servidor he escogido el Internet Information Server (IIS) ya que es compatible con Microsoft .NET, asi como SQL Server como gestor de Base de datos por ser todo compatible con la tecnología Microsoft, además es lo que utilizo diariamente en el trabajo.

Las pantallas de las páginas Web tanto como la programación las haré con Microsoft Visual Web Development 2005 Express edition. Las consultas y todo lo que este relacionado con la base de datos sera hecho con Microsoft SQL Server 2000. Los diagramas UML será hecho con Microsoft Visio y toda la documentación, al contrario del resto de herramientas, sera hecha con Open Office, porque es gratuito.

Definición de herramientas

1. **Servidos IIS**: Internet Information Services o IIS es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS. Este servicio convierte a una PC en un servidor web para Internet o una intranet, es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente. Los servicios de Internet Information Services proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor web seguro.

2. **Microsoft SQL Server**: Es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL. Soporte de transacciones, Escalabilidad, estabilidad y seguridad, Soporta procedimientos almacenados, Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente. Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información. Además permite administrar información de otros servidores de datos. Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en sus versiones 2005 y 2008 pasa a ser el SQL Express Edition, que se distribuye en forma gratuita. Es común desarrollar completos proyectos complementando Microsoft SQL Server y Microsoft Access a través de los llamados ADP (Access Data Project). De esta forma se completa la base de datos (Microsoft SQL Server), con el entorno de desarrollo (VBA Access), a través de la implementación de aplicaciones de dos capas mediante el uso de formularios Windows. En el manejo de SQL mediante líneas de comando se utiliza el SQLCMD. Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor sólo está disponible para Sistemas Operativos Windows.

3. **Microsoft Visual Web Development Express Edition 2005**: Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. A partir de la versión 2005 Microsoft ofrece gratuitamente las Express Editions. Estas son varias ediciones básicas separadas por lenguajes de programación o plataforma enfocadas para novatos y entusiastas. Estas ediciones son iguales al entorno de desarrollo comercial pero sin características avanzadas.

4. **HTML**: siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. Las pantallas ASP.NET terminan siendo finalmente una pantalla HTML, por lo tanto es fundamental conocer este lenguaje también.

5. **Javascript**: es un lenguaje de scripting basado en objetos sin tipo y liviano, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas.

# 4. Análisis de aplicaciones

## *4.1. Requerimientos no funcionales*

**Fiabilidad:**

En cualquier momento se ha de poder obtener el estado en el cual se encuentra la empresa. Cuantos productos hay en producción, cuantos pedidos se encuentran en marcha, cuantos pedidos se entregaron y retrasos.

- Buena capacidad para tolerar errores
- Buena capacidad para recuperarse de errores
- Capacidad para tolerar sobrecargas de volumen de información.
- Captura de excepciones.
- Contingencias para eventos de caída de sistema

**Tiempo de Respuesta:**

Garantizar la información para poder ser consultada y actualizada en cualquier momento sin que afecte el tiempo de respuesta en cada proceso del sistema.

- Los tiempo de respuesta tienen que ser aceptables en los procesos en linea del sistema.
- La velocidad tiene que ser estable para los usuarios del sistema.
- Optimizar la ejecución de procesos para disminuir la congestión del sistema.
- Tiempos aceptables máximos del sistema fuera de linea

**Seguridad:**

Se diferencia varios tipo de usuarios para que cada uno haga las funcionalidades que le corresponden.

- Permitir el registro de eventos del sistema empleando la información de la sesión de usuario (LOG).
- Permitir almacenamiento cifrado de datos determinados.
- Permitir canales cifrados de transmisión de datos para procesos específicos.
- Presentación procesos de autentificación de usuarios.
- Implementar una auditoría
- Guardar los cambios que se hacen en el sistema.

**Mantenibilidad:**

Intentar lograr que el mantenimiento del sistema sea lo mas natural posible. Empleando un modelo unificado de desarrollo y teniendo toda la documentación correspondiente del diseño del sistema.

- Desarrollar manual técnico de referencia de la aplicación.
- Seguir una metodología para la implementación del diseño del sistema.

**Escabilidad:**

En algún futuro el sistema ha de ser capaz de implementar módulos, componentes o

extensiones, presentan directrices en el diseño y la arquitectura.

- Se empleara módulos independientes
- Empleo en tecnología XML o Web Services para tener compatibilidad con otros sistemas.
- Diseñar el sistema compuesto por subsistemas para que agrupen una funcionalidad común.
- Las actualizaciones del sistema se harán solo del lado del servidor.

**Amigabilidad:**

La presentación del sistema en cuanto al al diseño gráfico y las facilidades de uso del sistema por parte del usuario final.

- El sistema tendrá que ser de fácil uso haciendo uso de un buen diseño de interfaz de usuario con criterios generales de diseño
- Mensajes en la interfaz claros
- colores
- tipo y tamaño de letra.

**Concurrencia de Usuarios:**

Permitir el acceso a múltiples usuarios a la vez a los servicios del sistema.

- Cada usuario tendrá una sesión en el servidor

## *4.2. Requerimientos Funcionales y casos de uso:*

A continuación se mostrarán los diferentes casos de uso que se utilizan para desarrollar el proyecto.

**Usuario:**

Un usuario es una persona que accede al sistema. Cada usuario tiene un tipo de usuario que lo define, pero igualmente todos los usuarios comparten las mismas funcionalidades básicas que son las siguientes.

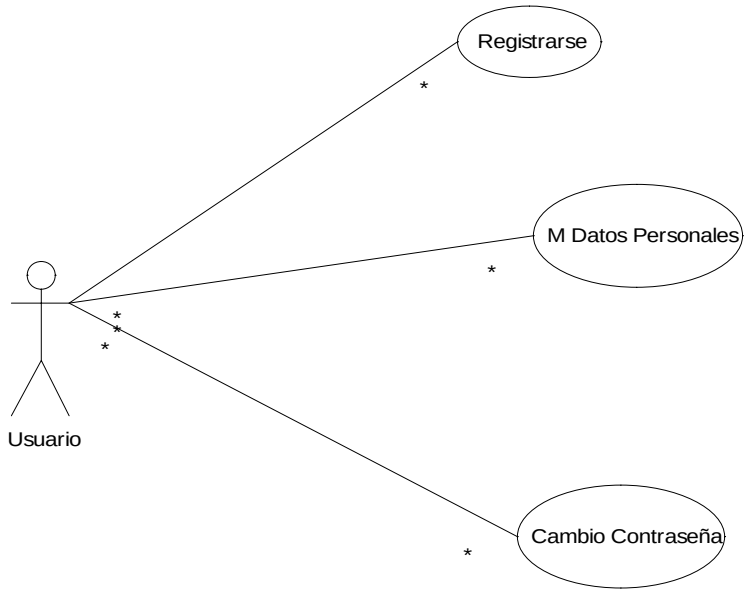



*Ilustración 1: Diagrama de Usuarios*

Dependiendo el tipo de usuario que sean, todos tendrán las mismas funcionalidades básicas.

1. **Registrarse:**

Cada usuario que quiera usar el sistema tendrá que registrarse. Al registrarse el sistema cargará sus datos y los permisos adaptando el sistema al usuario registrado y mostrando solo las funcionalidades que puede utilizar.

2. **Modificación de Datos Personales:**

Cada usuario podrá ver sus datos personales que tiene guardado el sistema y tendrá la libertad de modificarlos.

3. **Cambio de Contraseña:**

Un usuario registrado tendrá siempre la libertad de modificar su contraseña . Para ello tendrá que que introducir la contraseña anterior.

**Administrador:**

Un administrador tiene la función de administrar a los usuarios del sistema, los permisos y los flujos del sistema. Además de ello tiene permiso para acceder a cualquier pantalla del sistema y rienda libre sin ningún tipo de restricción.

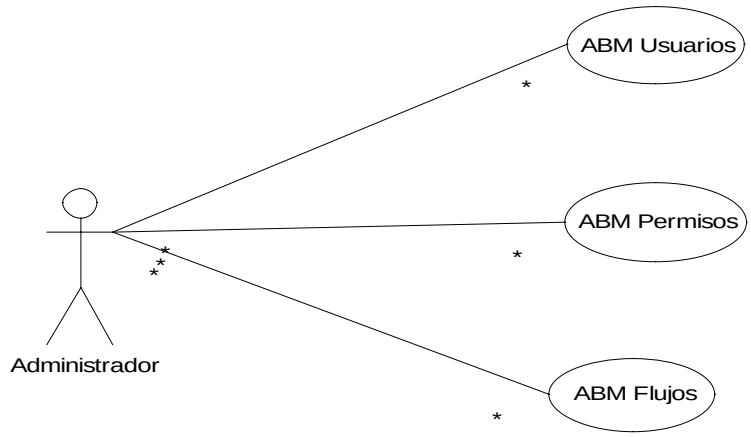



*Ilustración 2: Diagrama de Administrador*

1. **ABM Usuarios:**

El administrador es el encargado de administrar los usuarios del sistema. Es el único usuario que es capaz de ver a todos los usuarios, dalos de alta, modificar sus datos o darles de baja.

2. **ABM Permisos:**

El administrador es el encargado de administrar los permisos del sistema a los usuarios. Cada pantalla tendrá un permiso distinto y cada usuario tendrá los permisos que le correspondan para poder utilizar el sistema.

3. **ABM Flujos:**

El administrador es el encargado de definir los distintos flujos del sistema. Un flujo determina como evoluciona un pedido en el sistema, si en algún momento se cambia la metodología de trabajo en la empresa, quizás haya que alterar el funcionamiento del flujo.

**Comercial:**

Un comercial es el encargado de vender los productos del sistema. Para ello el sistema le ofrece la posibilidad de almacenar los clientes y los pedidos que vayan surgiendo. El comercial es el que inicia el proceso del sistema creando pedidos. Cuando un pedido es aceptado por un cliente es cuando el resto de la empresa comienza a funcionar.

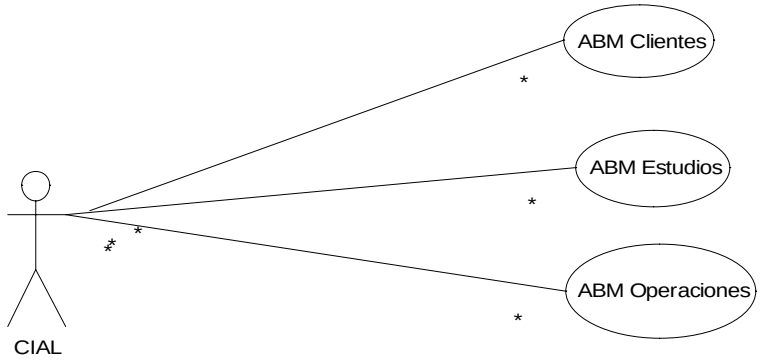



*Ilustración 3: Diagrama de CIAL*

1. **ABM Clientes:**

Un comercial podrá administrar los clientes del sistema. Se podrán dar de alta, modificar los datos o mismo darlos de baja. Cada cliente tiene asociado un comercial responsable.

2. **ABM Estudios:**

Un comercial podrá administrar los estudios del sistema. Se podrán dar de alta, modificar los datos o mismo darlos de baja.

3. **ABM Operaciones:**

Un comercial podrá administrar las operaciones del sistema. Se podrán dar de alta, modificar los datos o mismo darlas de bajas.

**Oficinas Técnicas:**

Oficinas técnicas (OT) es el encargado de definir técnicamente el pedido creado por el comercial y ver si es viable. Gestiona las compras de la empresa y crea las fichas técnicas de la maquina pedida con especificaciones técnicas.

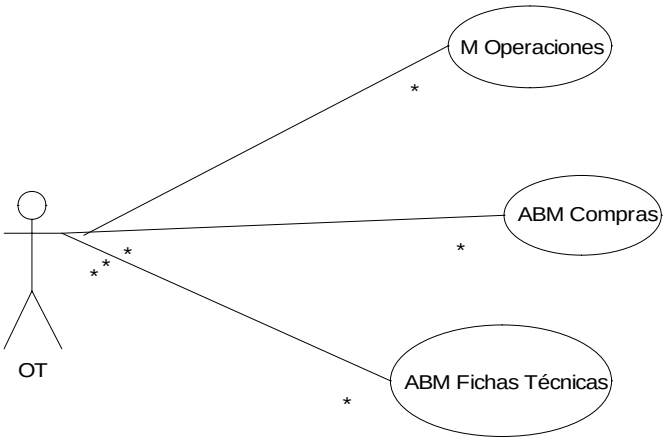



*Ilustración 4: Diagrama Oficinas Técnicas*

1. **Modificación de Operaciones:**

Un usuario OT podrá modificar los datos de la operación que han venido determinadas por el comercial.

2. **ABM Compras:**

Un usuario OT podrá administrar las compras del sistema. Se podrán dar de alta, modificarlas o darlas de baja. Pero una vez pedidas no se podrán tocar sin retrasar el estado de la operación.

3. **ABM Fichas Técnicas:**

Un usuario OT es el encargado de generar las fichas técnicas de las operaciones donde se define técnicamente el producto. Se podrán dar de alta, modificar los datos o mismo darlas de baja. Pero una vez hecho el pedido no se podrán modificar a no se que se retrase el estado de la operación.

**Planificación:**

Un usuario de planificación es el que organiza la parte de taller de la empresa y da la fecha de entrega estimada al cliente de cuando se hará la entrega. También es el encargado de hacer las compras, gestionar los proveedores y productos de la empresa.

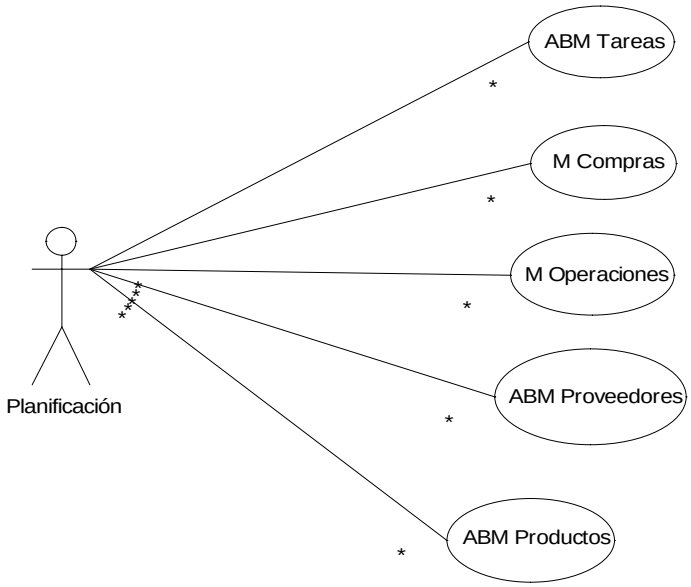



*Ilustración 5: Diagrama de Planificación*

1. **ABM Tareas:**

Un usuarios de planificación es el encargado de definir las tareas para llevar a cabo el producto. Para ello podrá dar de alta, baja o modificar las tareas del sistema para llevar a cabo la realización del pedido del cliente. Las tareas marcarán la planificación del sistema y el tiempo que tardará en terminar el pedido. Cada tarea tendrá una fecha de planificación estimada y otra real que serán las horas que los usuarios de taller marquen en ellas.

2. **Modificación de las Compras:**

El usuario de planificación es el encargado de pedir las compras al proveedor. Antes de hacer el pedido podrá modificar los pedidos. Una vez hecho el pedido, la operación pasara al siguiente estado del flujo y bloqueara varias opciones para no alterar la configuración con la que fue pensada. En el caso que se haya cometido un error o sea obligatorio modificar la operación, se podrá retroceder el estado de la operación.

3. **Modificación de las Operaciones:**

Un usuario de planificación podrá determinar la fecha de entrega de la maquina y hacer las ultimas modificaciones de la operación antes de pedir las compras.

4. **ABM Proveedores:**

Un usuario de planificación es el encargado de administrar los proveedores del sistema. Los podrá dar de alta, modificar sus datos o bien darlos de baja.

5. **ABM Productos:**

Un usuario de planificación es el encargado de administrar los productos que existen en el sistema

para confeccionar los pedidos. Para ello podra darlos de alta, modificar sus datos o bien darlos de baja.

**Taller:**

Un usuario de taller es el encargado de montar las maquinas. Para ello seguirá las tareas que le fueron asignadas desde planificación para la generación de la maquina. Un usuario de taller marcará su horario de entrada y salida contra una tarea para determinar su trabajo.

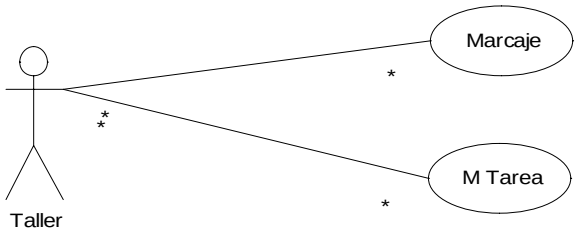



*Ilustración 6: Diagrama Taller*

1. **Marcaje:**

Un usuario de taller al entrar al sistema hará un marcaje de horario de entrada y horario de salida contra una tarea. De esta manera quedarán registradas las horas trabajadas en cada tarea que en un principio deberían coincidir con las estimadas en planificación.

2. **Modificación de Tareas:**

Un usuario de Taller tendrá un acceso limitado a las tareas y a los datos que pueden modificar. Mas específicamente podrán añadir notas explicando el trabajo realizado o añadir incidencias en el caso que haya pasado algún problema que retrase la realización de la tarea.

**Contabilidad:**

Un usuario de contabilidad accederá al sistema tan solo para relacionar ciertos aspectos contable. Este sistema no contendrá soporte contable.

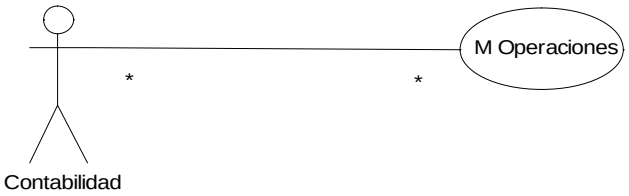



*Ilustración 7: Diagrama de Contabilidad*

1. **Modificación de Operaciones:**

Un usuario contable tendrá un acceso limitado al sistema. Tan solo modificará algunos datos de la operación relacionando la operación con un número de factura y las fechas de pago establecidas con el cliente.

**Contacto:**

Un contacto es un usuario externo al sistema perteneciente a un cliente. Su única función es la posibilidad de ver como va creciendo su pedido y darle la confianza necesaria para darle mayor transparencia con sus pedidos y ganar mayor confianza.

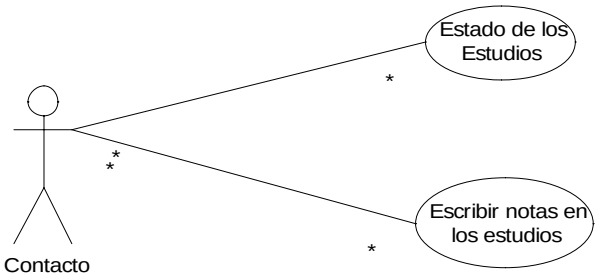



*Ilustración 8: Diagrama de Contacto*

1. **Estado de los estudios:**

Un contacto tendrá tan solo acceso a los estudios que le pertenezcan. Pudiendo ver el estado del estudio y el avance de su pedido.

2. **Escribir notas en los estudios:**

Un contacto puede escribir notas sobre cosas que el crea necesarias sobre su pedido.

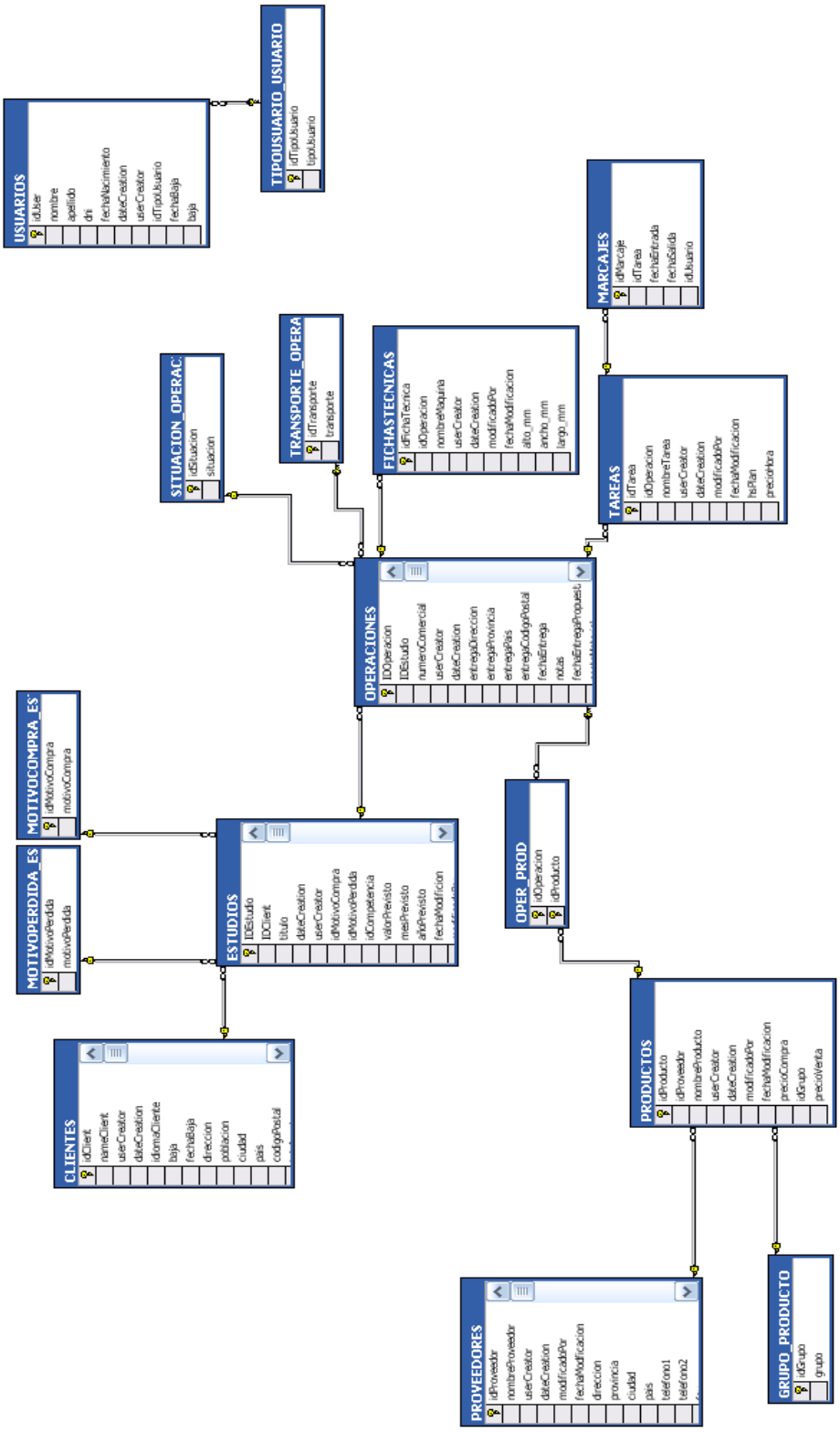
# 5. Diseño

## *5.1. Diseño de la base de datos*

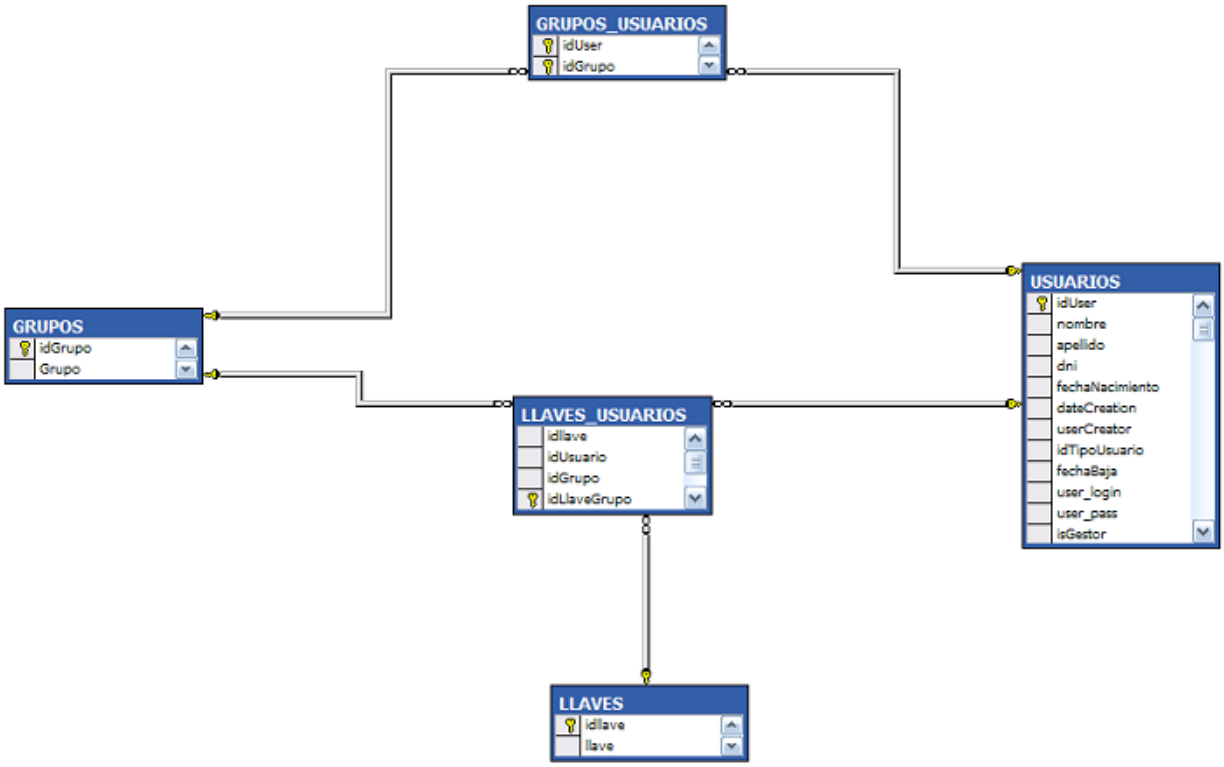
Para poder manipular y gestionar la base de datos con la que estamos trabajando se ha utilizado la herramienta SQL Server 2000, que es uno de los sistemas de gestión de base de datos mas utilizado.

El siguiente diagrama de Base de datos muestra el esquema general de la empresa. En la empresa partimos siempre desde un cliente y de este se van abriendo estudios, operaciones, fichas técnicas, tareas, etc. Cada una de estas partes pertenece a otra anterior y así formando grupos y subgrupos. De esta manera una operación siempre pertenecerá a un estudio que fue hecho para un cliente en particular.

**Diagrama de relación de base de datos de ElevatinSys.**



### Diagrama de las llaves de ElevationSys

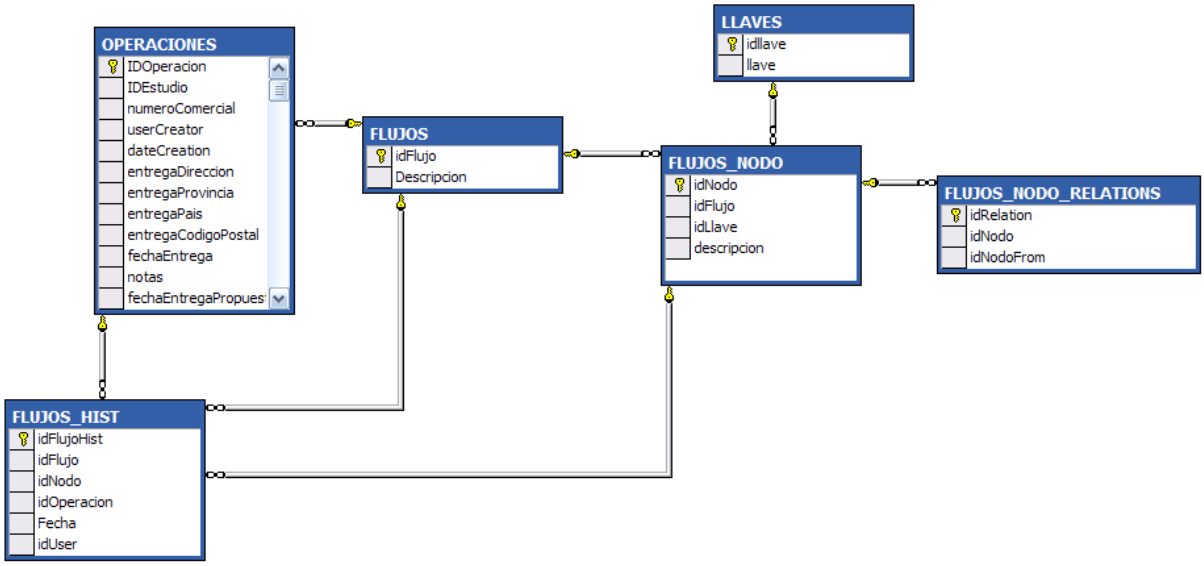



En este diagrama podemos ver el grado de llaves que da permiso a los usuarios para acceder a diferentes partes del sistema. Básicamente cada vez que se quiera restringir acceso a un lugar, se le asigna una llave. Y si el usuario pertenece a un grupo con la llave o si tiene la llave directamente, entonces podrá acceder a este apartado de la empresa.

Se crearon grupos para poder agilizar la asignación de permisos dependiendo en el departamento en el cual trabajen, Comercial, Oficinas Técnicas, etc. Al asignarle a un usuario un grupo, le estamos dando todas las llaves que este grupo contiene. Además se puede hacer excepciones y darle individualmente a cualquier usuario una llave determinada. En la tabla “GUPOS_USUARIOS” tenemos asignados los usuarios a cada grupo.

La tabla “LLAVES_USUARIOS” contiene a todos los grupos o usuarios que tienen las llaves. Basta con agregar un usuario o grupo asociado con una llave y este grupo o usuario tendrá el acceso.

## Diagrama de los Flujos de ElevationSys




Este diagrama representa un diagrama de flujos de dependencia. Cada Flujo tienen nodos y cada nodo tiene una relación de dependencia. A su vez, cada nodo tiene una llave para que solo pueda ser accedido por las personas con permisos a ellos.

La tabla “FLUJOS_NODO_RELATION” contiene la relación que hay entre los nodos. Si el valor *idNodoFrom* es “null”, quiere decir que ese nodo es el que comienza el flujo. Luego, cada nodo contiene de que nodo depende. De esta manera si un nodo es aprobado, podremos saber cuales son los que dependen de él y cual es el siguiente estado a seguir.

Cada vez que se pasa un nodo, se agrega en la tabla “FLUJOS_HIST”. Aquí estarán los ***nodos pasados*** de cada operación. Así que para saber en que estado estamos, hay que preguntar que idNodoFrom esta en la tabla “FLUJOS_HIST” que no este en dicha tabla.  De esta manera obtenemos los ***nodos actuales*** del flujo. Los ***nodos futuros*** son los que no tienen ningún idNodoFrom en la misma tabla.

Se pueden definir diferentes flujos. La idea de cada uno de ellos es que representen el funcionamiento que tiene la empresa. Cada vez que se define un flujo, es necesario definir todos sus nodos, sus dependencias y asignarles las llaves que creamos necesarias. En nuestro caso tenemos tres tipos distintos de flujos, oferta, pedido y demo.

## *5.2. Diseño de la interfaz gráfica*

Este sistema BackOffice esta pensado para ser lo mas intuitivo posible para todo tipo de usuario. Con ello se quiere lograr pantallas claras sin mecanismos raros a la hora de ponerse a trabajar. Es por ello que en cada pantalla se intenta de respetar básicamente el mismo formato para que no se tenga que estar buscando los botones. Otra cosa que se intento tener en cuenta es de no llenar la pantalla con información de mas y que sea visualmente agradable.

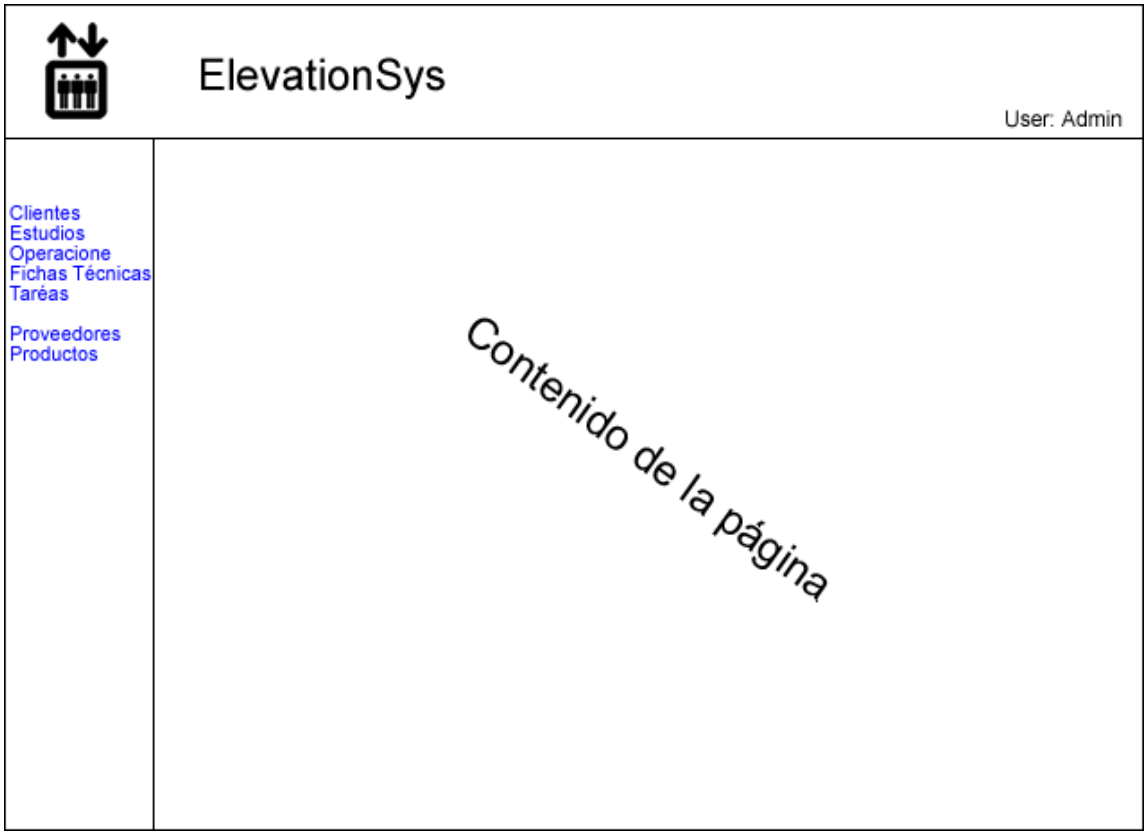
**Características:**

El sistema esta preparado para tener un funcionamiento bajo cualquier navegador actual de hoy en día. Tanto en Microsoft Explorer, como Mozilla o Chrome.

La configuración recomendada para trabajar es de 1280x800 y una paleta de 16 bits de colores.

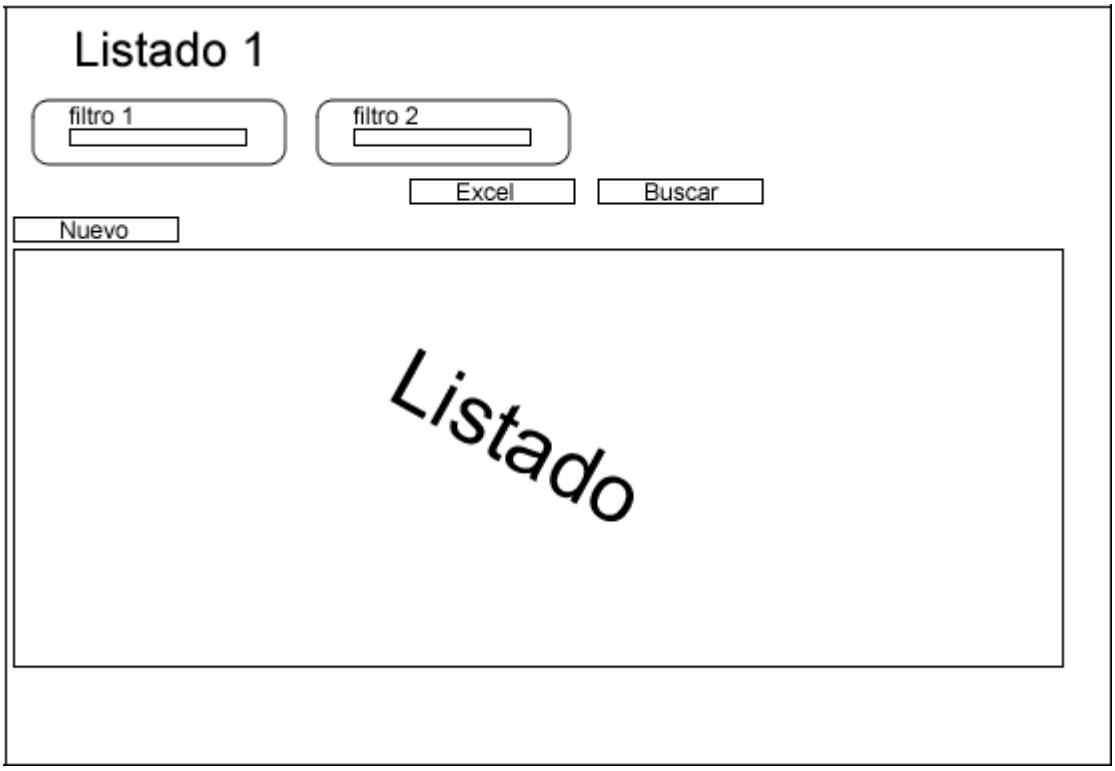
**Descripción:**

Nuestro sitio contiene una “Master Page”. La característica de las "Master Pages" nos proporciona la habilidad de definir una estructura y unos elementos de interfaz comunes para nuestro sitio, tales como la cabecera de página o la barra de navegación, en una ubicación común denominada "master page", para ser compartidos por varias páginas del sitio. Ésto mejora la mantenibilidad de nuestro sitio y evita la duplicación innecesaria de código para estructuras o comportamientos del sitio que son compartidos.
La Master Page viene definida de la siguiente manera.



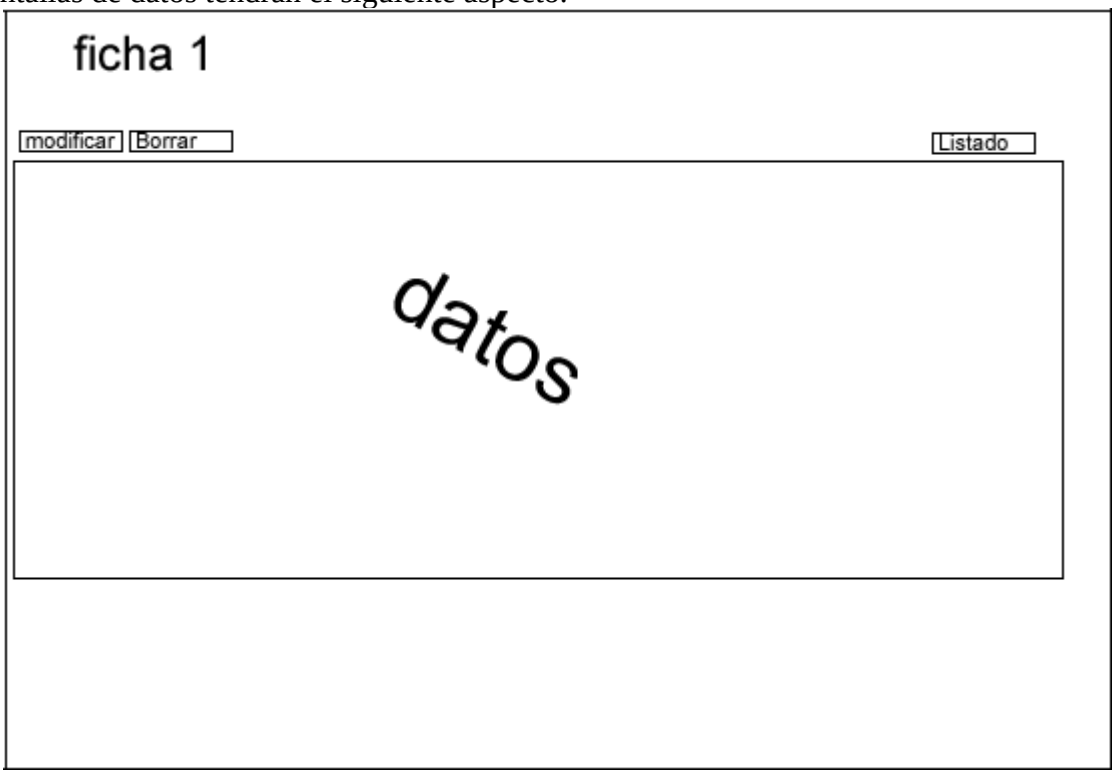
Donde en la parte superior viene definido el logo de la empresa junto con el nombre de usuario que esta conectado. En el lateral izquierdo están todos los links a cada parte del sistema. Cabe destacar que estos links pueden variar según quien este conectado ya que cada usuario tendrá diferentes permisos y por lo tanto tendrá diferentes accesos a cada pantalla.
El resto de la pantalla es donde se pondrá el contenido y es aquí donde iremos haciendo nuestra aplicación. Todas las pantallas heredaran esta “Master Page”.

Luego las pantallas se dividirán prácticamente en dos tipos. En informes y pantalla de datos. Los informes tendrán la siguiente apariencia



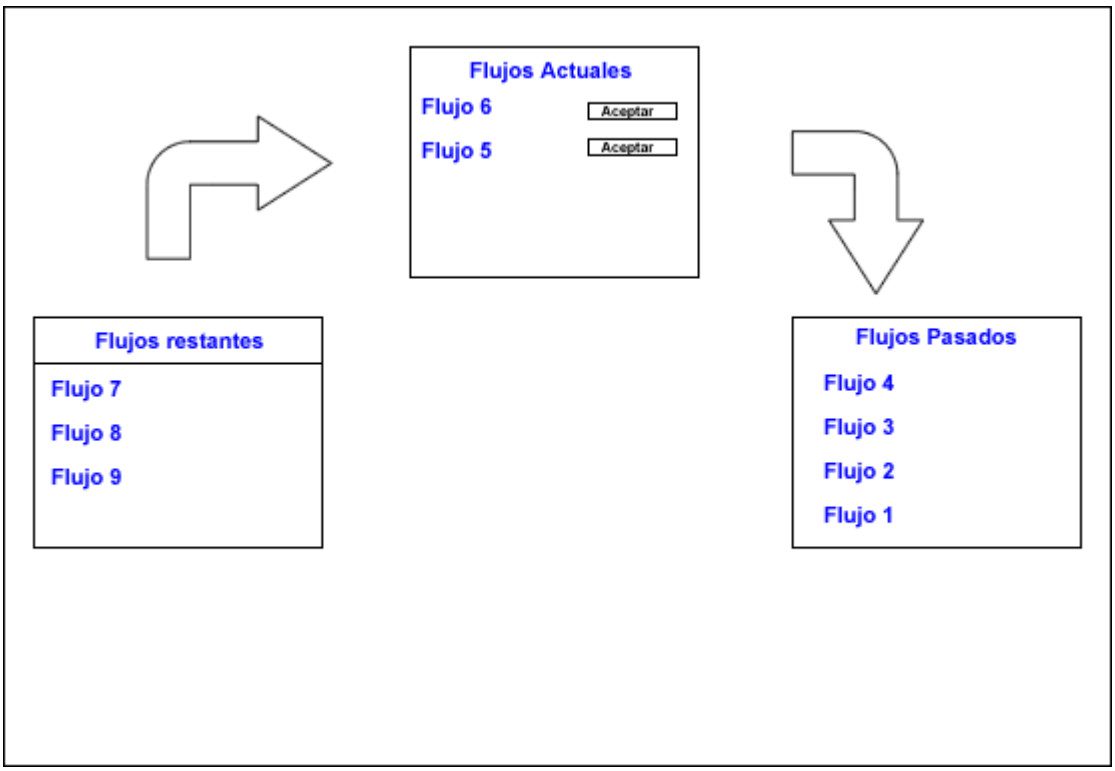

Cada informe será dinámico por lo tanto la cantidad de filtros variará y dependerá de cada uno. Nuevamente dependiendo de los permisos de usuarios se mostrará o no el botón de nuevo para crear una ficha nueva en el informe. Inicialmente los informes aparecerán vacíos hasta que se filtre. Una vez filtrado o en cualquier momento se podrá exportar el resultado de la información a Excel. Cada filtro a su vez dará distintos tipos de posibilidades dependiendo el tipo de dato que se trate. Si es un tipo de dato texto se podrá incluir las opciones de “%” para mejorar la búsqueda. Los tipo numéricos se podrá elegir tanto igual, mayor o menor al nro deseado. Y en todas ellas también existirá la opción de negarlos.

Las pantallas de datos tendrán el siguiente aspecto.




En las pantallas de datos se verá la información detallada de cada ficha. Dependiendo de los permisos de cada usuario cada uno de ellos podrá o no modificar o eliminar dichas fichas.
Para poder editar dicha información se deberá presionar el botón “modificar”. A partir de aquí la pantalla pasará en modo de edición y liberará todos los datos para poder modificarlos. Los botones de “modificar” y “borrar” cambiaran a llamarse “Guardar” y “Cancelar” respectivamente. Si no se desean cambiar los cambios basta con presionar el botón “Cancelar” y de lo contrario el botón “Guardar”. Cada pantalla de datos a su vez tendrá un botón de informe el cual les permite volver al informe de donde estaban.

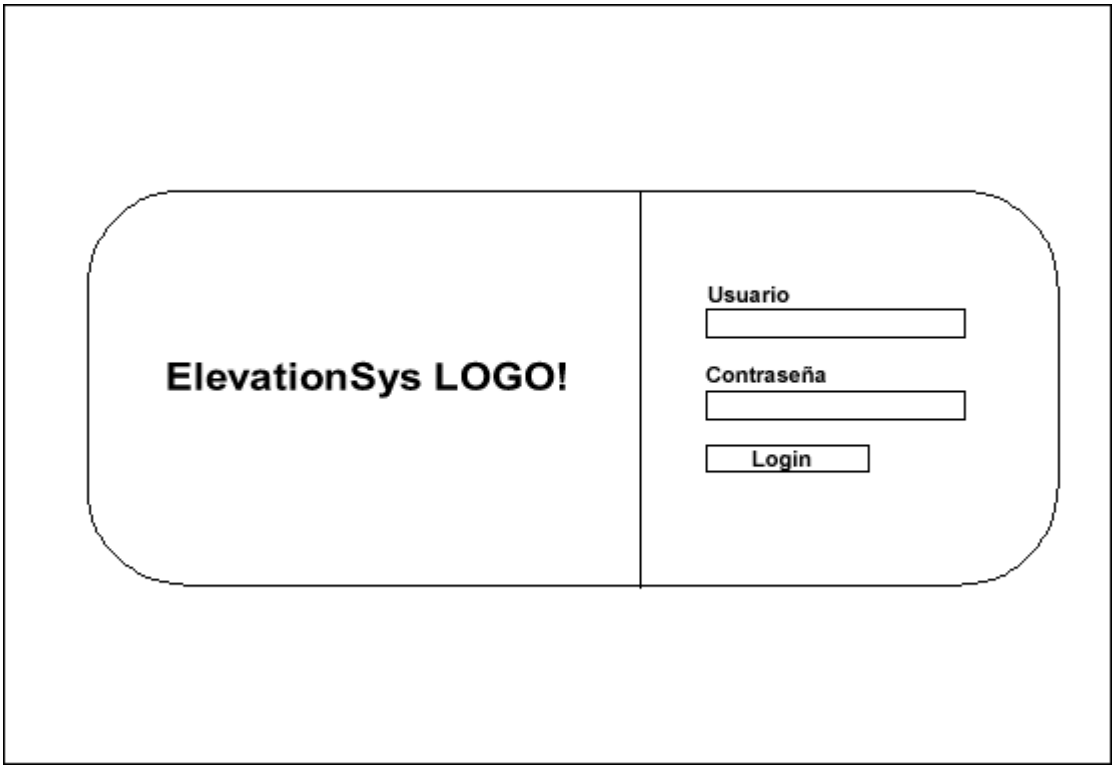
La pantalla de flujo tendrá el siguiente aspecto.




En la parte inferior a la izquierda se podrán ver los estados restantes. En la parte inferior a la derecha estarán los estados pasados. En la parte superior estarán los estados actuales esperando a ser terminados y aceptados para seguir con el flujo. Nuevamente el botón de aceptar solo le aparecerán a las personas que tengan el permiso para dicha acción.

Cada pantalla mencionada estará dentro de la “Master Page” por lo tanto mantendrá el aspecto principal siempre igual.

Por ultimo nos queda la pantalla de login. Esta pantalla será independiente al resto y no estará dentro de una “Master Page”. El aspecto de la pantalla de Login será el siguiente.




Aquí es donde se podrán autentificar los usuarios. Al ser un backOffice si pierden la contraseña o tienen algún problema que se pongan en contacto con el administrador del sistema.

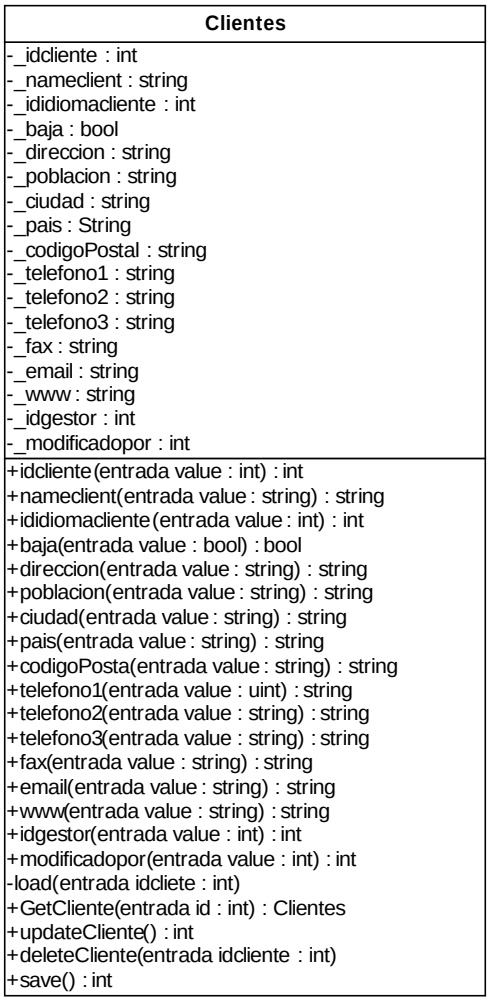
## *5.3. Diagrama de clases*

ElevationSys tiene una interfaz implementada en objetos. Dicha interfaz se encarga de la gestion de los datos sobre la base de datos y algunas funcionalidades mas. Prácticamente cada elemento que aparece en el sistema tiene un objeto detrás que lo gestiona. A continuación expongo dichas clases.

Estas Clases se adaptan a las tablas que ya hay en el sistema siendo una mera representación de lo que hay en ellas. Los métodos que tienen son justamente de carga, guardado y eliminación. Una vez que tenemos la clase instanciada, podemos trabajar con ella en el sistema.

En cada una de las clases siguientes, las propiedades serán todas privadas. Y para acceder a ellas habrán unos métodos públicos. Estos métodos en Visual Basic.NET tienen un nombre especial, se llaman “properties”. Un “property” es un miembro de la clase que actúa como intermediario entre una variable miembro de la clase. Esto se hace para no exponer las variables a otras partes del programa.

### Clase Clientes

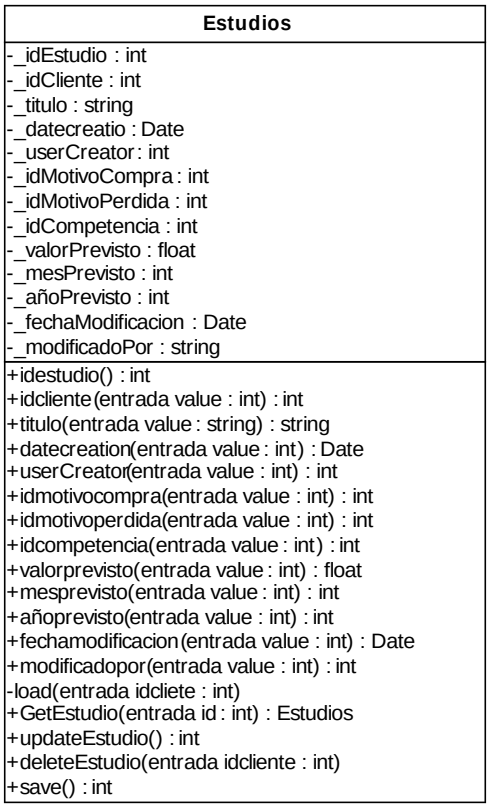



Explicación de los métodos.

- **idcliente**: es un property de lectura.
- **Namecliente**: es un property de lectura y escritura
- **ididiomacliente**: es un property de lectura y escritura
- **baja**: es un property de lectura y escritura
- **dirección**: es un property de lectura y escritura
- **poblacion**: es un property de lectura y escritura
- **ciudad**: es un property de lectura y escritura
- **pais**: es un property de lectura y escritura
- **codigoPostal**: es un property de lectura y escritura
- **telefono1**: es un property de lectura y escritura
- **telefono2**: es un property de lectura y escritura
- **telefono3**: es un property de lectura y escritura
- **email**: es un property de lectura y escritura
- **www**: es un property de lectura y escritura
- **idGestor** : es un property de lectura y escritura
- **modificadoPor**: es un property de lectura

- **Load**: es un método privado que tiene como entrada el identificador del cliente y hace la llamada a la base de datos cargando toda la información
- **GetCliente**: es el método publico para cargar el cliente
- **updateCliente**: Esta propiedad tiene como entrada cada propiedad de la clase y se encarga de actualizar el objeto en la base de datos, guardando así las modificaciones. Si el identificador que se le pasa es 0, entonces insertará un nuevo cliente.
- **DeleteCliente**: Borra el cliente de la base de datos.
- **Save**: una vez cargado el objeto, se hace el llamado a la base de datos para guardar el objeto en la tabla.

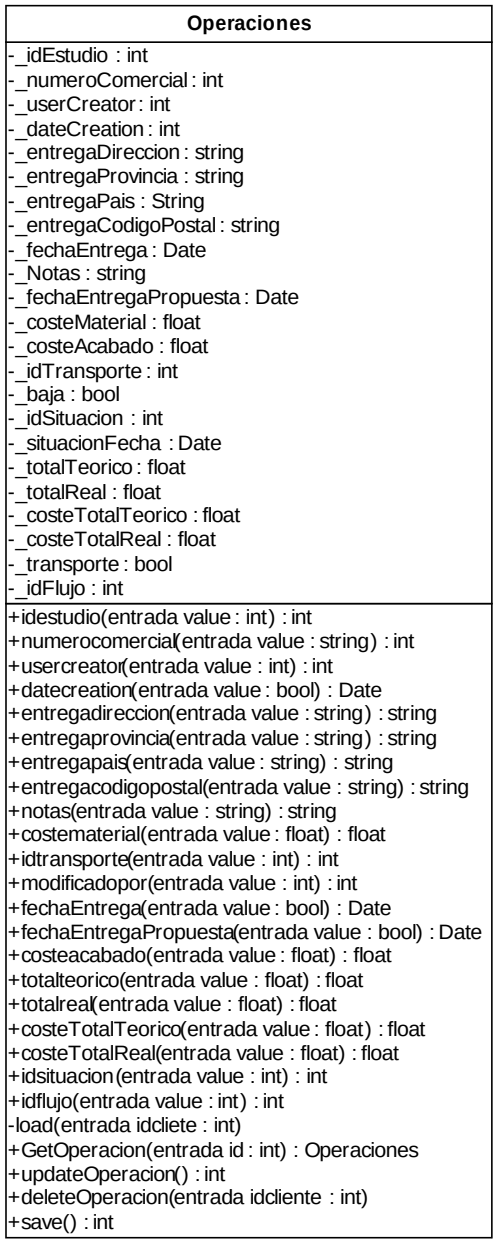
### Clase Estudios:




Explicación de los métodos.

- **idestudio**: es un property de lectura.
- **idcliente**: es un property de lectura.
- **titulo**: es un property de lectura y escritura
- **datecreation**: es un property de lectura
- **userCreator**: es un property de lectura
- **idmotivocompra**: es un property de lectura y escritura
- **idmotivoperdida**: es un property de lectura y escritura
- **valorprevisto**: es un property de lectura y escritura
- **mesprevisto**: es un property de lectura y escritura
- **añoprevisto**: es un property de lectura y escritura
- **fechamodificacion**: es un property de lectura
- **modificadoPor**: es un property de lectura
- **Load**: es un metodo privado que tiene como entrada el identificador del cliente y hace la llamada a la base de datos cargando toda la información
- **GetEstudio**: es el método publico para cargar el estudio
- **updateEstudio**: Esta propiedad tiene como entrada cada propiedad de la clase y se encarga de actualizar el objeto en la base de datos, guardando así las modificaciones. Si el identificador que se le pasa es 0, entonces insertará un nuevo estudio.
- **DeleteEstudio**: Borra el estudio de la base de datos.
- **Save**: una vez cargado el objeto, se hace el llamado a la base de datos para guardar el objeto en la tabla.

### Clase Operaciones:

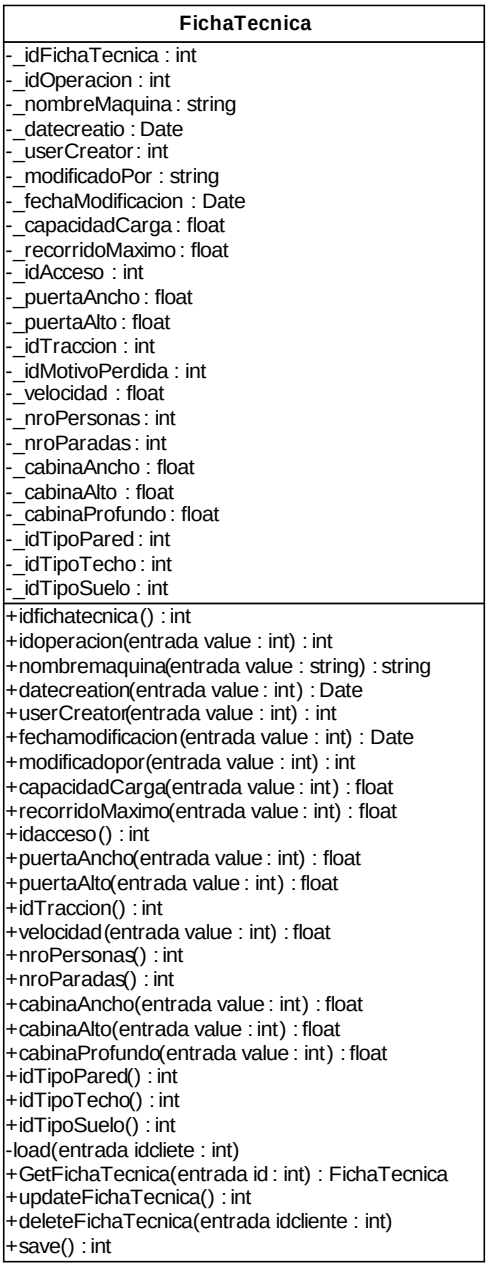



Explicación de los métodos:

- **idOperacion:** es un property de lectura.
- **idestudio**: es un property de lectura.
- numeroComercial: es un property de lectura y escritura
- **datecreation**: es un property de lectura
- **userCreator**: es un property de lectura
- **fechamodificacion**: es un property de lectura
- **modificadoPor**: es un property de lectura
- **entregadireccion**: es un property de lectura y escritura
- **entregaprovincia**: es un property de lectura y escritura

- **entregapais**: es un property de lectura y escritura
- **entregacodigopostal**: es un property de lectura y escritura
- **notas**: es un property de lectura y escritura
- **costematerial**: es un property de lectura y escritura
- **idtransporte**: es un property de lectura y escritura
- **fechaentrega**: es un property de lectura y escritura
- **fechaentregapropuesta**: es un property de lectura y escritura
- **totalteorico**: es un property de lectura y escritura
- **totalreal**: es un property de lectura y escritura
- **costetotalteorico**: es un property de lectura y escritura
- **costetotalreal**: es un property de lectura y escritura
- **idsituacion**: es un property de lectura y escritura
- **idflujo**: es un property de lectura y escritura
- **Load**: es un método privado que tiene como entrada el identificador de la operación y hace la llamada a la base de datos cargando toda la información
- **GetOperacion**: es el método publico para cargar la operación
- **updateOperacion**: Esta propiedad tiene como entrada cada propiedad de la clase y se encarga de actualizar el objeto en la base de datos, guardando así las modificaciones. Si el identificador que se le pasa es 0, entonces insertará una nueva operación.
- **DeleteOperacion**: Borra la operación de la base de datos.
- **Save**: una vez cargado el objeto, se hace el llamado a la base de datos para guardar el objeto en la tabla.

## Clase Ficha Técnica

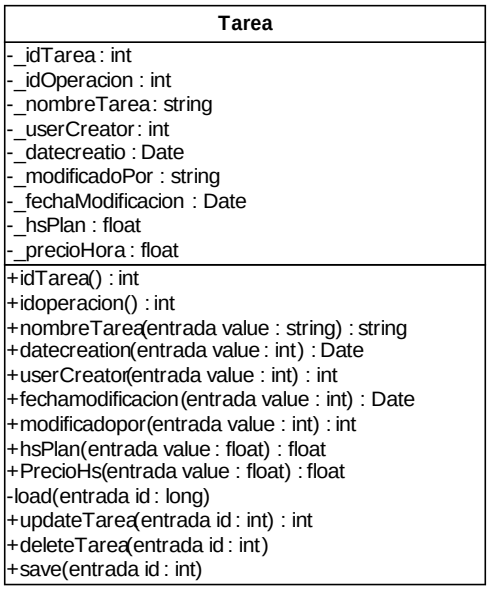



Explicación de los métodos:

- **idfichatecnica**: es un property de lectura.
- **idOperacion:** es un property de lectura.
- **nombremaquina**: es un property de lectura y escritura
- **datecreation**: es un property de lectura
- **userCreator**: es un property de lectura
- **fechamodificacion**: es un property de lectura
- **modificadoPor**: es un property de lectura

- **capacidadCarga**: es un property de lectura y escritura
- **recorridoMaximo**: es un property de lectura y escritura
- **idacceso**: es un property de lectura y escritura
- **puertaAncho**: es un property de lectura y escritura
- **puertaAlto**: es un property de lectura y escritura
- **idtraccion**: es un property de lectura y escritura
- **velocidad**: es un property de lectura y escritura
- **nroPersonas**: es un property de lectura y escritura
- **nroParadas**: es un property de lectura y escritura
- **cabinaAncho**: es un property de lectura y escritura
- **cabinaAlto**: es un property de lectura y escritura
- **cabinaProfundo**: es un property de lectura y escritura
- **idtipoPared**: es un property de lectura y escritura
- **idtiposuelo**: es un property de lectura y escritura
- **idTipoTecho**: es un property de lectura y escritura
- **Load**: es un método privado que tiene como entrada el identificador de la operación y hace la llamada a la base de datos cargando toda la información
- **GetFichaTecnica**: es el método publico para cargar la ficha técnica.
- **updateFichaTecnica**: Esta propiedad tiene como entrada cada propiedad de la clase y se encarga de actualizar el objeto en la base de datos, guardando así las modificaciones. Si el identificador que se le pasa es 0, entonces insertará una nueva ficha técnica.
- **DeleteFichaTecnica**: Borra la ficha Técnica de la base de datos.
- **Save**: una vez cargado el objeto, se hace el llamado a la base de datos para guardar el objeto en la tabla.

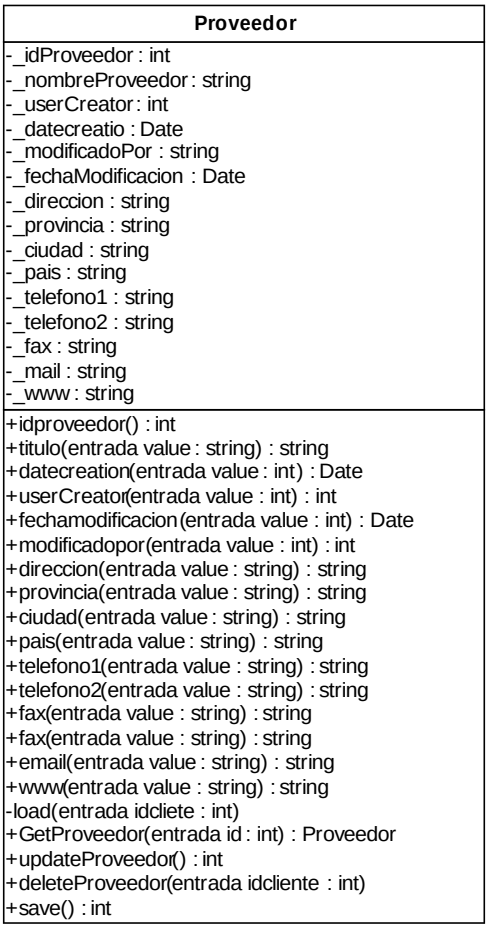
### Clase Tarea:




Explicación de los métodos:

- **idtarea**: es un property de lectura.
- **idOperacion:** es un property de lectura.
- **nombreTarea**: es un property de lectura y escritura
- **datecreation**: es un property de lectura
- **userCreator**: es un property de lectura
- **fechamodificacion**: es un property de lectura
- **modificadoPor**: es un property de lectura
- **hsPlan**: es un property de lectura y escritura
- **precioHora**: es un property de lectura y escritura
- **Load**: es un método privado que tiene como entrada el identificador de la tarea y hace la llamada a la base de datos cargando toda la información
- **GetTarea**: es el método publico para cargar la tarea.
- **updateTarea**: Esta propiedad tiene como entrada cada propiedad de la clase y se encarga de actualizar el objeto en la base de datos, guardando así las modificaciones. Si el identificador que se le pasa es 0, entonces insertará una nueva tarea.
- **DeleteTarea**: Borra la tarea de la base de datos.
- **Save**: una vez cargado el objeto, se hace el llamado a la base de datos para guardar el objeto en la tabla.

### Clase Proveedor

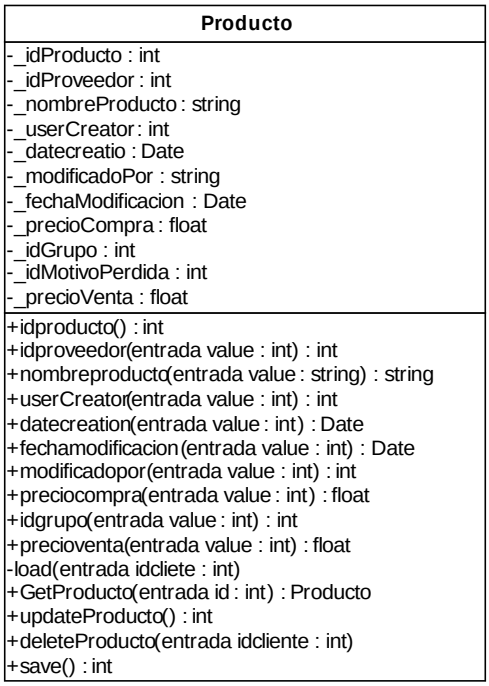



Explicación de los métodos:

- **idProveedor**: es un property de lectura.
- **nombreProveedor**: es un property de lectura y escritura
- **datecreation**: es un property de lectura
- **userCreator**: es un property de lectura
- **fechamodificacion**: es un property de lectura
- **modificadoPor**: es un property de lectura
- **direccion**: es un property de lectura y escritura
- **provincia**: es un property de lectura y escritura
- **ciudad**: es un property de lectura y escritura
- **telefono1**: es un property de lectura y escritura
- **telefono2**: es un property de lectura y escritura
- **fax**: es un property de lectura y escritura
- **email**: es un property de lectura y escritura
- **www**: es un property de lectura y escritura
- **Load**: es un método privado que tiene como entrada el identificador del proveedor y hace la llamada a la base de datos cargando toda la información

- **GetProveedor**: es el método publico para cargar el proveedor.
- **updateProveedor**: Esta propiedad tiene como entrada cada propiedad de la clase y se encarga de actualizar el objeto en la base de datos, guardando así las modificaciones. Si el identificador que se le pasa es 0, entonces insertará un nuevo proveedor.
- **DeleteProveedor**: Borra el proveedor de la base de datos.
- **Save**: una vez cargado el objeto, se hace el llamado a la base de datos para guardar el objeto en la tabla.

### Clase Producto:

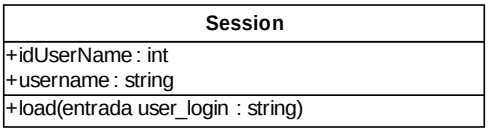



Explicación de los métodos:

- **idProducto**: es un property de lectura.
- **idProveedor**: es un property de lectura.
- **nombreProducto**: es un property de lectura y escritura
- **datecreation**: es un property de lectura
- **userCreator**: es un property de lectura
- **fechamodificacion**: es un property de lectura
- **modificadoPor**: es un property de lectura
- **precioCompra**: es un property de lectura y escritura
- **idgrupo**: es un property de lectura y escritura
- **precioVenta**: es un property de lectura y escritura
- **Load**: es un método privado que tiene como entrada el identificador del producto y hace la llamada a la base de datos cargando toda la información
- **GetProducto**: es el método publico para cargar el producto.
- **updateProducto**: Este método tiene como entrada cada propiedad de la clase y se encarga de actualizar el objeto en la base de datos, guardando así las modificaciones. Si el identificador que se le pasa es 0, entonces insertará un nuevo producto.
- **DeleteProducto**: Borra el producto de la base de datos.
- **Save**: una vez cargado el objeto, se hace el llamado a la base de datos para guardar el objeto en la tabla.

### Clase Sesión:

Esta clase se creo para mantener la información que nos interesa guardada en la sesión. En este caso el usuario que se registra al sistema. Si en algún futuro es necesario tener mas información guardada en la sesión, basta con tan solo modificar este objeto que se adaptará a cualquier cambio sin ningún tipo de problema.

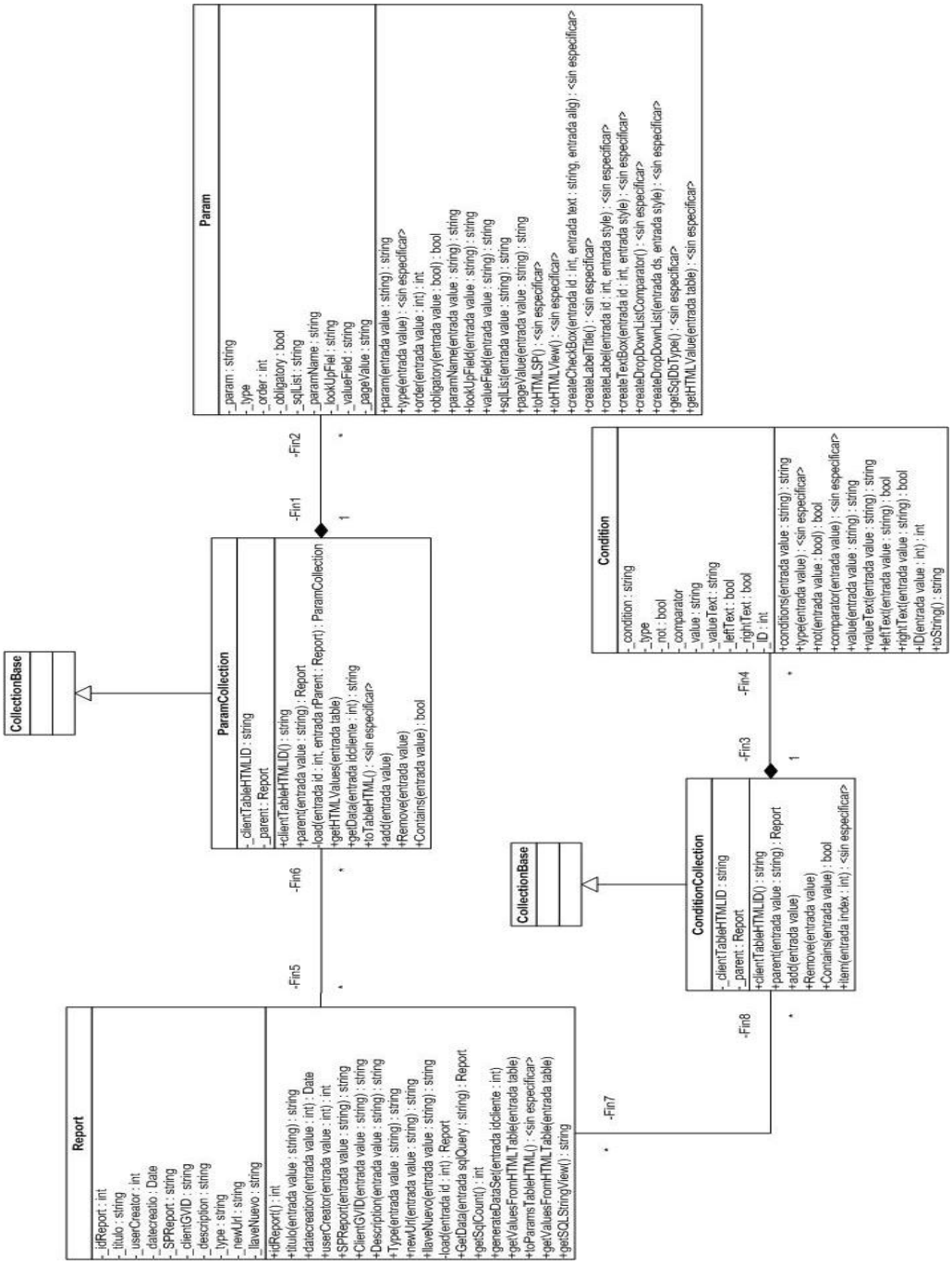



Explicación de los métodos

- **load**: cuando se registra con éxito un usuario, con el mismo user_login se carga sus datos en la sesión y estos quedan guardados hasta que se termina.

### Clase Report:

Los informes que aparecen en el sistema están procesados de una manera especial. En vez de crearse una pantalla para cada informe, se creo una sola que recoge el informe que le interesa de la base de datos. Para ello se creo una estructura de clases que se encarga de mostrar la información que nos interesa. Además cuenta con filtros y diferentes tipos de condiciones. La estructura para esta clase es mas compleja que las anteriores pero una vez implementada tan solo tenemos que agregar la consulta y los filtros deseados en las tablas de la base de datos y ya tenemos una nueva consulta para utilizar en el sistema. El diagrama UML de la clase es el siguiente:



Propiedades:

- _idReport : identificador del Reporte
- _titulo : titulo del reporte
- _userCreator : usuario que creo el reporte
- _datecreatio : fecha de creación
- _SPReport : Comando SQL que se ejecutará para crear el informe
- _clientGVID : nombre del gridView donde se devolverá la información
- _description : Descripción de la consulta.
- _type : Hay dos tipos de report, los que vienen de una vista (View) o de un procedimiento almacenado (SP)
- _newUrl : Esta variable guarda la dirección que debe irse para crear un nuevo objeto.
- _llaveNuevo : es la llave para mostrar o no el botón de “nuevo”

Métodos:

- idReport: es un property de lectura
- titulo: es un property de lectura
- datecreation: es un property de lectura
- userCreator: es un property de lectura
- SPReport: es un property de lectura
- ClientGVID: es un property de lectura
- Description: es un property de lectura
- Type: es un property de lectura
- newUrl: es un property de lectura
- llaveNuevo: es un property de lectura
- load: Carga el objeto entero desde la base de datos.
- GetData: carga las propierdades del objeto
- getSqlCount: obtiene el nro de lineas que devuelve el objeto
- generateDataSet: genera el dataSet que será puesto en el gridView posteriormente
- getValuesFromHTML: obtiene los valores de los filtros dependiendo el tipo de reporte que sea.
- toParamsTableHTML: establece los parámetros filtrados anteriormente.
- getSQLString: obtiene la lista de condiciones que se filtraron en la pantalla de cliente.

### Clase Param

Esta clase representa los parámetros de filtro de los informes. Por cada consulta que creemos, a esta le podremos agregar paramentos para poder filtrar la búsqueda.

Propiedades

- _param: nombre del parámetro a filtrar en la base de datos
- _type: tipo de parámetro (int, string, date, list)
- _order: orden en que aparecen los parámetros
- _obligatory: Si el parámetro es obligatorio
- _sqlList: Si el parámetro es de tipo list, aquí va el generador de la lista

- _paramName: nombre del parámetro a mostrar en pantalla
- _lookUpFiel: si el parámetro es de tipo list, este es el valor que se muestra en pantalla
- _valueField: si el parámetro es de tipo list, este es el valor que se usa para filtrar
- _pageValue: Es el valor con el que viene desde la pantalla de usuario para filtrar

Métodos:

- param: es un property de lectura
- type: es un property de lectura
- order: es un property de lectura
- obligatory: es un property de lectura
- paramName: es un property de lectura
- lookUpField: es un property de lectura
- valueField: es un property de lectura
- sqlList: es un property de lectura
- pageValue: es un property de lectura y escritura
- toHTMLSP: esta función crea una tabla con todos los parámetros para filtrar y mostrar en la pantalla del tipo procedimiento almacenado
- toHTMLView: esta función crea una tabla con todos los parámetros para filtrar y mostrar en la pantalla del tipo vista
- createCheckBox: crea un checkbox para los filtros
- createLabelTitle: crea un titulo para los filtros
- createLabel: crea un label para los filtros
- createTextBox: crea un textbox para los filtros
- createDropDownListComparator: crea un informe para los filtros numéricos
- createDropDownList: crea un informe para los filtros
- getSqlDbType: obtiene el tipo y determina como crear el filtro en el reporte.
- GetHTMValue: Obtiene los valores que se pusieron en los filtros para filtrar el informe.

### Clase ParamCollection:

Esta clase hereda de *CollectionBase*

Propiedades:

- _clientTableHTMLID : Es el nombre de la tabla donde se pondrán los filtros
- _parent : Es el report al cual hacen referencia

Métodos:

- clientTableHTMLID: es un property de lectura
- parent: es un property de lectura
- load: se encarga de cargar los filtros desde la base de datos que corresponden al parent.
- GetHTMLValues: Obtiene los valores para filtrar desde la pagina del cliente.
- getData: rellena el objeto con los datos de la base de datos
- toTableHTMLSP: crea la tabla con los filtros
- add: Agrega un nuevo “param”

- Remove:  quita un “param”
- Contains: verifica si tiene un “param” específico.

### Clase Condition

Esta clase es la que se encarga de crear los filtros para los “QuerySQL”. A partir de la información obtenida de los parámetros, se va armando las condiciones para filtrar.

Propiedades:

- _condition:  condición que afectara al filtro.
- _type: tipo de dato de la condición
- _not: si esta la condición negada
- _comparator: si es del tipo numérico, tenemos distintos tipos de comparador (=, >, <, etc)
- _value: El valor por el cual se quiere filtrar
- _valueText: el valor que se muestra en el cliente
- _leftText: Si es de tipo texto, para saber el tipo de filtro izquierdo
- _rightText: si es tipo Texto, para saber el tipo de filtro derecho
- _ID: identificador

Métodos:

- conditions: es un property de lectura  y escritura
- type: es un property de lectura  y escritura
- not: es un property de lectura  y escritura
- comparator: es un property de lectura  y escritura
- value: es un property de lectura  y escritura
- valueText: es un property de lectura  y escritura
- leftText: es un property de lectura  y escritura
- rightText: es un property de lectura  y escritura
- ID: es un property de lectura  y escritura
- toString: devuelve un texto para mostrar las condiciones por las cuales se esta filtrando al cliente

### Clase ConditionCollection

Esta clase hereda de *CollectionBase*

Propiedades:

- _clientTableHTMLID : Es el nombre de la tabla donde se pondrán los filtros
- _parent : Es el report al cual hacen referencia

Métodos:

- clientTableHTMLID: es un property de lectura
- parent: es un property de lectura

- add:  Agrega un nuevo “condition”
- Remove:  quita un “condition”
- Contains: verifica si tiene un “condition” específico.
- Item: devuelve una “condition” específica.

## ***Diagramas de Secuencias:***

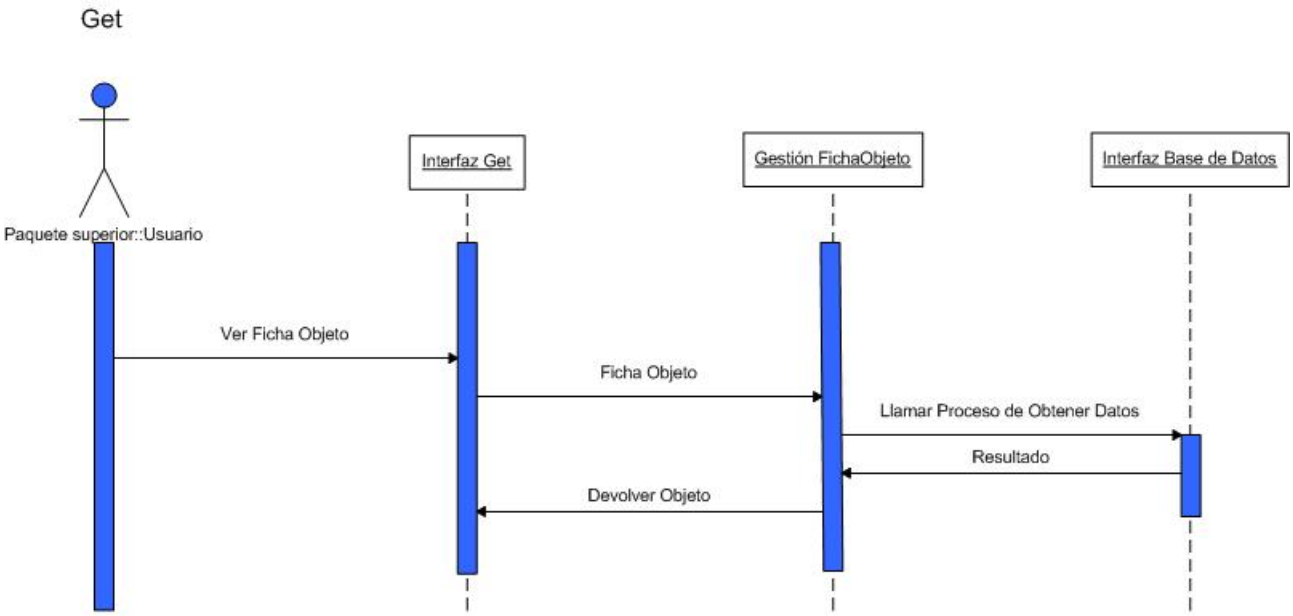
Dado que nuestro sistema tiene unas clases cerradas con prácticamente las mismas funcionalidades mostraré el diagrama de secuencia de esas funcionalidades básicas.

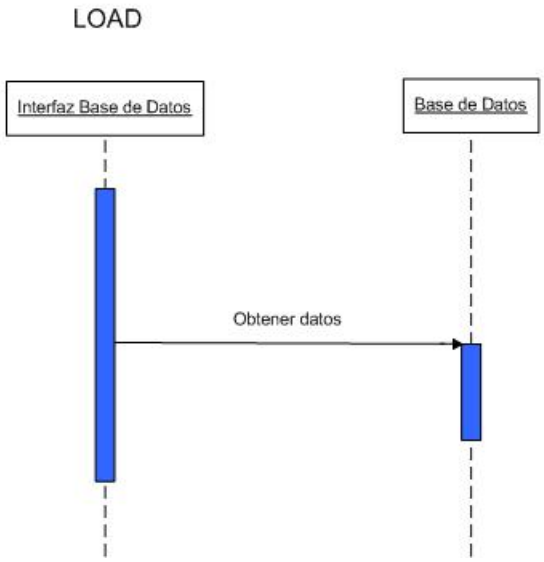
Las funcionalidades básicas son las siguientes:

- Get
- Load
- Update
- Save
- Delete

Cuatro de estas secuencias están relacionadas. La funcion get, llama a la de load y la función Update llama a la de Save. Esto se verá en los diagramas. La idea de esto era tener separado las llamadas a las base de datos y que sean accesibles desde el sistema y no desde el usuario.

**Diagrama de secuencia Get y Load.**



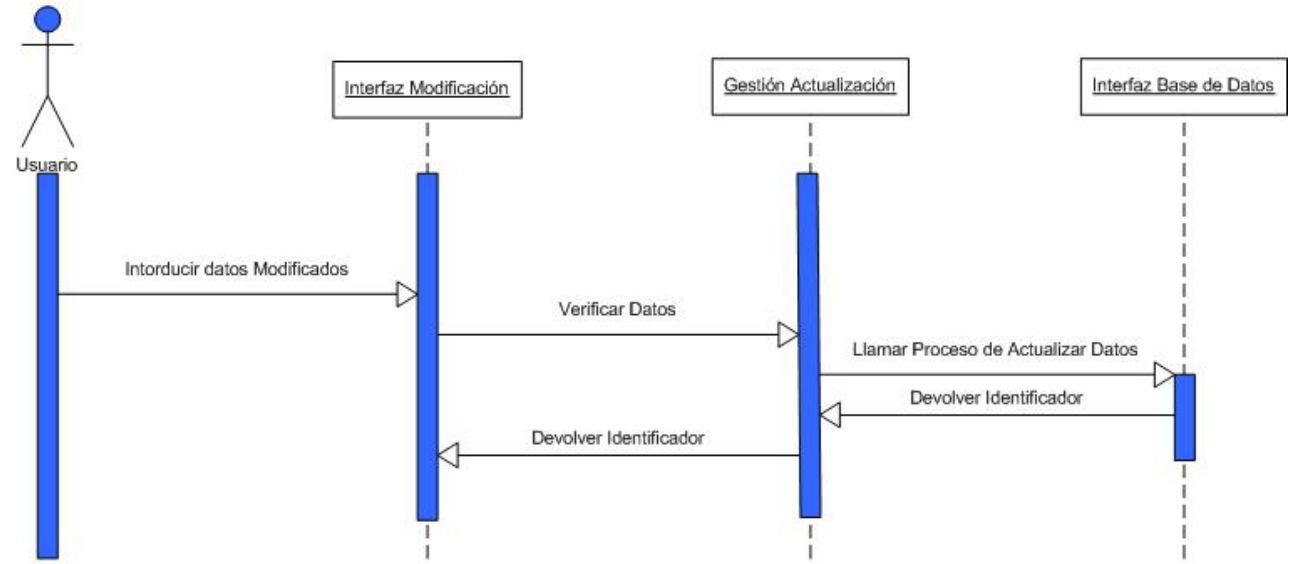

Con esta opción el usuario podrá pedir la información contenida en un objeto determinado.
Estará disponible para los usuarios que contengan la llave pertinente en el sistema.
Cada vez que se muestra la ficha de un objeto en pantalla se utilizará esta opción.

Precondiciones:

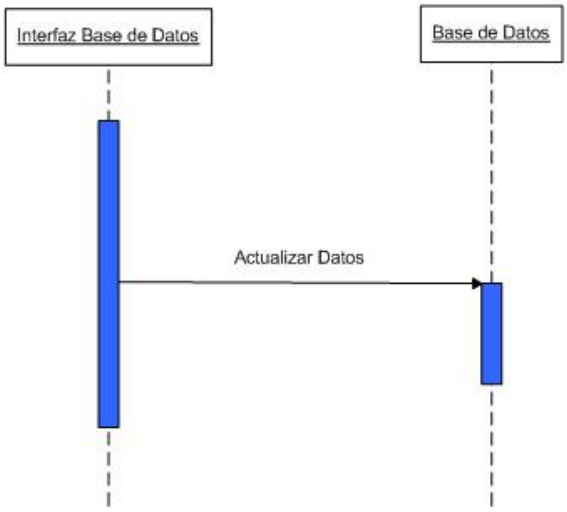
- Conexión correcta a la base de datos
- El usuario tiene la llave correspondiente

### Diagrama de secuencia Update y Save

Update




SAVE

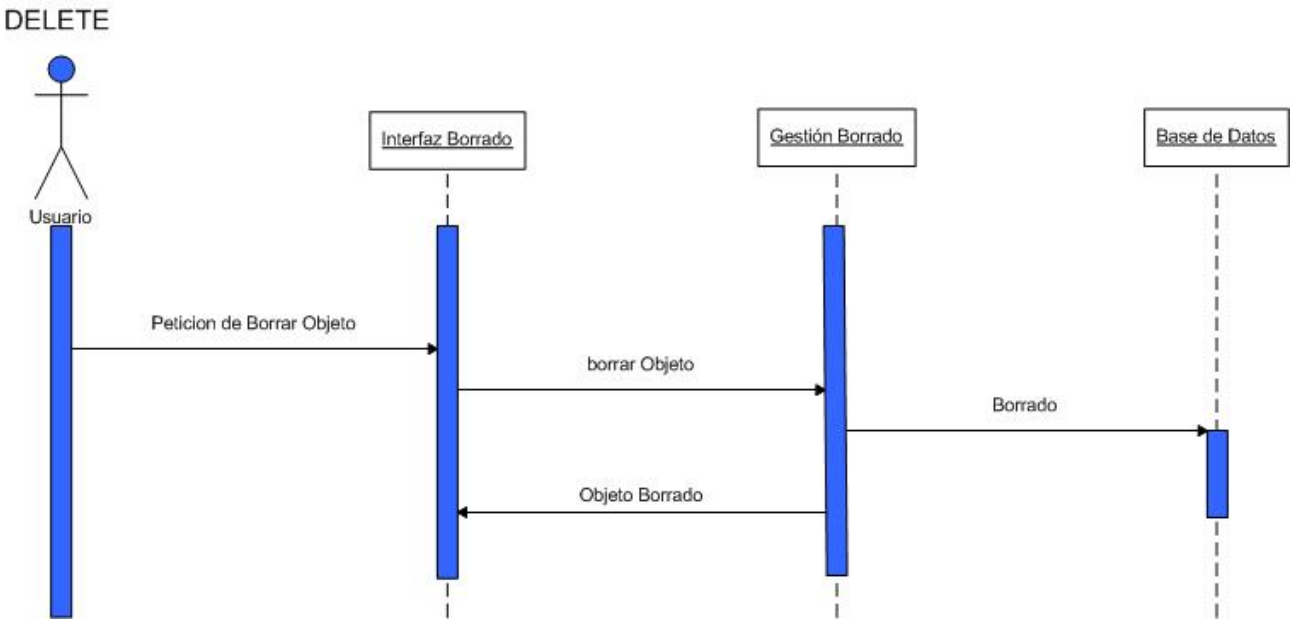



Con esta opción el usuario podrá guardar las modificaciones hechas en dicho objeto.
Estará disponible para los usuarios que contengan la llave pertinente en el sistema.
Cada vez que se guarde un objeto se llamará a esta opción.

Precondiciones:

- Conexión correcta a la base de datos
- El usuario tiene la llave correspondiente

### Diagrama de Secuencia Delete




Con esta opción el usuario podrá eliminar la información contenida en un objeto determinado.
Estará disponible para los usuarios que contengan la llave pertinente en el sistema.
Cada vez que se elimina un objeto se llamará a esta opción

Precondiciones:

- Conexión correcta a la base de datos
- El usuario tiene la llave correspondiente

# 6. Pruebas de ejecución

Esta es la última parte que concluye con el ciclo de vida del proyecto. Se han hecho diferentes tipos de pruebas según cada pantalla lo requiriese. Pruebas de interfaz, accesibilidad, etc.

Para asegurar que el resultado final fuese el que se esperaba se hicieron se trabajo de la siguiente manera. Dado que cada pantalla, exceptuando la de los flujos, tenia un comportamiento idéntico, se trabajo en una primera instancia con una sola, con la de clientes. Se vio la mejor manera para implementar el backOffice y luego las otras páginas surgieron a raíz de esta. A medida que se iban agregando se iban probando y verificando el correcto funcionamiento. Con estas pruebas se ha conseguido reducir los posibles errores que el usuario final podrá encontrar en el uso común del Back Office.

## *6.1. Pruebas de Unidad*

Con estas pruebas se pretende determinar el correcto funcionamiento de cada clase implementada, de manera independiente. Se ha llevado a cabo una serie de pruebas que aseguran que cada funcionalidad actúa tal y como se esperaba.

Las principales pruebas que se hicieron fueron:

- Realizar acciones de cada tipo y perfil de usuario (Comercial, Oficinas técnicas, planificación, taller, contabilidad y administrador). Verificando que cada uno tenga accesible la información que le corresponde como también las acciones.
- Realizar funciones propias en cada clase.
- Provocar errores en la edición de contenidos, de usuarios y en la carga de imágenes, verificando que la aplicación responda correctamente.
- Realizar varios test en el apartado de los flujos.
- Agregar y quitar llaves a los usuarios o a los grupos de usuario desde la base de datos y comprobar que se reflejaban en el sistema.
- Comprobación de tipos de datos
- Inserción de datos

Durante la fase de estas pruebas se han detectado errores y se han corregido correctamente.

## 6.2. Pruebas de accesibilidad

Las pruebas de accesibilidad se basan en la fácil navegación por las diferentes secciones del Back Office, llevando a crear una interfaz que ayuda al usuario a ubicarse mejor en el sistema sin la necesidad de un estudio previo. El objetivo es hacer un sistema intuitivo, con botones claros donde cada uno sabe exactamente lo que representan, respetando siempre los lugares de los mismos, con el mismo estilo y la misma funcionalidad.

En estas pruebas de interfaz se han establecido en buscar los colores mas acordes y la mejor distribución y colocación de cada una de las partes del Back Office. Intentando que la profundidad de navegación sea la mínima posible.

Las pruebas que se hicieron dieron como resultado el aspecto actual del Back Office. La distribución de la cabecera, junto que el menú vertical a la izquierda del sistema fueron diseñados varias veces al igual que la pantalla de datos y la de los informes. Con esto se logró hacer un sistema homogéneo, sencillo y que cumple con las expectativas creadas en un principio.

La solución presentada a todos los problemas surgidos en este apartado ha sido buscar las medidas idóneas para encajar todos los elementos. La parte de diseño artístico se tomo mucho en cuenta y se compararon con otros sistemas para sacar mejores ideas.

## 6.3. Pruebas de compatibilidad

Desde un principio se trato de buscar que la aplicación sea accesible desde cualquier navegador. Aunque se recomienda trabajar con Internet Explorer, también se enfocó en que no sea el único navegador accesible.

Básicamente lo que se hizo fue no confiar mucho en las opciones que trae el visual studio para la interfaz gráfica y manejarse mas con las plantillas de estilos. Los resultados fueron muy positivos ya que no he encontrado errores por lo menos con los siguientes navegadores, que son los que yo utilizo.

- Mozilla firefox 3.0 o superior
- Internet Explorer 7 o superior
- Google Chrome

La aplicación funciona de forma correcta y se visualiza como es debido en ambos navegadores.

## 6.4. Pruebas de seguridad

Las pruebas de seguridad se basan en mantener la intergirdad de los datos que los usuarios introducen en la base de datos.

Las pruebas realizadas han sido el intentar entrar en alguna página de la cual no se tiene permiso de acceso, donde se ha constatado que se deniega siempre la entrada redireccionando a la página de

login.

Asimismo si hay un tiempo de inactividad superior a 15 minutos la sesión queda caducada. Todas las pruebas realizadas no han generado mayores problemas y se han resuelto a medida que fueron surgiendo.

## *6.5. Pruebas de Regresión*

Esta aplicación necesita una persona administradora, por lo menos de la base de datos, para llevar a cabo tareas que los usuarios normales no pueden hacer. Es indispensable que haya una persona dedicada a esto y que además vaya haciendo copia de seguridad por posibles pérdidas de información. Se puede también elegir un proveedor web y de bases de datos donde se alojara todo, pero es indispensable una persona que se encargue del mantenimiento del Back Office.

# 7. Conclusiones

## *7.1. Generales*

Se han querido destacar las conclusiones desde diferentes puntos de vista, tanto en los aspectos que se ha podido alcanzar, como en el tiempo total de desarrollo y las posibles mejoras o ampliaciones que se pueden realizar sobre este proyecto.

## *7.2. Valoración del Proyecto*

### Objetivos.

En gran parte los objetivos se cumplieron, pero no en su totalidad. La razón de esto es que desde un principio este fue quizás un proyecto mas bien ambicioso. El funcionamiento principal y las cosas básicas que se plantearon fueron llevadas a cabo. Los objetivos que no se pudieron llevar a cabo son los siguientes.

- Pantalla de administración de usuarios: el administrador del sistema tendría que ser capaz de administrar los usuarios desde el sistema. Como era una funcionalidad mas específica de administradores y a la hora de crear usuarios y permisos se podían hacer desde la base de datos sin muchos inconvenientes fue algo que se fue dejando para el final, para que luego sea una de las cosas que no se han hecho.

- Pantalla de administración de Flujos: esta pantalla tenía su grado de complejidad y es algo que si bien puede ser provechoso cuando se tenga que usar, un flujo no es algo que estemos modificando constantemente, mas bien es una estructura fija y el tiempo que se podría llegar a invertir en esto no iba a valer tanto la pena para la valoración final del proyecto.

- Sistema de Compras para el departamento de Planificación: Un sistema de compras es algo necesario en el modelo de empresa que estamos representando, pero honestamente plantear bien este aspecto del proyecto es casi como plantear un nuevo proyecto. Se decidió no hacerlo por esta misma razón.

- Marcajes: este apartado se implementa con las planificación de las tareas por parte del departamento de planificación. La idea era estimar una duración de las tareas y luego corroborarlas con las horas marcadas y ver de ahí si hay una buena o mala planificación. Se decidió no hacerla por una cuestión de tiempo.

- Contactos de clientes que accedan al sistema desde fuera de la empresa: esta opción en un principio era muy atractiva, pero nuevamente el planteamiento de la misma es casi tan complejo como otro proyecto. Acceder desde fuera del BackOffice requiere mas seguridad, mas cuidado con la privacidad de la información, nuevas pantallas y un rediseño gráfico.

Además de inclusiones de nuevas opciones y restricciones que se le pueden llegar a dar al contacto del cliente. Básicamente la idea era que el cliente tenga conocimiento del estado de sus máquinas en la empresa.

El objetivo principal de este proyecto se centró en la administración de la información de la empresa y el manejo de la misma a través de diferentes flujos según el tipo de usuario que la este utilizando. Este objetivo se ha cumplido en su totalidad respecto a las principales funcionalidades. Los objetivos cumplidos fueron los siguientes:

- Administración de datos: tanto los clientes, estudios, operaciones, etc se pueden modificar, dar de alta o borrar. Toda la información que aparece en el sistema es modificable.

- Diferentes tipos de usuario: Cada usuario cumple un rol en el sistema, dependiendo el tipo de usuario que sea, podrá hacer diferentes tareas en el sistema. La asignación de llaves a los usuarios para la administración de estas cosas ha sido una buena idea y se ha implementado con total éxito.

- Utilización de flujos: los flujos van marcando el ritmo de la empresa. Cada vez que se va avanzando en el flujo, es un paso menos para terminar con el trabajo asignado. De esta manera el trabajo de la empresa esta organizado y va afectando a los responsables en cada trabajo en el momento que es necesario. Los flujos ademas ayudan a la organización del trabajo de cada individuo que utilice el sistema.

Se han creado mas de diez pantallas y listados para la administración de la empresa, cada una relaciona entre sí. Todos los usuarios tienen los permisos definidos en el sistema y en cualquier momento se pueden agregar un solo permiso o a un grupo de permisos sin ningún problema. Los flujos son modificables también en cualquier punto dando completa flexibilidad y pudiéndose adaptar a cualquier modelo.
Cada aspecto del sistema fue pensado en que sea escalable y gracias a la organización de las clases y las interfaces es muy sencillo ir creciendo en el sistema desde el punto en el que esta ahora.

## Plan de Trabajo.

La planificación del proyecto no fue acertada. El tiempo estimado fue menor al real. La diferencia no ha sido muy grande, pero en donde he fallado en esta planificación fue en la distribución de las horas. Mi situación actual laboral no me dejaba seguir con la estimación que hice en un principio. Además de tener asuntos personales que me alejaban de tener una continuidad en el proyecto hicieron que me retrasase un año en la entrega.

Los motivos por los cuales me he retrasado son: la jornada laboral de 40 horas semanales, mas cursar materias durante la semana y hacer cursos los sábados en el 2009-2010. Además mi situación sentimental con una persona que vive en Argentina y con viajes de por medio hicieron que el proyecto lo pueda ir haciendo en mi tiempo libre.

Las horas previstas para este proyecto fueron menores a las reales. En general el desarrollo, testing y la documentación me han llevado bastante mas tiempo del que supuse en un principio. Tengo que admitir que la documentación del proyecto me ha costado mas tiempo del programado. El motivo

de esto será que no estoy familiarizado con la creación de memorias y cuesta un poco mas hacer un documento de este estilo sin tener un poco mas de práctica. En total he calculado que este proyecto me ha llevado unas 55 Hs. mas a las planificadas desde un principio.

| Tarea | Previsto | Real |
|---|---|---|
| Est. Viabilidad | 20 Hs | 20 Hs |
| Definición Proyecto | 35 Hs | 35 Hs |
| Desarrollo Web | 50 Hs | **70 Hs(+)** |
| Testing | 20 Hs | **40 Hs(+)** |
| Documentación | 25 Hs | **40 Hs(+)** |
| **TOTAL** | **150 hs** | **205 hs** |

Honestamente me es muy difícil hacer un diagrama de Gantt de como he trabajado de verdad, porque no tengo un patrón o no he seguido ningún tipo planificación. Durante todo este tiempo he trabajado en pequeños momentos libres de mi tiempo y en épocas donde me encontraba solo. He avanzado con el proyecto generalmente en fines de semana y en noches cuando eran días laborables. También he aprovechado los días festivos del año para sentarme y seguir adelante.

### Ampliaciones y mejoras.

Todos los objetivos no cumplidos son claras ampliaciones y mejoras al sistema. Tanto la pantalla de administración de usuarios, pantalla de administración de flujos, sistema de compras, marcajes y acceso al sistema a los contactos de los clientes son ampliaciones a tener en cuenta en un futuro y estás deberían ser las siguientes a seguir.

Durante el desarrollo de los flujos he pensado en la necesidad de retroceder un estado, no solo avanzar. Esta mejora la veo importante ya que cualquier modificación que se tenga que hacer en el proceso de fabricación pasado habría que notificar a todos los nodos afectados al cambio.

Una vez terminadas estas mejoras, pasaríamos a un nivel de abstracción mayor donde analizaríamos mejor la información y daríamos lugar a reports, informes con gráficos y cubos de información donde se podrá obtener mejores resultados de la empresa y sacar conclusiones a raíz de ellos. Por ultimo se podría tener un panel de control donde se podrá ver el estado de la empresa bajo unos parámetros fáciles de leer.

### Valoración personal

Personalmente estoy muy contento de concluir esta etapa de mi vida con el este broche de oro. Yo, que hace siete años he cambiado de país, de costumbre y de vida, me llena de orgullo ver lo capaz que he sido en seguir adelante con esta ultima prueba  y con ello dar por finalizada esta carrera.

Este proyecto me ha resultado un reto a lo personal puesto a que he decidido desde el principio todo

lo que se iba a hacer. No me he encontrado con cosas ya hechas o con un estilo ya definido. Siempre me ha tocado adaptarme al modelo de trabajo establecido a seguir o con una estructura preestablecida por otra persona. Con este proyecto he podido ver que yo también puedo armar la base y que a partir de ahí puedo seguir construyendo. Me ha costado mucho encontrar el momento adecuado para hacerlo, no he gozado de un tiempo libre adecuado para organizarme con ello. Pero pese a ello no he bajado los brazos en ningún momento y he podido con todas las dificultades

A lo largo de la carrera he tenido la suerte de poder trabajar de lo que estudio y en este proyecto he podido utilizar toda mi experiencia laboral mas todos mis conocimientos adquiridos en la facultad para la realización de todos los objetivos que me he impuesto.

Esto es todo lo que tengo que decir sobre mi punto de vista. Me siento preparado para encarar mas cosas de este estilo y espero que mi vida profesional a partir de aquí siga creciendo tanto como lo ha hecho este tiempo en la Universidad Autónoma de Barcelona.

# 8. Bibliografía

Diseño interfaz gráfica y programación en ASP.NET, JavaScript y SQL Server.


1. Titulo: Así es Microsoft .NET
   Autor: David S. Platt McGraw-Hill, 2001
   Descripción: Libro de Microsoft.NET
   URL:

2. Titulo: MSDN
   Autor: Microsoft
   Descripción: Contiene una gran cantidad de información técnica de programación, incluidos código de ejemplo, documentación, artículos técnicos y guías de referencia.
   URL:http://msdn.microsoft.com/es-es/default

3. Titulo: Manuales de Diseño Web, CSS, PHP, MySQL, JavaScript
   Autor: Miguel Angel Alvarez
   Descripción: Varios manuales muy bien especificados, acerca del diseño Web, CSS, PHP, MySQL, JavaScript
   URL:http://www.dearrolloweb.com/manuales/

4. Titulo: Guillermo Fernando Prestigiacomo
   Autor: Guillermo Fernando Prestigiacomo
   Descripción: mi hermano ha de aparecer aquí también ya que para mi fue una fuente de conocimiento y un lugar en donde aclarar dudas, más que en cualquier libro
   URL: www.gpresti.com

5. Titulo: WikiPedia
   Autor: El mundo entero
   Descripción: la wikipedia me ha ayudado mucho en varias dudas que aclarar
   URL: es.wikipedia.org

# 9. Agradecimientos

Mis agradecimientos van dirigidos a todos y cada uno de los profesores que me han enseñado a lo largo de toda mi vida. Tanto en la primaria, en la secundaria, en la facultad, en los cursos y en la vida misma. Toda persona que haya dejado un granito de arena en mi.

Agradecimientos especiales a aquellos que han estado a mi lado, a mi hermano por darme esta orientación y brindarme sus conocimientos, a mis padres que me han formado como persona y han puesto sus esperanzas e ilusiones en mi mismo, a mi familia que no han dejado de apoyarme y de alegrarme y a mi novia que me ha dado el pequeño empujón que me hacía falta para terminar con todo esto.

Quiero agradecer a mi director de Proyecto que ha acertado en su forma de orientarme, me ha marcado bien los tiempos y me ha sabido entender mi situación sobre el proyecto permitiéndome adaptarme a él.

Por ultimo agradecer a los amigos que han pasado a lo largo de la carrera y a lo largo de mi vida.

Gracias a todos.

Guillermo Federico Prestigiacomo
Enero del 2011