Universidad de las Ciencias Informáticas

cujae

Instituto Superior Politécnico "José Antonio Echeverria"
INGENIERIA INFORMATICA

Sistema de Reservación de Laboratorios

Trabajo para optar por el título de Ingeniería en Informática

Autor

Wilber Yoel Luque Pérez

Tutores

Ing. Joe Fernández Vanega
Ing. Rudel Cárdenaz Días

Ciudad de la Habana
Julio de 2005

Resumen:

El proyecto desarrollado consiste en la creación de un sistema que posibilite la gestión del proceso de reservación de varios recursos en la Universidad de las Ciencias Informáticas (UCI) en este caso los laboratorios.

El proyecto surge a partir de la idea de lograr la integración de todos los sistemas con el fin de aprovechar la infraestructura de redes y equipamiento informático, dando solución a un problema de primer orden, garantizar el acceso a los recursos dentro de los laboratorios al personal y estudiantado de una forma eficiente y organizada a través de los medios antes mencionados.

No existía en la universidad previamente a la creación del Sistema Integral de Reservaciones de Laboratorios ningún sistema que resolviera en alguna medida la gestión de la reservación los tiempos de máquina.

Este sistema permite el acceso de forma cómoda, segura y eficiente a reservar cualquier recurso de los actualmente estipulados dentro de los laboratorios. Además se integra a los sistemas existentes actualmente y se nutre de la información que estos manejan.

ÍNDICE

ÍNDICE

ÍNDICE

ÍNDICE DE FIGURAS

Introducción:

Debido a la creciente demanda de servicios aparejada a la construcción de la UCI, la formación del estudiantado y el funcionamiento en general de la entidad hacen necesario el acceso a estos de forma práctica y organizada. Pero no siempre están en disponibilidad en todo momento y para todo el personal. Entonces surge la necesidad de realizar una planificación personal del acceso a estos elementos en función de las necesidades propias, el momento deseado para tal y la disponibilidad; para lograr esto de una forma ordenada y segura se implementan mecanismos de reservación.

La UCI cuenta con una infraestructura que posibilita la informatización de los recursos para su mayor aprovechamiento por lo que esta es la idea fundamental desde donde partimos en este proyecto, un sistema que integrado a los demás sistemas existentes en estos momentos sea capaz de brindar un servicio, integro a la parte de reservación de laboratorios y tiempos de máquinas así como la supervisión del estado de los medios actuales y del personal responsable de estos.

Antes de la construcción de este sistema no existía ningún otro que implementara de alguna manera lo que este sistema propone, solo algunos intentos que hay que reconocer pero que no han cumplido con los requerimientos tales como integración con los demás sistemas y una visión real del tema. Por lo que este proceso se lleva “a mano” haciendo necesaria su rápida implementación.

INTRODUCCIÓN

Este sistema aportara una solución efectiva e integrada para la realización de reservaciones dentro de los laboratorios, además de cierto control del personal responsable del buen funcionamiento de los mismos.

Para la realización de este sistema se ha profundizado en gran medida dentro de la organización y métodos anteriores de realizar el proceso de reservación. Además se realizó una investigación de los sistemas actualmente implantados aquí en la UCI, luego de un amplio análisis de estos se llegó a un consenso de cuales nos servirían para integrarlos con nuestro sistema. Dando como resultado un sistema con una alta integración a los demás existentes.

Este proyecto tiene como objetivo fundamental la construcción de un sistema que además de estar integrado a la infraestructura existente permita de una forma fácil, cómoda y con una gran facilidad de comprensión la reservación de los recursos anteriormente expuestos.

Teniendo además como objetivos específicos:

- Permitir la reservación de tiempos de máquina por parte de los estudiantes desde cualquier sitio con acceso al sistema.
- Permitir la reservación de laboratorios por parte de los profesores.
- Manejar la información referente a la distribución o ubicación de técnicos por laboratorio.
- Manipular la información del estado de los laboratorios y máquinas en específico.

Para la correcta realización de estos objetivos se ha investigado a fondo las necesidades del personal responsable como jefes de área, técnicos de los laboratorios, se realizaron numerosas entrevistas formales y no formales a jefes de área, técnicos, a empleados responsables del proceso manual de reservación así como a los compañeros del departamento de informatización a los que va dirigida una gran parte del proyecto.

Además se trazaron las siguientes tareas:

- Estudio de las tendencias actuales que brindan solución a problemas similares.
- Fundamentación teórica del sistema.
- Reconocimiento de Requisitos funcionales y Confección de los casos de uso.
- Diseño del sistema
- Implementación del sistema
- Prueba e implantación del sistema.

Con esto se pretende realizar la confección de un software a la altura de las exigencias actuales que realice de una forma rápida y eficiente las tareas planteadas.

El presente trabajo ha sido organizado de la siguiente forma:

Capítulo 1: Fundamentación del tema: contiene los conceptos necesarios para la comprensión plena de los temas tratados en el resto del documento.

Capítulo 2: Trata la situación de las tecnologías a utilizar en el desarrollo de la aplicación, se comparan y seleccionan las mejores propuestas para el trabajo, y se explican los conceptos principales que se van a tratar.

Capítulo 3: Describe el negocio a través de un modelo de Dominio, y se hace el análisis del sistema a desarrollar. Se definen las funcionalidades del sistema y se describen detalladamente, utilizando herramientas de modelación.

Capítulo 4: Enfoca la construcción de la solución mediante diagramas de clases, de datos, y se plantean los principios para el diseño y la implementación. Aquí se construyen las funcionalidades que se definieron en el capítulo anterior.

Capítulo 5: Es un estudio de factibilidad sobre el sistema, obteniendo los beneficios tangibles e intangibles y analizando los costos del desarrollo de esta propuesta.

CAPÍTULO 1

FUNDAMENTACIÓN DEL TEMA

1.1 Introducción:

En el presente capitulo se brinda una vista general de los temas relacionados con los Servicios Informáticos de Reservaciones, además se muestran los principales conceptos asociados al dominio del problema que son necesarios para entender el modelo de negocio y la propuesta de la solución.

Se describen los procesos del negocio que se relacionan con el objeto de estudio de este trabajo. Se identifican los principales problemas que fundamentan la propuesta de solución, y se marcan los objetivos generales y específicos.

1.2 Principales conceptos asociados al dominio del problema.

¿Qué es Internet?

Es una gigantesca red de redes o interconexión de ordenadores entre sí. El desarrollo de la tecnología necesaria para poner en funcionamiento esta mega estructura de comunicación ha tenido un crecimiento exponencial en los últimos años, debido en gran medida a la popularización de los ordenadores personales, que ha llevado Internet tanto a las empresas como al hogar.

Historia de Internet

-En los años sesenta, el ejercito de EE.UU. desarrollo un nuevo sistema de comunicaciones llamado ARPANET que luego se convertiría en Internet.
-Pero antes en 1962 J.C.Licklider postula la idea de una interconexión entre diferentes ordenadores para compartir información.
-En los siguientes años se desarrolla ARPANET. Para que en 1985 se cree TCP/IP el lenguaje de Internet.
-En 1991 aparece la World Wide Web (WWW).
-En 1995 se desarrolla el programa Java.

1.2.1 World Wide Web o WWW.

Es el mecanismo proveedor de información electrónica para usuarios conectados a Internet. El acceso a cada sitio Web se canaliza a través del URL o identificador único de cada página de contenidos que permite a los usuarios el acceso a una gran cantidad de información. Gracias a la forma en que está organizada la World Wide Web, los usuarios pueden saltar de un recurso a otro con facilidad.

Los usuarios visualizan estos datos mediante una aplicación, conocida como explorador o *browser*, que muestra en pantalla una página con el texto, las imágenes, los sonidos y animaciones relativas al tema que previamente ha sido seleccionado. Existen múltiples enlaces Web por todo el mundo, que forman una base de información a gran escala en formato multimedia, aunque todavía los contenidos se encuentran mayoritariamente en inglés.

Las páginas Web pueden estar escritas en HTML (siglas de *Hypertext Markup Language*), DHTML o XML (*Extended Markup Language*), lenguajes de marcado de hipertexto. El protocolo HTTP (siglas de *Hypertext Transfer Protocol*) es el

encargado de hacer llegar las diferentes páginas desde los servidores remotos al equipo del usuario.

1.2.2 Intranet

Es la integración en una red local de tecnologías avanzadas de publicación electrónica basadas en WEB en combinación con servicios de mensajería, compartición de recursos, acceso remoto y toda una serie de facilidades cliente/servidor. Diseñado inicialmente para la red global Internet. Su propósito fundamental es optimizar el flujo de información con el objeto de lograr una importante reducción de costes en el manejo de documentos y comunicación interna.

Es una herramienta de gestión que permite una potente difusión de información y mecanismos de colaboración entre el personal.

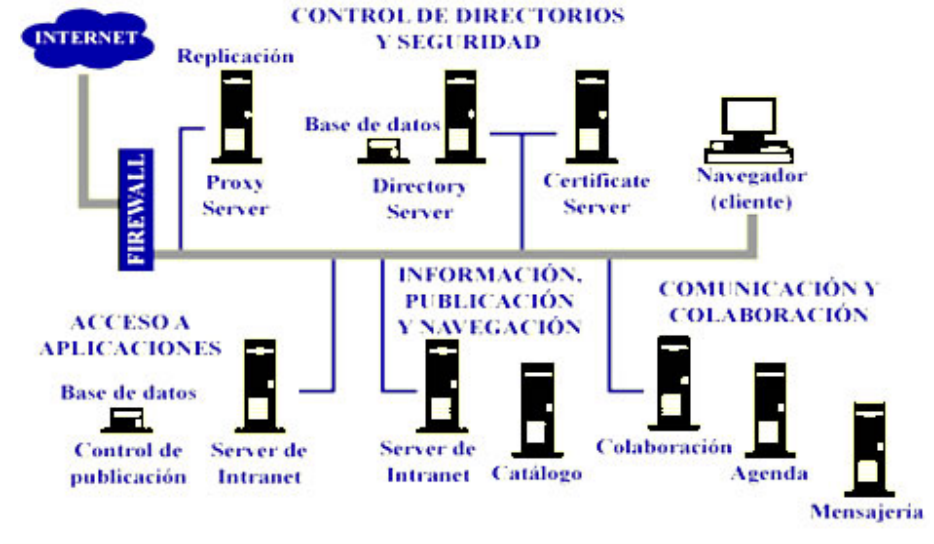


FIGURA 1 ESTRUCTURA DE UNA INTRANET

TCP/IP

Permite que diferentes tipos de computadoras se comuniquen a través de redes heterogéneas.

Se desarrolló originalmente para la comunicación de computadoras con sistema operativo UNIX a través de ARPANET, pero ahora está disponible para establecer una conexión a través de Internet usando cualquier sistema operativo. TCP define distintos parámetros de transmisión de datos. IP define el modo en que los datos se dividen en bloques, denominados paquetes, y establece el camino que recorre cada paquete hasta su destino.

NAVEGADORES

Son aquellos programas que permiten visualizar las páginas Web de la red.

Nestcape.

Creado por la empresa Communication Corp., es hasta ahora el navegador más popular. Pero no el más usado, ya que éste es su más directo competidor: explorer

Explorer.

Es el programa mas utilizado en la red. Pertenece a la empresa Microsoft. Además ofrece una aplicación para construir Webs y un cliente de correo.

1.3 Objeto de estudio.

1.3.1 Descripción general.

En la UCI se prestan servicios de reservación de laboratorios y tiempos de máquina, planificando de una manera organizada y lo mas igualitaria posible el tiempo de utilización de los recursos dentro de los laboratorios. Estos servicios son brindados a lo estudiantes y profesores que no son más que los posibles usuarios del sistema.

1.3.2 Descripción del proceso de negocio actual

Actualmente en la uci no existe ningún sistema que de alguna manera de solución rápida y efectiva a la reservación de laboratorios y tiempos de máquina. Este proceso se realiza de forma manual es decir una empleada en cada planta de los docentes en un determinado horario, con una plantilla donde tiene los puestos de trabajo y a medida que van pasando los estudiantes se les va asignando un puesto de trabajo en un horario determinado, mientras los profesores debían ir ante la vicedecana de su facultad y hacerle un pedido para un día determinado. Por lo que es bastante tedioso y lento. Por otra parte la gestión de reparaciones de las máquinas es mediante un modelo a modo de reporte que cada técnico llenaba en su turno de trabajo y una vez terminado este era entregado a su respectivo jefe de área que llenaba un modelo general y se lo entregaba al compañero de COPEXTEL que le correspondía esa área.

1.3.3 Situación Problémica.

No existe en la UCI ningún mecanismo capaz de brindar al usuario un servicio de reservación eficiente, capaz de emitir una respuesta inmediatamente.

1.4 Objetivos generales y específicos.

Para dar respuesta rápida y eficiente a la situación problémica planteada anteriormente nos hemos propuesto una serie de objetivos:

1.4.1 Objetivo general.

Como objetivo general nos planteamos hacer la versión beta del Sistema de Reservación de Laboratorios, el cual permitirá la procesar las solicitudes de reservación de los recursos en los laboratorios y emitir una respuesta. Aumentando la eficiencia de estos servicios.

1.4.2 Objetivos específicos.

El sistema debe ser capaz de:

- Realizar la reservación de los tiempos de máquina por parte de los estudiantes.
- Realizar la reservación de laboratorios.
- Mostrar el estado de los medios.
- Mostrar distribución de técnicos por laboratorio.
- Mostrar el estado de las reservaciones.

1.5 Conclusiones.

En el presente capítulo se detallaron las condiciones y problemas que rodean el objeto de estudio, y a través de las definiciones y conceptos, se determinaron las condiciones específicas que rodean al problema y en base a esto se obtuvieron los objetivos generales y específicos para este trabajo.

CAPITULO 2

TENDENCIAS Y TECNOLOGIAS ACTUALES A CONSIDERAR

2.1 Introducción:

En este capitulo, se hace un estudio de los temas relacionados con este trabajo, a nivel nacional e internacional, definiendo conceptos importantes de la teoría en que se basa la solución del problema, facilitando la comprensión del mismo. Además se analizaran las herramientas, las metodologías para el análisis y diseño del sistema y los lenguajes de programación.

2.2 Fundamentación de las tecnologías en que se basa la propuesta.

2.2.1 .NET Framework.

Los Ingenieros de Microsoft se han preocupado por brindarle a los desarrolladores un entorno de desarrollo que le permita disponer de una gran serie de herramientas y tecnologías tendientes a facilitar el desarrollo de aplicaciones web potentes y distribuidas, creando un ambiente multiplataforma, altamente deseado por todos los desarrolladores. [5]

El .NET Framework es un marco de trabajo multilenguaje, que le permite al desarrollador crear Aplicaciones y Servicios Web con las herramientas básicas para escribir el código.

De forma simple, el .NET Framework está formado por el Common Languaje Runtime o CLR, la Base Class Library, que funciona como una gran librería de clases unificada, que contiene todas las clases que funcionan dentro del entorno .NET y finalmente la nueva versión de ASP, denominada ASP.NET.

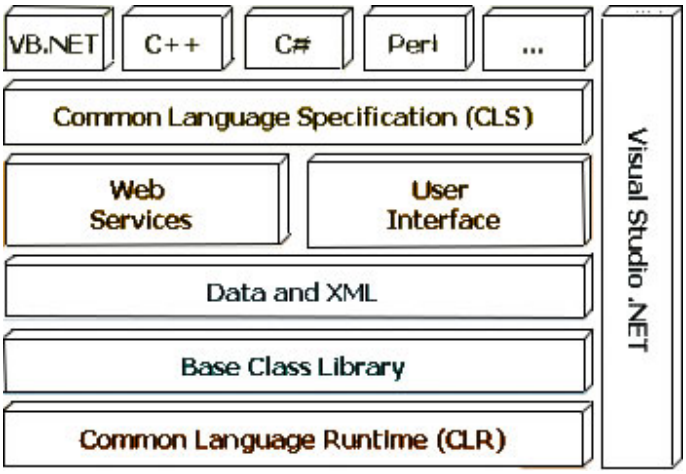


FIGURA 2 *COMPONENTES DE MICROSOFT .NET*

2.2.1.1 Common Language Runtime CLR

El Common Language Runtime (CLR) es el núcleo de la plataforma .NET. Es el motor encargado de gestionar la ejecución de las aplicaciones para ella desarrolladas y a las que ofrece numerosos servicios que simplifican su desarrollo y favorecen su fiabilidad y seguridad.

Las principales características y servicios que ofrece el CLR son:

Modelo de programación consistente: A todos los servicios y facilidades ofrecidos por el CLR se accede de la misma forma: a través de un modelo de programación orientado a objetos. Esto es una diferencia importante respecto al modo de acceso a los servicios ofrecidos por los algunos sistemas operativos actuales (por ejemplo, los de la familia Windows), en los que a algunos servicios se les accede a través de llamadas a funciones globales definidas en DLLs y a otros a través de objetos (objetos COM en el caso de la familia Windows)

Modelo de programación sencillo: Con el CLR desaparecen muchos elementos complejos incluidos en los sistemas operativos actuales (registro de

Windows, GUIDs, HRESULTS, IUnknown, etc.) El CLR no es que abstraiga al programador de estos conceptos, sino que son conceptos que no existen en la plataforma .NET

Eliminación del "infierno de las DLLs": En la plataforma .NET desaparece el problema conocido como "infierno de las DLLs" que se da en los sistemas operativos actuales de la familia Windows, problema que consiste en que al sustituirse versiones viejas de DLLs compartidas por versiones nuevas puede que aplicaciones que fueron diseñadas para ser ejecutadas usando las viejas dejen de funcionar si las nuevas no son 100% compatibles con las anteriores. En la plataforma .NET las versiones nuevas de las DLLs pueden coexistir con las viejas, de modo que las aplicaciones diseñadas para ejecutarse usando las viejas podrán seguir usándolas tras instalación de las nuevas. Esto, obviamente, simplifica mucho la instalación y desinstalación de software.

Ejecución multiplataforma: El CLR actúa como una máquina virtual, encargándose de ejecutar las aplicaciones diseñadas para la plataforma .NET. Es decir, cualquier plataforma para la que exista una versión del CLR podrá ejecutar cualquier aplicación .NET. Microsoft ha desarrollado versiones del CLR para la mayoría de las versiones de Windows: Windows 95, Windows 98, Windows ME, Windows NT 4.0, Windows 2000, Windows XP y Windows CE (que puede ser usado en CPUs que no sean de la familia x86) Por otro lado Microsoft ha firmado un acuerdo con Corel para portar el CLR a Linux y también hay terceros que están desarrollando de manera independiente versiones de libre distribución del CLR para Linux. Asímismo, dado que la arquitectura del CLR está totalmente abierta, es posible que en el futuro se diseñen versiones del mismo para otros sistemas operativos.

Integración de lenguajes: Desde cualquier lenguaje para el que exista un compilador que genere código para la plataforma .NET es posible utilizar código generado para la misma usando cualquier otro lenguaje tal y como si de código escrito usando el primero se tratase. Microsoft ha desarrollado un compilador de

C# que genera código de este tipo, así como versiones de sus compiladores de Visual Basic (Visual Basic.NET) y C++ (C++ con extensiones gestionadas) que también lo generan y una versión del intérprete de JScript (JScript.NET) que puede interpretarlo. La integración de lenguajes esta que es posible escribir una clase en C# que herede de otra escrita en Visual Basic.NET que, a su vez, herede de otra escrita en C++ con extensiones gestionadas.

Gestión de memoria: El CLR incluye un recolector de basura que evita que el programador tenga que tener en cuenta cuándo ha de destruir los objetos que dejen de serle útiles. Este recolector es una aplicación que se activa cuando se quiere crear algún objeto nuevo y se detecta que no queda memoria libre para hacerlo, caso en que el recolector recorre la memoria dinámica asociada a la aplicación, detecta qué objetos hay en ella que no puedan ser accedidos por el código de la aplicación, y los elimina para limpiar la memoria de "objetos basura" y permitir la creación de otros nuevos. Gracias a este recolector se evitan errores de programación muy comunes como intentos de borrado de objetos ya borrados, agotamiento de memoria por olvido de eliminación de objetos inútiles o solicitud de acceso a miembros de objetos ya destruidos.

Seguridad de tipos: El CLR facilita la detección de errores de programación difíciles de localizar comprobando que toda conversión de tipos que se realice durante la ejecución de una aplicación .NET se haga de modo que los tipos origen y destino sean compatibles.

Aislamiento de procesos: El CLR asegura que desde código perteneciente a un determinado proceso no se pueda acceder a código o datos pertenecientes a otro, lo que evita errores de programación muy frecuentes e impide que unos procesos puedan atacar a otros. Esto se consigue gracias al sistema de seguridad de tipos antes comentado, pues evita que se pueda convertir un objeto a un tipo de mayor tamaño que el suyo propio, ya que al tratarlo como un

objeto de mayor tamaño podría accederse a espacios en memoria ajenos a él que podrían pertenecer a otro proceso. También se consigue gracias a que no se permite acceder a posiciones arbitrarias de memoria.

Tratamiento de excepciones: En el CLR todos los errores que se puedan producir durante la ejecución de una aplicación se propagan de igual manera: mediante excepciones. Esto es muy diferente a como se venía haciendo en los sistemas Windows hasta la aparición de la plataforma .NET, donde ciertos errores se transmitían mediante códigos de error en formato Win32, otros mediante HRESULTs y otros mediante excepciones.

El CLR permite que excepciones lanzadas desde código para .NET escrito en un cierto lenguaje se puedan capturar en código escrito usando otro lenguaje, e incluye mecanismos de depuración que pueden saltar desde código escrito para .NET en un determinado lenguaje a código escrito en cualquier otro. Por ejemplo, se puede recorrer la pila de llamadas de una excepción aunque ésta incluya métodos definidos en otros módulos usando otros lenguajes.

Soporte multihilo: El CLR es capaz de trabajar con aplicaciones divididas en múltiples hilos de ejecución que pueden ir evolucionando por separado en paralelo o intercalándose, según el número de procesadores de la máquina sobre la que se ejecuten. Las aplicaciones pueden lanzar nuevos hilos, destruirlos, suspenderlos por un tiempo o hasta que les llegue una notificación, enviarles notificaciones, sincronizarlos, etc.

Distribución transparente: El CLR ofrece la infraestructura necesaria para crear objetos remotos y acceder a ellos de manera completamente transparente a su localización real, tal y como si se encontrasen en la máquina que los utiliza.

Seguridad avanzada: El CLR proporciona mecanismos para restringir la ejecución de ciertos códigos o los permisos asignados a los mismos según su procedencia o el usuario que los ejecute. Es decir, puede no darse el mismo nivel de confianza a código procedente de Internet que a código instalado localmente o procedente de una red local; puede no darse los mismos permisos a código procedente de un determinado fabricante que a código de otro; y puede no darse los mismos permisos a un mismo códigos según el usuario que lo esté ejecutando o según el rol que éste desempeñe. Esto permite asegurar al administrador de un sistema que el código que se esté ejecutando no pueda poner en peligro la integridad de sus archivos, la del registro de Windows, etc.

Interoperabilidad con código antiguo: El CLR incorpora los mecanismos necesarios para poder acceder desde código escrito para la plataforma .NET a código escrito previamente a la aparición de la misma y, por tanto, no preparado para ser ejecutando dentro de ella. Estos mecanismos permiten tanto el acceso a objetos COM como el acceso a funciones sueltas de DLLs preexistentes (como la API Win32)

Como se puede deducir de las características comentadas, el CLR lo que hace es gestionar la ejecución de las aplicaciones diseñadas para la plataforma .NET. Por esta razón, al código de estas aplicaciones se le suele llamar código gestionado, y al código no escrito para ser ejecutado directamente en la plataforma .NET se le suele llamar código no gestionado.

2.2.1.2 Framework Class Library (FCL).

Las necesidades de los desarrolladores han evolucionado mucho en los últimos tiempos debido al continuo incremento en la complejidad de las aplicaciones, esto hace necesario la utilización de marcos de trabajo o frameworks, que permitan a los desarrolladores simplificar el desarrollo de las aplicaciones. La Framework Class Library es un enorme conjunto de clases y estructuras, muy

bien estructuradas, que ofrecen a los desarrolladores una API muy diversa y completa.

2.2.2 ASP.NET.

Desde hace algún tiempo, Microsoft está llevando adelante una estrategia para construir una nueva tecnología tendiente a crear aplicaciones web distribuidas y que aprovechen al máximo las posibilidades que ofrece Internet. Esta tecnología, que lleva el nombre de .NET, y que incluye un nuevo lenguaje denominado C#, una nueva versión de Visual Basic, con el nombre de Visual Basic.Net y otra serie de tecnologías, entre las que se encuentra: ASP.NET, que viene a reemplazar a las Active Server Pages (ASP), logrando el desarrollo de aplicaciones web más dinámicas, con un código más claro y limpio, por ende reusable, multiplataforma y definitivamente más simple, ya que el entorno ASP.NET permite la creación automática de alguna de las tarea más comunes para un creador web, cómo los formularios o la validación de los datos.

Limitación de Lenguajes

ASP.NET incorpora soporte nativo para C#, Visual Basic y JScript. Logrando así dejar atrás las limitaciones ASP que sólo permitía código en VBScript y JScript.

Principales características de ASP.NET

Eficiencia

Desde el principio, uno de los objetivos más importantes del diseño de .NET ha sido su gran rendimiento y nivelación. Para que .NET tenga éxito, las empresas deben estar capacitadas para migrar sus aplicaciones y no sufrir de un rendimiento deficiente debido a la forma en que CLR ejecuta el código. Para asegurarse un óptimo rendimiento, el CLR compila, en algún punto, todos los códigos de aplicaciones en códigos naturales de máquina.

Esta conversión puede hacerse, o bien en el momento en que se ejecuta la aplicación (método por método), o cuando se instala la aplicación por primera vez. El proceso de compilación hará uso automáticamente de todas las características del microprocesador, disponibles en diferentes plataformas, algo que las aplicaciones tradicionales de Windows nunca podrían hacer, a menos que usted cargase distintos binarios para distintas plataformas.

Soporte de Lenguajes

Esta es una de las novedades más importantes que vienen de la mano de ASP.NET. La posibilidad de escribir código en diferentes lenguajes es un alivio para los desarrolladores que en numerosas ocasiones, veían acotadas sus aplicaciones web, al estar obligados a trabajar con VBScript o JScript. ASP.NET soporta la programación en lenguajes potentes cómo, VisualBasic.Net (VB) y C#, el nuevo lenguaje creado por Microsoft con la intención de aprovechar la potencia del C++ y combinarlo con las facilidades que brinda a la programación en Internet un lenguaje como Java.

Contenido y Código, por separado

Muchos desarrolladores de sitios web han tenido que lidiar con el inconveniente de tener que crear la interfaz de usuario y el código ASP todo junto. Esta mezcla de imágenes, botones y tablas en código HTML con pedazos de código en VBScript o Jscript llegaba a ser algo muy molesto para el desarrollador. ASP.NET viene a solucionar este problema, utilizando un criterio similar al que utiliza Visual Basic, es decir, separar la interfaz de usuario con el código.

Compatibilidad con Navegadores ASP.NET permite crear un página web que funcionará correctamente en todos los navegadores. Esta mejora está dada

especialmente por los controles de servidor incluidos en ASP.NET. Cuando una control es procesado, este automáticamente chequea el tipo de navegador que lo está ejecutando, generando una página adecuada para ese navegador.

Código Compilado

ASP.NET ya no interpreta el código como la hace la versión anterior de ASP. Dentro del entorno NGWS (New Generation Windows Services) el código es compilado just-in-time, logrando un enorme aumento en el rendimiento, a través de soporte nativo y servicios de caché. Controles de Servidor Uno de los aspectos más importantes dentro del .NET Framework es su librería de clases.

Esta librería es común en toda la plataforma .NET, lo que le brinda al programador una herramienta ideal para crear aplicaciones multiplataforma, con un considerable ahorro de líneas de código. Los controles de servidor están divididos en dos categorías: Controles Web y Controles HTML.

Posiblemente sean los Controles Web, lo que atractivos para el desarrollador, ya que permiten crear automáticamente controles que realicen tareas importantes en el servidor como validar la entrada de formularios, verificar las capacidades de los navegadores o implementar un sistema de banners rotativos.

Los nuevos Controles Web Forms

ASP.NET adopta el modo de Visual Basic a la hora de utilizar controles. Esto permite separar el código de la interfaz del usuario de forma sencilla y clara. En este pequeño ejemplo, se ve la utilización de la sentencia runat="server" que le indica al servidor ASP.NET que debe procesar el control de servidor, que es en este caso es un Botón.

El atributo Text se utiliza, como sucede en Visual Basic, para establecer el texto que se mostrará en el botón. Esto es consistente con los otros controles. El atributo OnClick finalmente identifica el evento que se ejecutará cuando se haga

clic en el botón. Debido a que es un control del servidor, este procedimiento se ejecuta en el servidor.

Tipos de Controles

- Controles del Servidor HTML, que son los equivalentes del servidor de los elementos HTML.
- Controles de Formulario Web, que planifican aproximadamente elementos HTML individuales.
- Controles de Lista, que planifican grupos de elementos de HTML, que producen grillas o un diseño similar.
- Controles Ricos, que producen ricos contenidos y encapsulan funcionalidad compleja, y producirá HTML puro o HTML y script. Un buen ejemplo de esto es el control Calendario, que provee al usuario un calendario de una sola línea de código.
- Controles de Validación, que son no-visibles, pero permiten el fácil uso de la validación del formulario por parte del servidor y del cliente.
- Controles Móviles, que producen HTML o WML dependiendo del dispositivo con el que se accede a la página.

2.2.3 XML Web Services.

Un servicio web es un servicio, con un interfaz definido y conocido, al que se puede acceder a través de internet. Igual que una página web está definida por un URL (Uniform Resource Locator), un servicio web está definido por un URI (Uniform Resource Identification) y por su interfaz, a través del cual se puede acceder a él. Igual que una página web puede ofrecer cotizaciones de la bolsa, un servicio web que haga lo mismo presentará un interfaz para que se pueda acceder fácilmente, una vez que se conozca el interfaz, a la aplicación. De esta forma, las aplicaciones se convierten en clientes que integran servicios web procedentes de diferentes proveedores, y además, se abre la posibilidad de que se cobre por uso del servicio, no por cada copia de la aplicación vendida. Este

es uno de los aspectos que más gusta a Microsoft: la posibilidad de acabar de una vez por todas con la piratería, a base de alojar partes importantes de las aplicaciones en sus propios servidores, no en el ordenador del cliente.

Los servicios web se dividen en servicios de transporte (los protocolos del nivel más bajo, que codifican la información independientemente de su formato, y que pueden ser comunes a otros servicios), de mensajería, de descripción y de descubrimiento. En la parte más baja se encuentran los servicios de transporte, que establecen la conexión y el puerto usado. Generalmente se usa HTTP, el mismo protocolo que la WWW, pero en se puede usar también SMTP (el mismo protocolo que el correo electrónico), FTP (File Transfer Protocol), o BEEP (blocks extensible exchange protocol), un protocolo específico para servicios web, que, a diferencia de los anteriores, no es cliente-servidor, sino "entre pares"; los dos ordenadores entre los que se establece la comunicación actúan como clientes y servidores a la vez. Es además extensible, y está especificado en XML; por eso se está haciendo mucho más popular para aplicaciones web.

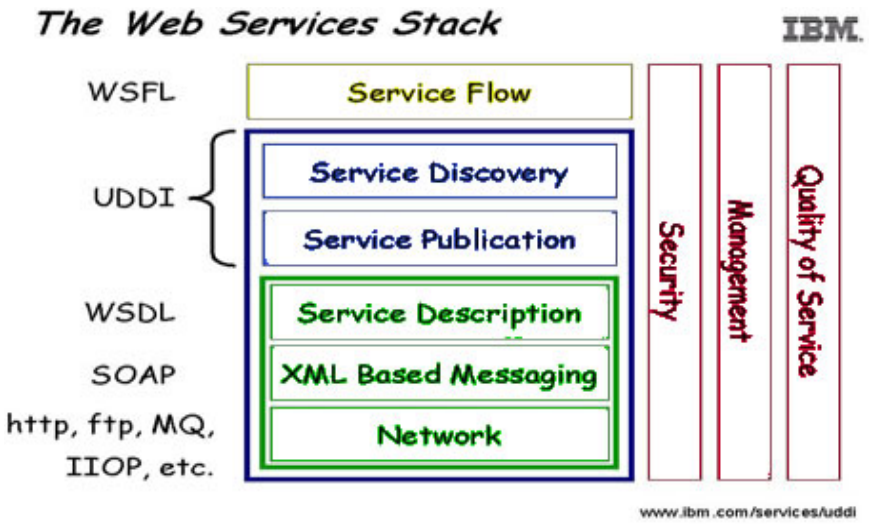


FIGURA 3 *WEB SERVICES*

2.2.4 C# Sharp

Con la idea de que los programadores más experimentados puedan obtener una visión general del lenguaje, a continuación se recoge de manera resumida las principales características de C# Alguna de las características aquí señaladas no son exactamente propias del lenguaje sino de la plataforma .NET en general. Sin embargo, también se comentan aquí también en tanto que tienen repercusión directa en el lenguaje, aunque se indicará explícitamente cuáles son este tipo de características cada vez que se toquen:

Sencillez: C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:

- El código escrito en C# es autocontenido, lo que significa que no necesita de ficheros adicionales al propio fuente tales como ficheros de cabecera o ficheros IDL
- El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad del código.
- No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (.) acceder a miembros de espacios de nombres (::)

Modernidad: C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción foreach que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico string para representar cadenas o la distinción de un tipo bool específico para representar valores lógicos.

Orientación a objetos: Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

Por otro lado y a diferencia de Java, en C# se ha optado por hacer que todos los métodos sean por defecto sellados y que los redefinibles hayan de marcarse con el modificador virtual (como en C++), lo que permite evitar errores derivados de redefiniciones accidentales. Además, un efecto secundario de esto es que las llamadas a los métodos serán más eficientes por defecto al no tenerse que buscar en la tabla de funciones virtuales la implementación de los mismos a la que se ha de llamar. Otro efecto secundario es que permite que las llamadas a los métodos virtuales se puedan hacer más eficientemente al contribuir a que el tamaño de dicha tabla se reduzca.

Orientación a componentes: La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros)

Gestión automática de memoria: Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active –ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente-,

C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción using.

Seguridad de tipos: C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:

- Sólo se admiten conversiones entre tipos compatibles. Esto es, entre un tipo y antecesores suyos, entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacenase una referencia del segundo (downcasting) Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución el CLR y no el compilador, por lo que en realidad el CLR y el compilador colaboran para asegurar la corrección de las conversiones.
- No se pueden usar variables no inicializadas. El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del flujo de control del fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.
- Se puede controlar la producción de desbordamientos en operaciones aritméticas, informándose de ello con una excepción cuando ocurra. Sin embargo, para conseguirse un mayor rendimiento en la aritmética estas comprobaciones no se hacen por defecto al operar con variables sino sólo con constantes (se pueden detectar en tiempo de compilación)
- A diferencia de Java, C# incluye delegados, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se comprueba que los métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.
- Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C#

siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

Instrucciones seguras: Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=); y todo caso de un switch ha de terminar en un break o goto que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.

Sistema de tipos unificado: A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán “objetos”)

2.2.5 SQL Server

Las aplicaciones en red son cada día más numerosas y versátiles. En muchos casos, el esquema básico de operación es una serie de scripts que rigen el comportamiento de una base de datos.[4]

Debido a la diversidad de lenguajes y de bases de datos existentes, la manera de comunicar entre unos y otras sería realmente complicada a gestionar de no ser por la existencia de estándares que nos permiten el realizar las operaciones básicas de una forma universal.

Es de eso de lo que trata el Structured Query Language que no es más que un lenguaje estándar de comunicación con bases de datos. Hablamos por tanto de un lenguaje normalizado que nos permite trabajar con cualquier tipo de lenguaje (ASP o PHP) en combinación con cualquier tipo de base de datos (MS Access, SQL Server, MySQL...).

El hecho de que sea estándar no quiere decir que sea idéntico para cada base de datos. En efecto, determinadas bases de datos implementan funciones específicas que no tienen necesariamente que funcionar en otras.

Aparte de esta universalidad, el SQL posee otras dos características muy apreciadas. Por una parte, presenta una potencia y versatilidad notables que contrasta, por otra, con su accesibilidad de aprendizaje.

2.3 Fundamentación de la metodología utilizada.

Para desarrollar la propuesta que presenta este trabajo, se ha decidido utilizar como metodología el Proceso Unificado de Desarrollo (Racional Unified Process), por sus características especiales y las facilidades que aporta a todo el proceso de desarrollo del software.

2.3.1 El proceso unificado de desarrollo (RUP)

A través de la historia se han desarrollado varios modelos de proceso de software (paradigmas de desarrollo) cada uno con sus ventajas, desventajas y utilidad en algunos tipos de proyectos y problemas. Al igual que cualquier notación, el proceso unificado actúa como un modelo que puede adaptarse a cualquier tipo de proyecto y empresa (grandes y pequeñas) [3]. Las características del proceso unificado de modelado son:

- **Centrado en los Modelos:** Los diagramas son un vehículo de comunicación más expresivo que las descripciones en lenguaje natural. Se trata de minimizar el uso de descripciones y especificaciones textuales del sistema.
- **Guiado por lo casos de uso:** Los casos de uso son el instrumento para validar la arquitectura del software y extraer los casos de prueba.

- **Centrado en la arquitectura:** Los modelos son proyecciones del análisis y el diseño constituye la arquitectura del producto a desarrollar.
- **Iterativo e incremental:** Durante todo el proceso de desarrollo se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo.

2.3.2 Lenguaje unificado de modelado (UML)

El Lenguaje Unificado de Modelado (UML) es una técnica para la especificación de sistemas en todas sus fases. Esta ha sido desarrollado por los más importantes autores en materia de Análisis y Diseño de Sistemas y ha sido usada con éxito en sistemas hechos para toda clase de industrias alrededor del mundo: Salud, Bancos, Comunicaciones, Aeronáutica, Finanzas, etc. [1]

Sin lugar a dudas OOAD (Object Oriented Analysis and Design), implementado con UML (Unified Modeling Language),es la metodología más avanzada en la actualidad. Esta metodología introduce los Casos de Uso, una poderosa herramienta para reducir los riesgos en la definición de requerimientos de sistemas nuevos. Los Casos de Uso sirven como columna vertebral del proceso de desarrollo de aplicaciones y tienen como objetivo garantizar que los resultados se ajusten completamente a las expectativas de los usuarios finales.

UML sirve para hacer modelos que permitan:

- Visualizar como es un sistema o como queremos que sea.
- Especificar la estructura y/o comportamiento de un sistema.
- Hacer una plantilla que guíe la construcción de los sistemas
- Documentar las decisiones que hemos tomado

El modelado sirve no solamente para los grandes sistemas; aún en aplicaciones de pequeño tamaño se obtienen beneficios de modelar, sin embargo, es un

hecho que entre más grande y más complejo es el sistema, el modelado juega un papel más importante. Esto se debe a una razón simple: “Hacemos modelos de sistemas complejos porque no podemos entenderlos en su totalidad”

Hay límites para el entendimiento de la complejidad. A través del modelado reducimos el ámbito del problema de estudio al enfocar solo un aspecto a la vez.

UML puede ser usado extensivamente en: Recopilación de requerimientos, Análisis de aplicaciones, Diseño de sistemas, en pruebas, en implementación, en reingeniería y prácticamente en cualquier actividad de desarrollo que sea susceptible de ser modelada.

Cabe aclarar que aunque UML es orientado a objetos preferentemente, es útil en cualquier modelo tecnológico ya que es independiente de lenguajes de programación o tecnología determinada.

2.4 Herramientas utilizadas para el diseño y construcción del sistema.

2.4.1 Visual Studio .NET.

Visual Studio .NET es una completa herramienta para generar e integrar con rapidez aplicaciones y servicios Web XML, lo que mejora notablemente la productividad del programador y abre las puertas a nuevas oportunidades empresariales.

Incluye funciones de integración para aprovechar aplicaciones existentes en la organización y optimizar los procesos empresariales con clientes y socios. La arquitectura abierta permite a los programadores utilizar cualquier lenguaje orientado a Microsoft .NET Framework y aprovechar los conocimientos de programación actuales, ahorrando así los cursos de reciclaje largos y costosos.

Visual Studio .NET está basado en la más reciente plataforma de servidor de Microsoft Windows®, lo que incorpora escalabilidad, confiabilidad y seguridad a las aplicaciones. Se han simplificado la administración y la implementación de las aplicaciones en un ambiente de producción, reduciendo así los costes totales del ciclo.[2]

2.4.2 Microsoft SQL Enterprise Manager.

SQL Server 2000 se creó para aportar todas las herramientas y la funcionalidad para crear y mantener aplicaciones de web potenciadas mediante una base de datos.

Es la plataforma idónea para el lanzamiento de esta nueva generación de aplicaciones automáticas. El rendimiento, la fiabilidad y la escalabilidad se combinan con potentes herramientas de gestión convirtiendo a SQL Server 2000 en la mejor base de datos del mercado para todos los tipos de aplicaciones de procesos empresariales.

2.4.3 Embarcadero ERStudio.

Embarcadero ER/Studio is a data modeling application for logical and physical database design and construction. ER/Studio's progressive interface and processes have been logically organized to effectively address the "ease-of-use" issues that have plagued data modeling tools for the past decade. Its powerful, multi-level design environment addresses the everyday needs of database administrators, developers, and architects who build and maintain large, complex database applications. The application equips the user to create, understand,

and manage the mission-critical database designs within an enterprise. ER/Studio offers the following functionality: [6]

Erwin studio es una herramienta para el diseño y la construcción de modelos ya sea lógico o físico de bases de datos. Proporciona una interface progresiva lógicamente organizados que lo hacen una herramienta de fácil uso. La aplicación permite al usuario crear, entender y administrar la critica misión de hacer el diseño de la base de datos y presenta las siguientes funcionalidades:

- Poderosas capacidades de diseño lógico.
- Sincronización bidireccional de diseños físicos y lógicos.
- Construcción automática de la base de datos.
- Realiza la ingeniería inversa de las bases de datos.
- Documentación y reportes en html.

2.4.4 Rational Rose.

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML.

Esta herramienta propone la utilización de cuatro tipos de modelos para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Rational Rose utiliza un proceso de desarrollo iterativo controlado (*controlled iterative process development*), donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos.

Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades.

Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.

Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño.

Rational también ofrece una alternativa a la plataforma .NET que permite crear aplicaciones de nuevas generaciones fiables y productivamente.

2.5 La solución.

Según lo antes expuesto en este capitulo se propone:

- Construir una aplicación web acorde al entorno donde se usara el sistema, a su vez esto permitirá a todos los usuarios acceder a la versión más actualizada del sistema, con solo usar el navegador.
- Establecerla en la plataforma .NET debido a las características de esta plataforma, ya que esta es probablemente la mejor opción hoy en día para las aplicaciones web.

- Hacerla en el lenguaje de programación C#, por ser el lenguaje hecho especialmente para ser usado con la plataforma .NET
- Utilizar el gestor de bases de datos SLQ Server 2000. Uno de los mejores gestores de bases de datos que existen actualmente.

2.6 Conclusiones.

En este capítulo se han analizado las tecnologías actuales y se ha profundizado en conceptos imprescindibles para la comprensión de la solución de este trabajo. También se han fundamentado la elección de cuales herramientas se utilizaran para el desarrollo de la aplicación. Finalmente se ha llegado a la conclusión de que el sistema se desarrollará sobre la plataforma .NET, usando como lenguaje de programación C#, y SQL Server 2000 como gestor de bases de datos.

CAPÍTULO 3

DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción.

En este capítulo se describe la solución propuesta para el sistema, para ello se describen los procesos del negocio. También se exponen los requisitos funcionales y no funcionales del sistema que se propone, permitiendo hacer una concepción general del sistema y representar a través del diagrama de caso de uso, las relaciones de los actores del sistema y la secuencia de acciones con las que interactúan.

3.2 Reglas del negocio a considerar.

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio.

3.2.1 Conceptos asociados al modelo de negocios.

Tiempo de Máquina: Tiempo asignado a un estudiante en un puesto de trabajo dentro de un laboratorio determinado.

Técnico: Persona a cargo de un laboratorio.

Jefe de Área: Persona que dirige un área de determinada en nuestro caso un docente.

Encargado de las Reservaciones: Persona encargada de las reservaciones.

Usuario: Todo estudiante o profesor en la UCI.

Solicitud de tiempo de máquina: Petición por parte de un estudiante de un servicio en un laboratorio.

Los estudiantes y profesores con su solapín deben solicitar la reservación de un tiempo de máquina en un horario determinado solo en los laboratorios de su facultad.

3.3 Descripción de los procesos del negocio actual.

Actualmente labora un empleado en cada planta donde existen laboratorios docentes que tienen como objetivo de trabajo la recepción y reservación de las solicitudes de tiempo de máquina por parte de los estudiantes, estas solicitudes deben ser personalmente en esa área. Una vez realizada la solicitud el estudiante espera que le den respuesta a su solicitud mientras el empleado realiza las operaciones correspondientes, como verificar si hay capacidad o no, y a que facultad. Además existen los reportes de desperfectos en los laboratorios.

Los profesores deben acudir a la vicedecana de su facultad con anterioridad y comunicarle su solicitud de reservación de un laboratorio en una fecha dada. Luego de la vicedecana verificar disponibilidad en caso de ser afirmativa la respuesta, deberá enviar un correo a la dirección de los laboratorios de su facultad y comunicar el cambio.

Todos estos procesos se realizan manualmente.

3.3.1 Actores y trabajadores del negocio actual.

Actores del Negocio	**Justificación**
Estudiantes	Es el que solicita el tiempo de máquina.
Profesores	Es el que solicita la reservación de un laboratorio.

Trabajadores del Negocio	**Justificación**
Jefe de Área	Encargado de la distribución y organización de los

	medios entiéndase laboratorios.
Técnico	Personal encargado de los laboratorios.
Encargado de las Reservaciones	Personal encargado de las reservaciones.

3.3.2 Descripción de los Casos de Usos del Modelo de Negocio Actual.

Los casos de uso proporcionan a los analistas del sistema un medio intuitivo de captura de los requisitos funcionales para cada usuario. Estos le permiten a los desarrolladores y a los clientes lleguen a tener una visión común sobre el problema en sí.

3.3.2.1 Diagrama de Casos de Uso del modelo de negocio actual.

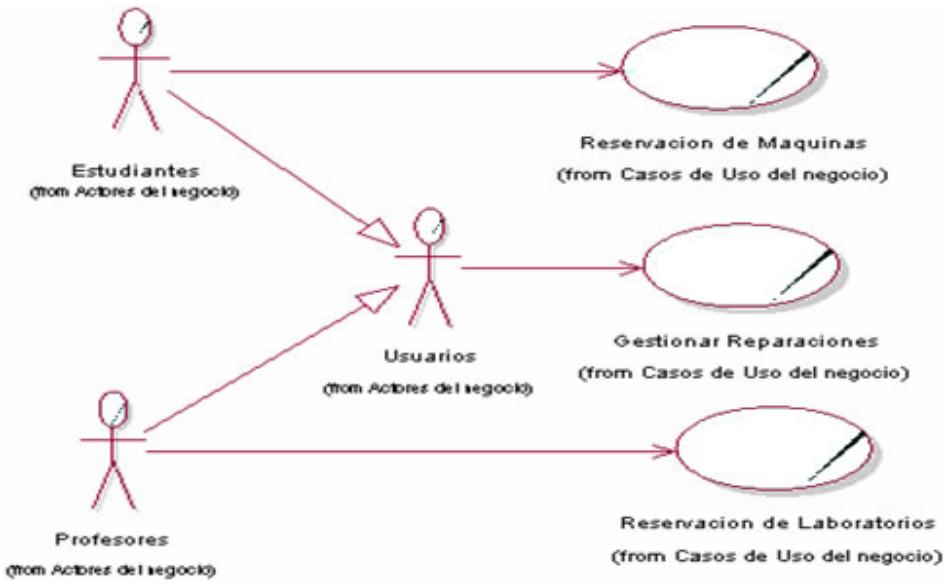


FIGURA 4. *DIAGRAMA DE CASOS DE USO DEL NEGOCIO.*

3.3.2.2 Expansión de los casos de uso.

Caso de Uso del Negocio	**Reservar Máquina**
Actores	Estudiante
Propósito	Hacer la reservación de un tiempo de

	máquina en un horario dado.
Resumen: El caso de uso se inicia cuando el estudiante solicita un puesto de trabajo, el empleado debe verificar disponibilidad y de existir esta entonces procede a pedir los datos del estudiante, con estos llena el modelo que contiene la máquina, número de solapín y horario de reservación luego informa el resultado.	

Acción del actor		**Respuesta del proceso de negocio**	
1	El estudiante solicita un puesto de trabajo (máquina).	2	El empleado verifica disponibilidad.
		3	En caso de que exista alguna disponibilidad sino va al paso 7.
		4	El empleado pide los datos para llenar el modelo.
5	El estudiante le da los datos requeridos	6	El empleado llena el modelo.
		7	El empleado informa al estudiante el resultado.
Prioridad		Alta(De que se realice una solicitud, depende todo lo demás en el sistema)	
Mejoras		La automatización de este proceso de evaluación reducirá el tiempo de respuesta y permitirá a los Estudiantes mejorar su gestión. El Estudiante no tendrá que interactuar con el empleado, pues este último desaparecerá al automatizarse este proceso.	
Otras secciones			

Diagrama de actividades

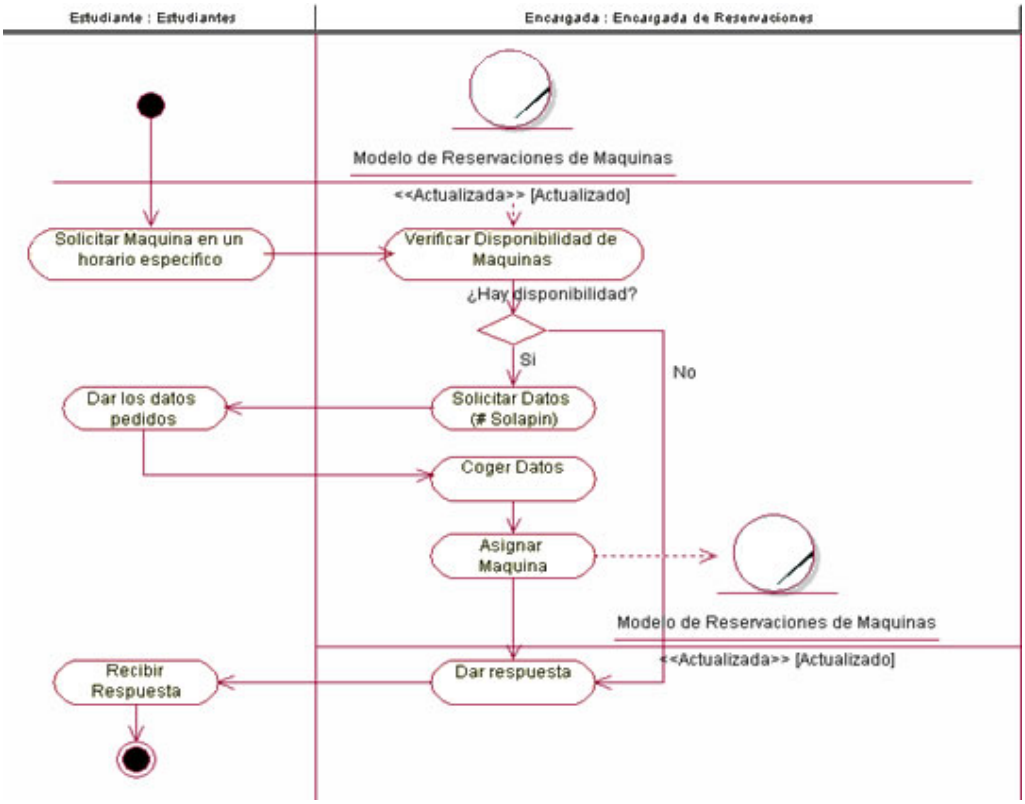


FIGURA 5 *DIAGRAMA DE ACTIVIDADES DEL CU "RESERVAR MÁQUINA*

Caso de Uso del Negocio	**Gestionar Reparación**
Actores	Usuario
Propósito	Gestionar la Reparación de un puesto de trabajo.
Resumen: El usuario detecta la falla y la comunica al técnico quien verifica si es o no una falla, de serlo llena un modelo con los detalles, este modelo después de terminado su turno lo entrega al jefe de turno quien hace un modelo general con los desperfectos encontrados y lo entrega a su jefe de área de copextel correspondiente.	

Acción del actor		**Respuesta del proceso de negocio**	
1	El usuario detecta la falla.	2	El técnico la verifica, de no ser realmente no hace nada.
		3	Llena el modelo con los datos de la máquina, el fallo etc.

		4	Luego de terminado su horario entrega el modelo al jefe de turno.
		5	El jefe de turno hace un modelo con todos los desperfectos encontrados en su turno y lo entrega a su jefe de área de copextel.
Prioridad		Alta de esto depende que se repare con una mayor rapidez	
Mejoras		La automatización de este proceso disminuirá el tiempo de respuesta, ya que el jefe de turno no tendrá que esperar que el turno termine para saber los desperfectos y reportarlos.	
Otras secciones			

Diagrama de actividades

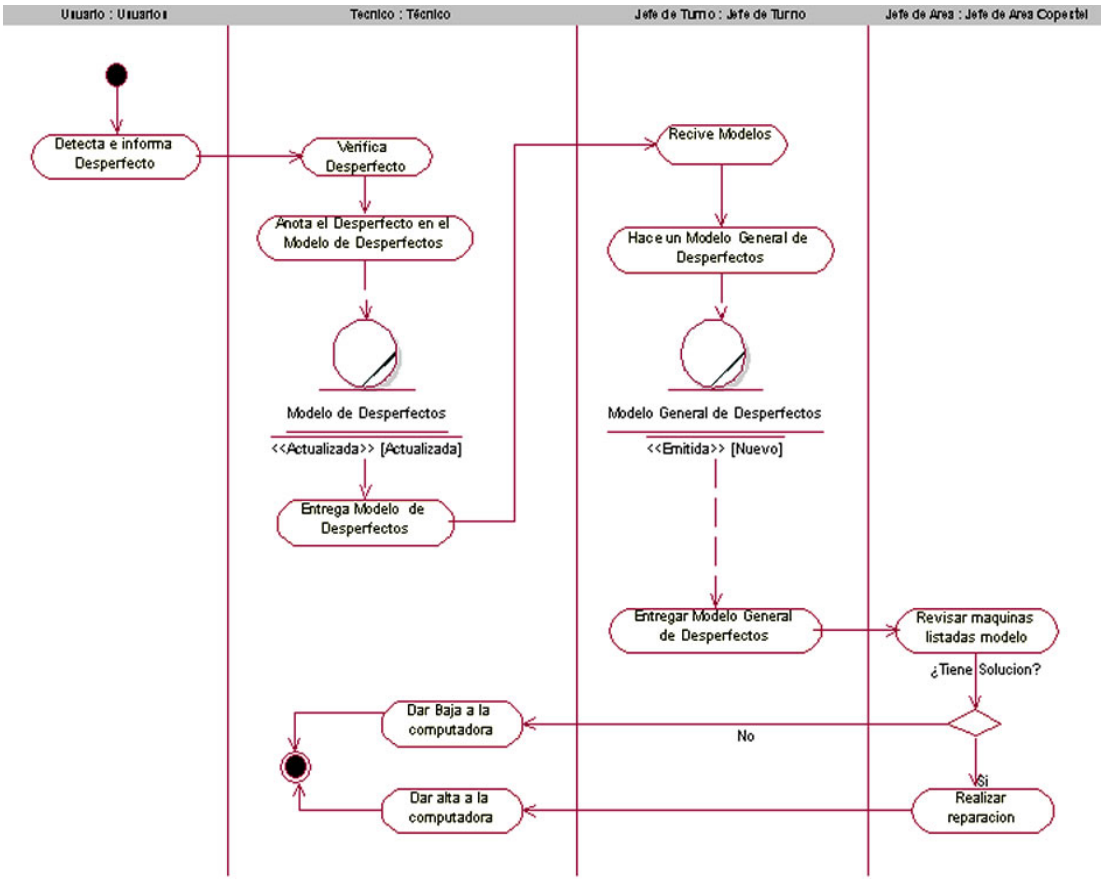


FIGURA 6 *DIAGRAMA DE ACTIVIDADES DEL CU "GESTIONAR REPARACIONES"*

Caso de Uso del Negocio	**Reservar laboratorio**
Actores	Profesor
Propósito	Hacer la reservación de un laboratorio en un horario dado.
Resumen: El caso de uso se inicia cuando el profesor solicita un laboratorio un día determinado a su vicedecana, esta verifica la disponibilidad de los laboratorios y de haber un horario libre envía un correo a la dirección de los laboratorios informando el cambio y da la respuesta al profesor. Este proceso es muy lento y no siempre es efectivo y se debe solicitar con varios días de antelación para lograr su correcto funcionamiento.	

Acción del actor		**Respuesta del proceso de negocio**	
1	El profesor solicita un laboratorio	2	La vicedecana recibe la solicitud.
		3	La vicedecana verifica la disponibilidad para ese día y horario solicitado, en caso de que exista un espacio libre se le otorga el tiempo pedido sino va al paso 6.
		4	La vicedecana envía un correo a la dirección de los laboratorios de su docente informando el cambio.
		5	En caso de que no exista alguna disponibilidad va al paso 5.
		6	La vicedecana informa al profesor el resultado.
7	El profesor recibe la respuesta.		
Prioridad			
Mejoras		Con la automatización de este proceso se agilizará de una forma	

	extraordinaria la demora de la respuesta (comparada con la forma actual del proceso de reservación de laboratorios).
Otras secciones	

Diagrama de actividades

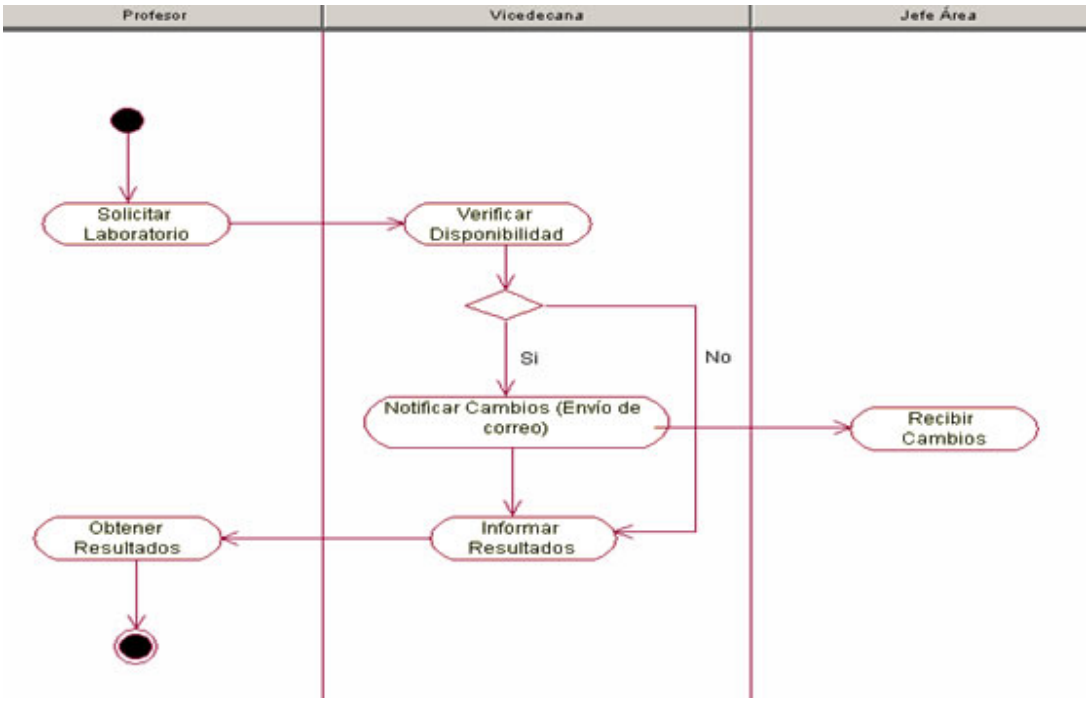


FIGURA 7 *DIAGRAMA DE ACTIVIDADES DEL CU "RESERVAR LABORATORIOS"*

3.3.3 Diagrama de clases del Modelo de Objetos.

El diagrama de clases es usado para describir el Modelo de Objetos de nuestro negocio, muestra la participación de los trabajadores y entidades del negocio y la relación entre ellos.

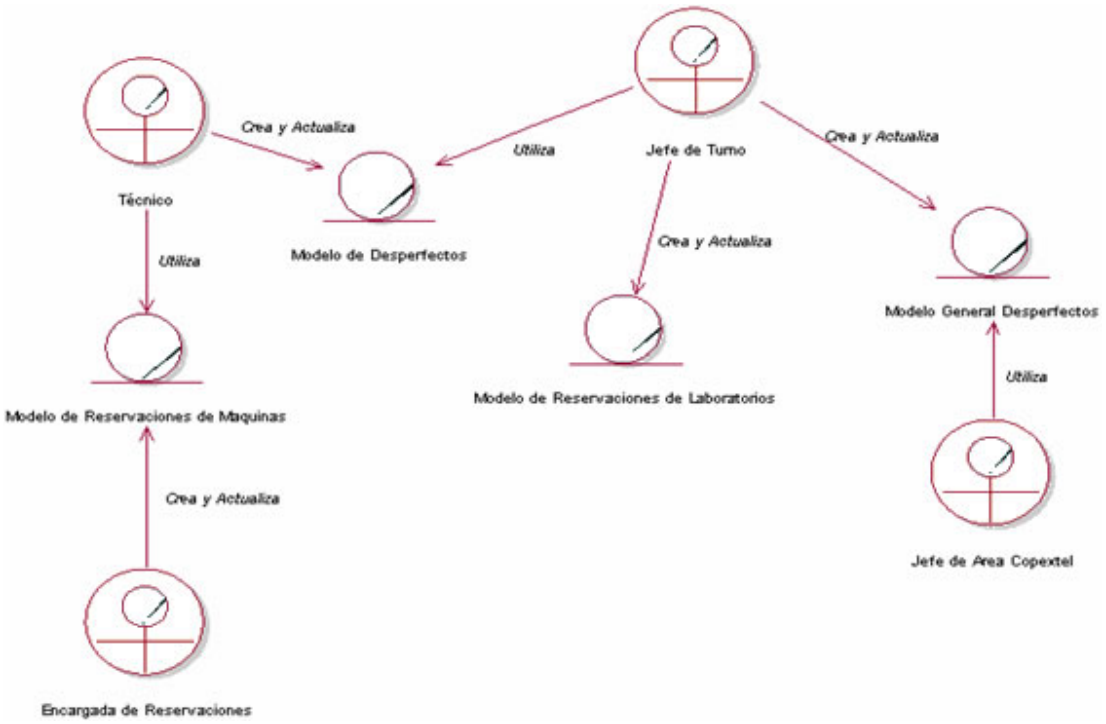


FIGURA 8 *DIAGRAMA DE CLASES DEL MODELO DE OBJETOS DEL NEGOCIO.*

3.4 Requisitos funcionales.

1. **Reservar Tiempo de Máquina.**

1.1 Datos necesarios

1.1.1 Nombre usuario que solicita el tiempo de máquina.

1.1.2 Facultad a la que pertenece.

1.1.3 Horario Solicitado para la reservación.

1.2 Verificar que no este reservado anteriormente.

1.3 Una vez realizada la recepción de datos se almacenan los siguientes datos.

1.3.1 Número de identificación de la persona solicitante.

1.3.2 Puesto asignado.

1.3.3 Horario y fecha de la reservación otorgada

1.3.4 Una máquina puede estar reservada dos veces en diferentes horarios para una misma fecha.

2. **Reservar Laboratorio.**

2.1 Datos necesarios

2.1.1 Nombre usuario que solicita el laboratorio.

2.1.2 Facultad a la que pertenece.

2.1.3 Horario Solicitado para la reservación.

2.2 Verificar que no este reservado anteriormente.

2.3 Una vez terminada la recepción de datos se procederá a guardar los siguientes datos.

2.3.1 Número de identificación de la persona solicitante.

2.3.2 Laboratorio asignado.

2.3.3 Fecha.

3. **Visualizar Reservaciones**

3.1 Datos necesarios

3.1.1 Docente a visualizar.

3.1.1 Horario a visualizar

3.1.2 Laboratorio específico que se desea visualizar y si este está reservado por un profesor en ese mismo horario

4. **Gestionar Reparaciones.**

4.1 Datos necesarios

4.1.1 Nombre del medio.

4.1.2 Descripción del problema.

4.2 Una vez obtenidos los datos se procederá a guardar los datos.

4.2.1 id del medio.

4.2.2 Descripción del problema.

4.2.3 Horario del reporte.

4.3 Luego se procederá a actualizar el estado del medio.

5. **Gestionar Distribución de Técnicos**

5.1 Datos necesarios.

5.1.1 Docente de los laboratorios a asignar.

5.1.2 Laboratorio a asignar.

5.1.3 Técnico a ser asignado dentro de ese docente.

5.2 Una vez obtenidos los datos se procederá a guardar los siguientes datos.

5.2.1 id del laboratorio.

5.2.2 id del técnico.

6. **Administrar Docentes**

6.1 Datos necesarios.

6.1.1 Nombre del nuevo docente.

6.1.2 Jefe de Área del nuevo docente.

6.2 Una vez obtenidos los datos se guardara en la base de datos lo siguiente.

6.2.1 id del nuevo docente.

6.2.2 Nombre del nuevo docente.

6.2.3 id del jefe de área asignado.

6.3 Si el jefe de área no es técnico se procederá a guardar sus datos además en la tabla de los técnicos.

7. **Gestionar distribución de Facultades**

7.1 Datos necesarios.

7.1.1 Nombre del docente donde radicará la facultad.

7.1.2 Nombre de la facultad a asignar.

7.2 Si la facultad estaba asignada se eliminará todas sus asignaciones anteriores.

7.3 Una vez obtenidos los datos se procederá a guardar en la base de datos.

7.3.1 id docente asignado.

7.3.2 id facultad.

7.3.3 Nombre Facultad.

8 **Gestionar distribución de laboratorios por facultad**

8.1 Datos necesarios.

8.1.1 Docente de esa facultad.

8.1.2 Nombre de la facultad.

8.1.3 Nombre del laboratorio.

8.2 Una vez obtenidos los datos se procederá a guardar en la base de datos lo siguiente:

8.2.1 id de la facultad.

8.2.2 id del docente.

8.2.3 Nombre del laboratorio.

8.3 Se procederá a la creación de 31 puestos de trabajo en ese laboratorio, para eso se guardarán los siguientes datos en la base de datos.

8.3.1 id del laboratorio.

8.3.2 Nombre de la Máquina o puesto de trabajo.

8.4 Se procederá a añadir los nuevos medios con los siguientes datos.

8.4.1 id del medio

8.4.2 id estado del medio. OK por defecto.

8.4.3 id del tipo de medio. (Laboratorio o Máquina)

9. **Gestionar Distribución de Técnicos x Laboratorio**

9.1 Datos Necesarios

9.1.1 Nombre del técnico

9.1.2 Laboratorio a Asignar

9.1.3 Verificar que el laboratorio este en el docente donde este asignado el técnico.

9. 2 Una vez obtenidos los datos se procederá a guardar en la base de datos lo siguiente:

9.2.1 id del laboratorio.

9.2.2 id del técnico

10. Eliminar Recursos.

10.1 Datos Necesarios.

10.1.1 Tipo de recurso.

10.1.2 id del recurso.

10.2 Una vez reunidos los datos en dependencia del tipo se procederá a la eliminación del recurso según la lógica seguida en la implementación.

11. Autenticarse

11.1 Datos necesarios.

11.1.1 Usuario del dominio.

11.2 Una vez tomado y verificado el usuario el sistema procederá a su redireccionamiento a la página asignada a su rol dentro del sistema.

3.5 Requisitos no funcionales.

Cualidades que debe cumplir el producto.

Se aplicará el estándar del proyecto UCI Ciudad Digital para el diseño de interfaz de usuario. Esta será legible, simple de usar, e interactiva.

Usabilidad

De fácil uso para todos los posibles clientes, incluyendo personas con pocos conocimientos en el uso de las PCs.

Manual del usuario. Material de entrenamiento. Ayuda en Línea, soportada por páginas Web, que estará disponible al usuario en todo momento.

Soporte:

El sistema será de fácil instalación, adaptable a numerosas plataformas y de fácil mantenimiento.

Portabilidad:

Facilidad de adaptación a diferentes ambientes sin necesidad de usar otros medios además de los previstos. Se requiere de Windows como plataforma.

Seguridad:

Se implementarán varios niveles de usuarios con permisos que correspondan al rol que desempeñado en la aplicación.

Se implementarán mecanismos de tolerancias a faltas. Predicción de fallos.

Se hará un mecanismo de protección contra los fallos. El sistema debe ser capaz en pocos segundos recuperarse de un fallo de una operación.

El sistema permite que los usuarios pocos familiarizados con el sistema perciban sin problemas las salidas del mismo

Legales:

El sistema debe cumplir con lo establecido en las leyes del sistema de seguridad y protección de nuestro país.

Software:

Sistemas Operativos de la familia Windows (Windows NT 4.0, Windows 9.x, Windows 2000). La aplicación se realizará en un ambiente Web, la Base de Datos es independiente de la aplicación.

Hardware:

Es compatible con procesadores x486 o superior, con 64 MB de memoria RAM. Un mínimo de 200 MB de espacio disponible en disco.

Soportado por una red hasta 100 Mbps de velocidad.

Restricciones:

Para la documentación del sistema se utilizó para la realización del análisis y el diseño del sistema la metodología UML (Unified Modeling Language), como herramienta de apoyo a este se utilizó el Rational Rose. Su desarrollo se llevará a cabo con .Net, y como gestor de base de datos se utilizará el SQL Server.

3.6 Descripción del sistema propuesto.

Con el objetivo de dar solución a los requisitos planteados en este trabajo, se propone un sistema para la Reservación de Laboratorios.

El sistema define todo lo referente a los laboratorios y a las principales operaciones que se realizan dentro de ellos. Para solicitar un tiempo de máquina el estudiante debe escoger un horario, un laboratorio y el puesto dentro de este donde quiere reservar su tiempo de máquina. Los profesores por su parte solo deben escoger el laboratorio donde desean reservar por defecto reservara de 8:30 PM hasta las 10:30 PM. En ambos casos la respuesta es inmediata y además el sistema lo confirmara con el envío de un correo con los datos de la reservación.

Los técnicos por su parte son los responsables de reportar desperfectos en los puestos de trabajo y en los laboratorios inhabilitándolos para su reservación hasta que el Jefe de Área correspondiente los reporte en buen estado, además los técnicos son los que verifican el cumplimiento de los tiempos de máquina para esto contarán con un reporte donde se muestran las reservaciones según el horario y el laboratorios.

Los jefes de área son los encargados de definir los docentes, las facultades asignadas a los docentes y sus laboratorios además el estado actual de los medios así como la distribución de técnicos por laboratorio.

Además se han definido un conjunto de roles para restringir el acceso a las partes del sistemas según el tipo de usuario que ingrese al sistema estos son estudiante, profesor, Jefe Área y técnico.

3.6.1 Actores del sistema.

Actores del SISTEMA	Justificación
Estudiante	Cualquier estudiante de la UCI que solicite un servicio.
Jefe de área	Persona al mando de un área de laboratorios de la UCI.
Profesor	Cualquier profesor de la UCI que solicite un servicio.
Técnico	Cualquier técnico de la UCI.

3.6.2 Modelo de casos de uso del sistema.

El modelado por Casos de Uso es la técnica más efectiva y la más simple para modelar los requisitos del sistema desde la perspectiva del usuario. Los Casos de Uso se utilizan para modelar cómo un sistema o negocio funciona, o cómo los usuarios desean que funcione el futuro sistema. No es realmente una aproximación a la orientación a objetos; es una forma de modelar procesos.

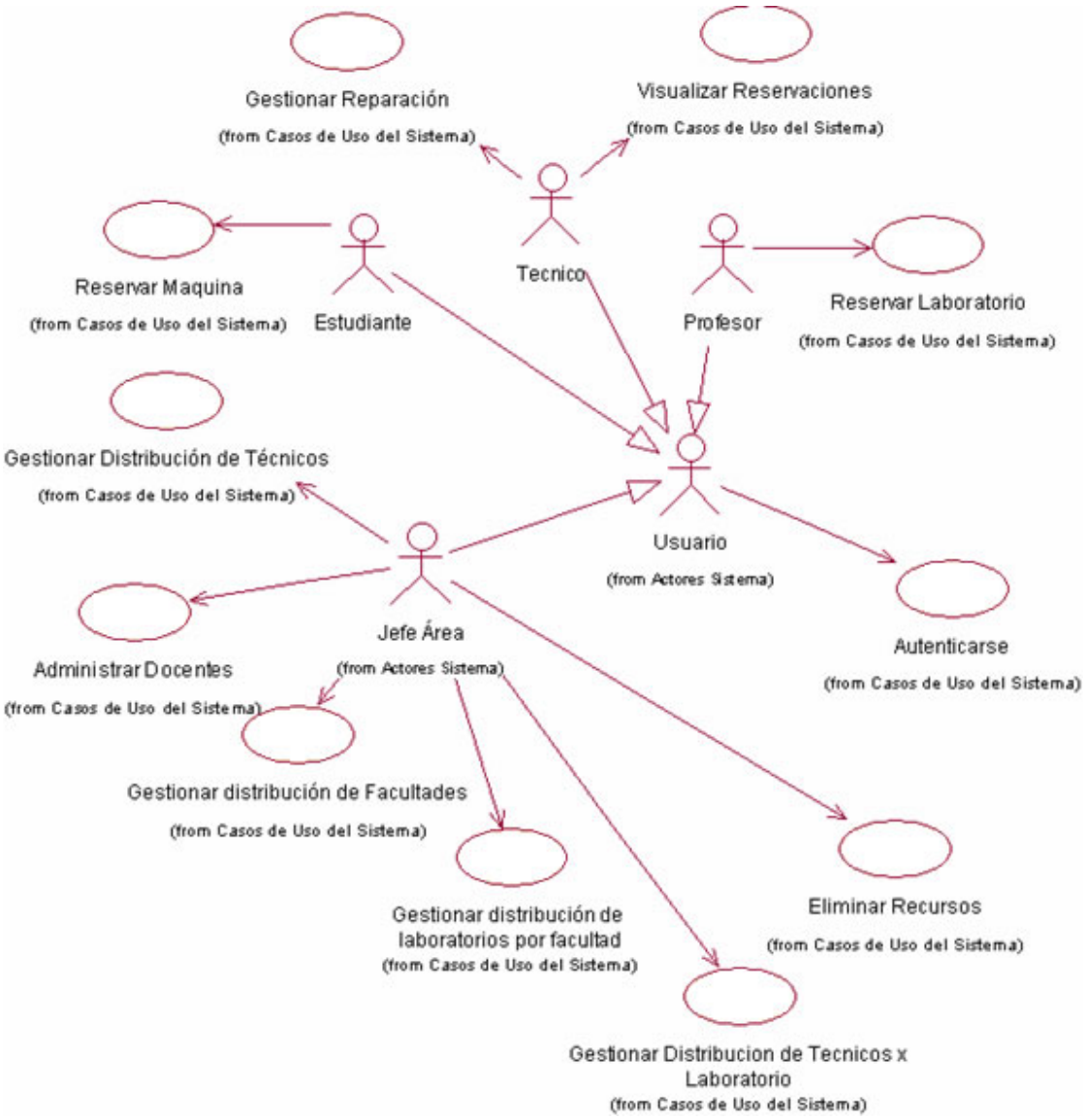
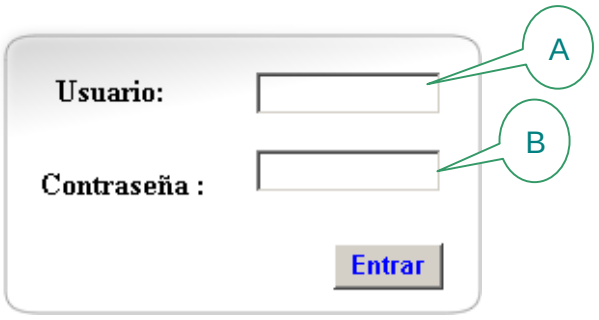


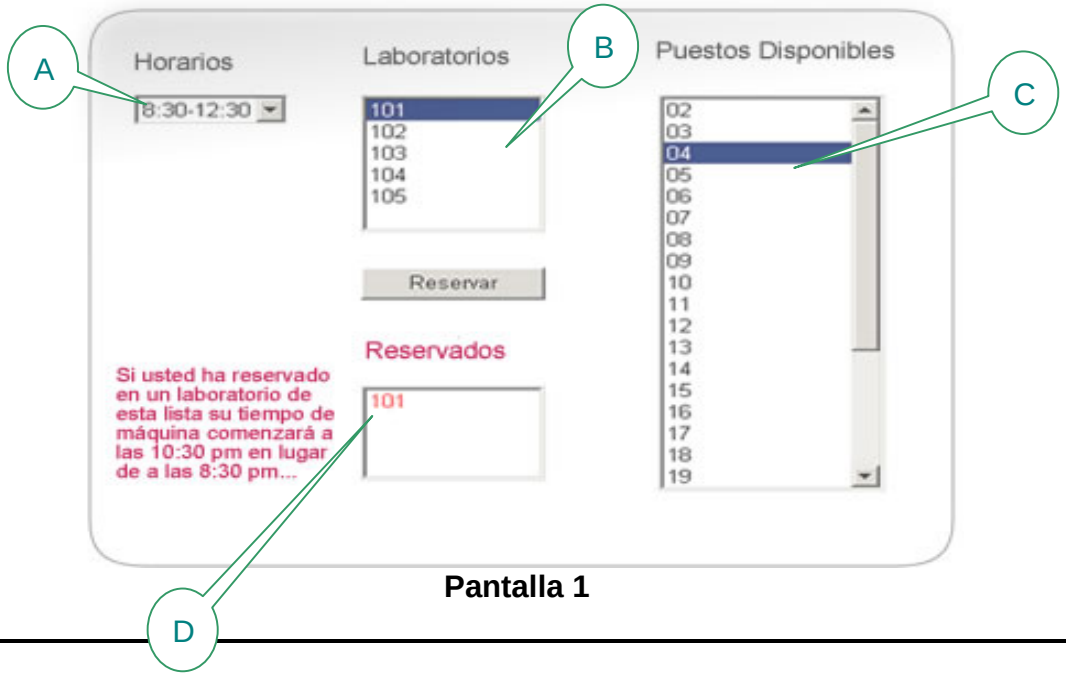
***FIGURA 9**. DIAGRAMA DE CASOS DE USO DEL SISTEMA.*

3.6.3 Expansión de los casos de uso del sistema.

Aquí se describe la secuencia en detalles que siguen los actores para realizar los procesos a través del sistema.

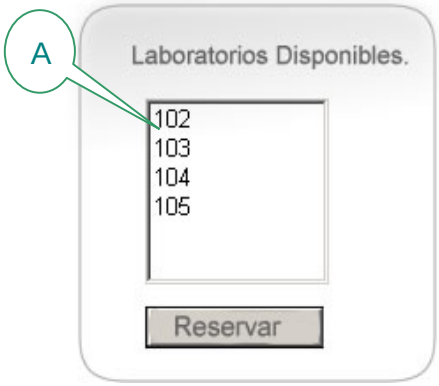
~~Caso de Uso:~~	**Autenticarse**
Actor(es):	Usuario(Inicia)
Propósito:	Que el usuario se autentique en el sistema
Resumen:	El caso de uso se inicia cuando el usuario accede a la página inicial con el propósito de realizar alguna acción dentro del sistema, este mostrará la opción de entrar con usuario actual o entrar usuario ***(Pantalla 1,A)*** y contraseña (***Pantalla 1,B***) una vez escogido la opción o entrado los datos el sistema procederá a su autenticación de ser positiva tomará algunos datos y luego enviará al usuario a la página que tenga asignada dentro del sistema.
Referencias:	R11
Precondiciones:	
Poscondiciones:	
Requisitos Especiales:	
Prototipo	 **Pantalla 1**

~~Caso de Uso:~~	**Reservar Tiempo de Máquina.**
Actor(es):	Estudiante(Inicia)
Propósito:	Que el estudiante reserve un tiempo de máquina.
Resumen:	El caso de uso se inicia cuando un estudiante accede al sistema este reúne todos los datos que necesita de este y le muestra la página donde el estudiante debe terminar la solicitud escogiendo los datos de su solicitud de reservación como horario, laboratorio y puesto de trabajo. (**Pantalla 1**). Una vez reunidos los datos se procede a verificar que no haya reservado en ese mismo horario, y que no exceda la norma de reservaciones por día que en estos momentos es dos. Se muestra: -Horarios a reservar (***Pantalla 1,A***) -Laboratorios en buen estado de su facultad (***Pantalla 1,B***) -Máquinas disponibles en ese horario para el laboratorio escogido (***Pantalla 1,C***) -Además se muestra un lista con los laboratorios reservados por los profesores (***Pantalla 1,D***)
Referencias:	R1
Precondiciones:	Debe haber al menos un laboratorio en buen estado y un puesto de trabajo libre para ese horario.
Poscondiciones:	
Requisitos Especiales:	
Prototipo	



Pantalla 1

~~Caso de Uso:~~	**Reservar Laboratorio.**
Actor(es):	Profesor(Inicia)
Propósito:	Que el profesor reserve un laboratorio un día determinado.
Resumen:	El caso de uso se inicia cuando el profesor accede al sistema, este recopila los datos necesarios de ese profesor y le muestra la página donde el profesor escogerá los datos que faltan para realizar la reservación (***Pantalla 1***). Se muestra: -Laboratorios Libres (***Pantalla 1,A***)
Referencias:	R2
Precondiciones:	Debe haber al menos un laboratorio en buen estado y sin reservar de los asignados a la facultad del profesor.
Poscondiciones:	
Requisitos Especiales:	
Prototipo	



Pantalla 1

~~Caso de Uso:~~	**Visualizar Reservaciones**
Actor(es):	Técnico(Inicia)
Propósito:	Que el Técnico visualice las reservaciones echas por estudiantes y profesores un día en específico.
Resumen:	El caso de uso se inicia cuando el técnico accede al sistema, este recopila los datos necesarios y le muestra la página donde escogerá los datos que faltan para visualizar las reservaciones (***Pantalla 1***) Se muestra. -Horario a visualizar (***Pantalla 1,A***) -Laboratorio a visualizar (***Pantalla 1,B***) -Si el Laboratorio escogido ha sido reservado por un profesor (***Pantalla 1,C***) -Una lista de los estudiantes que reservaron en ese laboratorio

	(***Pantalla 1,D***)
Referencias:	R3
Precondiciones:	Debe haber reservaciones hechas para ese día.
Poscondiciones:	
Requisitos Especiales:	
Prototipo	

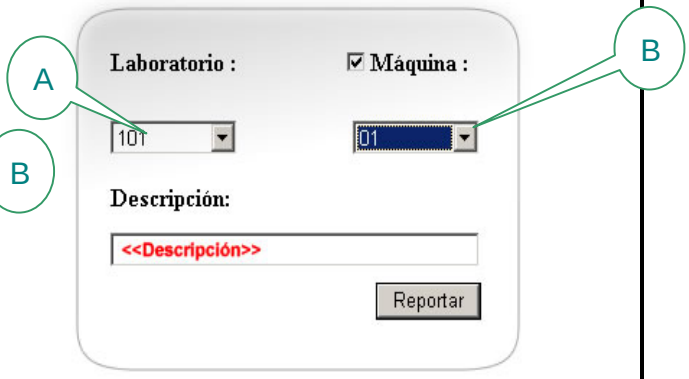
Pantalla 1

~~**Caso de Uso:**~~	**Gestionar Reparaciones.**
Actor(es):	Técnico (Inicia), Jefe de Área.
Propósito:	Llevar un control de los medios fuera de servicio dentro de los docentes.
Resumen:	El caso de uso se inicia cuando el técnico reporta un fallo en un medio bajo su responsabilidad para esto accede sistema y escoge reportar desperfecto una vez en la página (***Pantalla 1***) se muestra dos opciones al seleccionar Laboratorio se muestra una lista de laboratorios en buen estado asignados a él (***Pantalla 1,A***) y un espacio para la descripción del problema (***Pantalla 1,B***), si selecciona máquina (***Pantalla 2***) se muestra una lista con los laboratorios (***Pantalla 2,A***) en buen estado, y una lista de las máquinas en buen estado que están en él (***Pantalla 2,B***), una vez escogido el laboratorio y o las máquinas con problema el sistema actualizará el estado del laboratorio o la máquina seleccionado. Luego de verificar la correcta reparación y funcionamiento del medio anteriormente defectuoso el Jefe de Área accede al sistema, y escoge actualizar medios fuera de servicio una vez en la página (***Pantalla 3***), se muestra una lista de los laboratorios en buen estado (***Pantalla 3,A***), así como una lista de las máquinas fuera de servicio dentro de estos (***Pantalla***

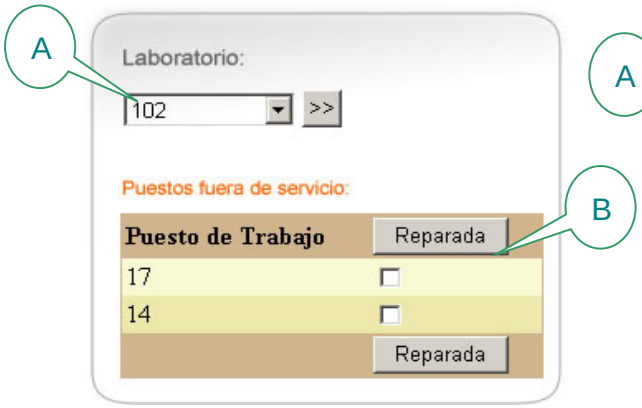
	3,B), de otra manera se muestran los laboratorios fuera de servicio (***Pantalla 4,A***). Una vez escogidas las actualizaciones el sistema tomará los datos y actualizará el estado del medio en la base de datos.
Referencias:	R4
Precondiciones:	Debe haber al menos un laboratorio o un puesto de trabajo en mal estado.
Poscondiciones:	
Requisitos Especiales:	

Prototipo

Pantalla 1



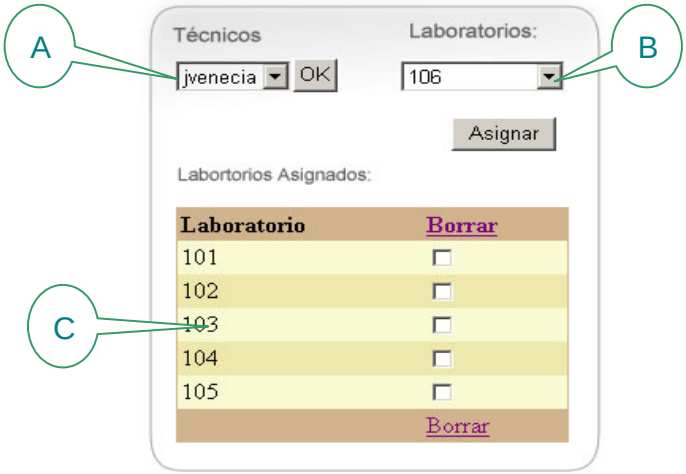
Pantalla 2



Pantalla 2

Pantalla 3

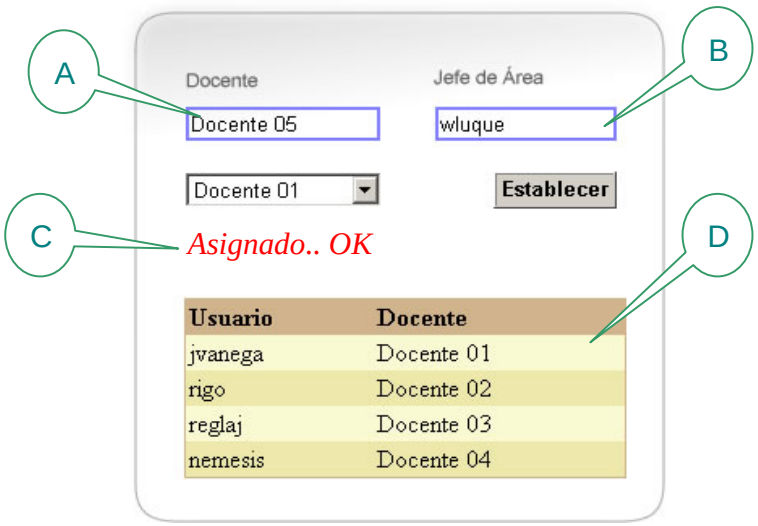
~~Caso de Uso:~~	**Gestionar Distribución de Técnicos.**
Actor(es):	Jefe Área(Inicia)
Propósito:	Que el Jefe de Área realice cambios en la distribución de los técnicos por laboratorio.
Resumen:	El caso de uso se inicia cuando el Jefe de Área desea realizar un cambio o una asignación nueva de un técnico a un laboratorio, para esto debe escoger el técnico (***Pantalla 1,A)*** Luego seleccionar el laboratorio al cual será asignado (***Pantalla 1,B)*** el sistema coge estos datos e inserta o modifica la distribución existente luego informa el resultado. Se muestra además: -Una tabla con los técnicos asignados a ese laboratorio (***Pantalla 1,C)***
Referencias:	R5
Precondiciones:	Debe haber técnicos en la base de datos y asignados a su docente.
Poscondiciones:	
Requisitos Especiales:	
Prototipo	



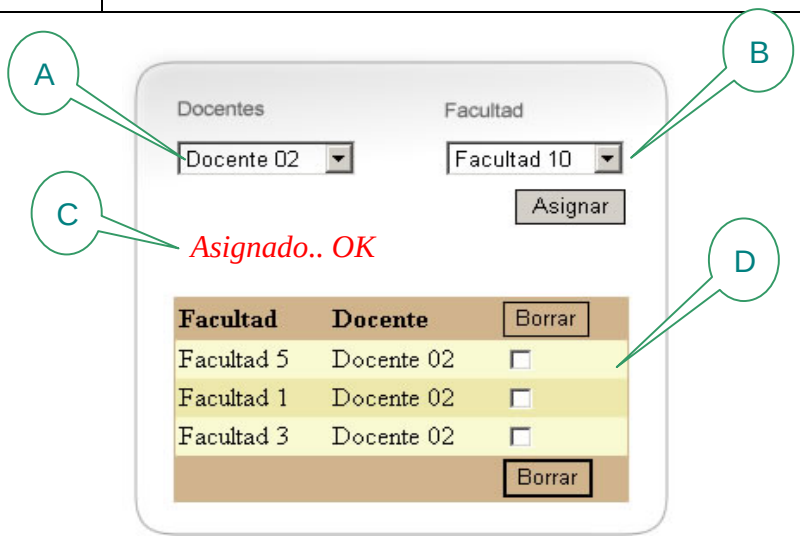
Pantalla 1

~~Caso de Uso:~~	**Administrar Docentes.**
Actor(es):	Jefe Área(Inicia)
Propósito:	Que el Jefe de Área realice cambios en los docentes dentro del sistema.
Resumen:	El caso de uso se inicia cuando el Jefe de Área desea realizar un cambio o una asignación nueva de un docente para esto debe escoger el docente (***Pantalla 1,A)*** Luego entrar el usuario del Jefe de Área que le corresponde (***Pantalla 1, B)*** el sistema coge estos datos e inserta o modifica el docente e informa resultados (***Pantalla 1, C***). Se muestra además: -Una tabla con los Docentes que están en el sistema con sus Jefes de Área. (***Pantalla 1, D***)
Referencias:	R6
Precondiciones:	
Poscondiciones:	
Requisitos Especiales:	

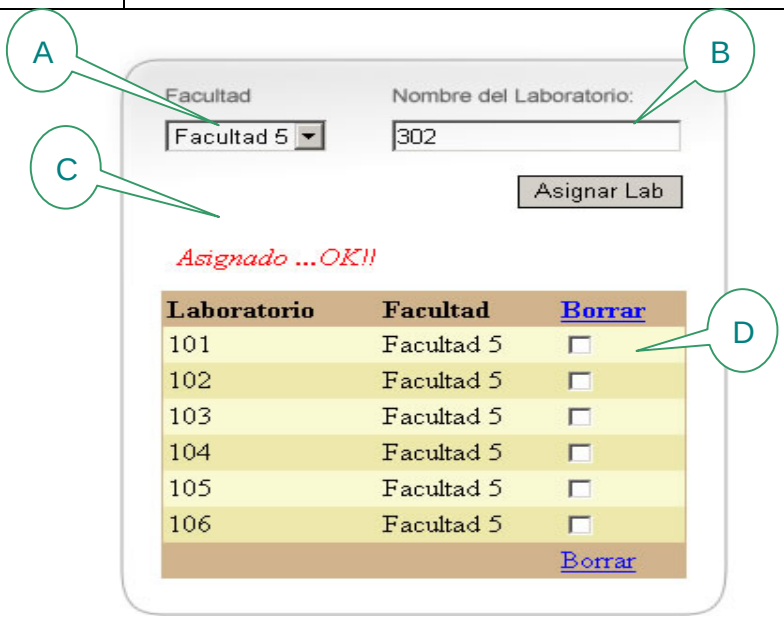
Prototipo



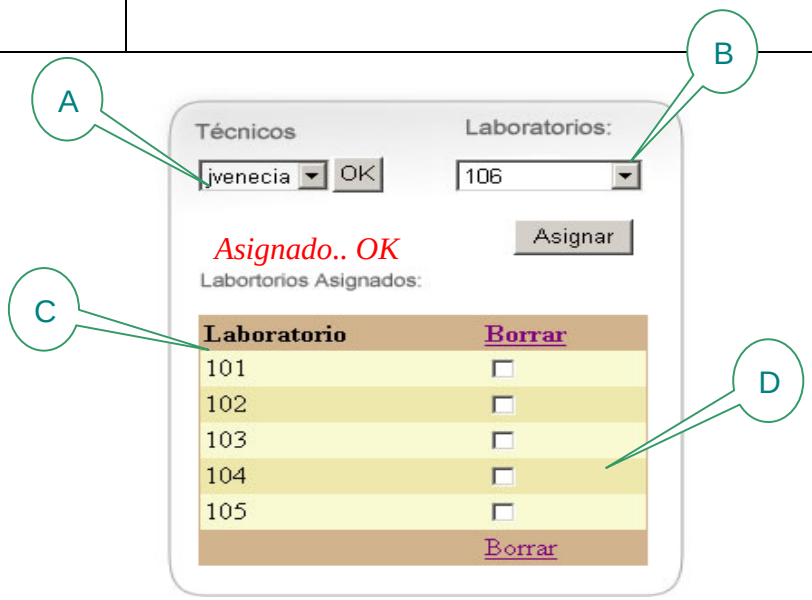
Pantalla 1

~~Caso de Uso:~~	**Gestionar distribución de Facultades**
Actor(es):	Jefe Área(Inicia)
Propósito:	Que el Jefe de Área realice cambios en la asignación de las facultades dentro de docentes.
Resumen:	El caso de uso se inicia cuando el Jefe de Área desea realizar un cambio o una asignación nueva de una facultad dentro de un docente para esto debe escoger el docente (***Pantalla 1,A)*** y la facultad (***Pantalla 1, B***) el sistema coge estos datos e inserta o modifica el docente donde radicará la facultad e informa resultados (***Pantalla 1, C***). Se muestra además: -Una tabla con las facultades que están en el docente seleccionado dentro del sistema (***Pantalla 1, D***)
Referencias:	R7
Precondiciones:	Debe haber Docentes en la base de datos.
Poscondiciones:	
Requisitos Especiales:	
Prototipo	

Pantalla 1

~~Caso de Uso:~~	**Gestionar distribución de laboratorios por facultad**
Actor(es):	Jefe Área(Inicia)
Propósito:	Que el Jefe de Área realice cambios en la asignación de los laboratorios dentro de facultades.
Resumen:	El caso de uso se inicia cuando el Jefe de Área desea realizar un cambio o una asignación nueva de un laboratorio dentro de una facultad para esto debe escoger la facultad (***Pantalla 1,A)*** y entrar un nombre para el laboratorio (***Pantalla 1, B***) el sistema coge estos datos e inserta la facultad donde radicará el laboratorio e informa resultados (***Pantalla 1, C***). Se muestra además: -Una tabla con los laboratorios que están asignados a la facultad seleccionada dentro del sistema (***Pantalla 1, D***)
Referencias:	R8
Precondiciones:	Debe haber Docentes en la base de datos y al menos una facultad asignada a este.
Poscondiciones:	
Requisitos Especiales:	
Prototipo	

Pantalla 1

~~Caso de Uso:~~	**Gestionar Distribución de Técnicos x Laboratorio**
Actor(es):	Jefe Área(Inicia)
Propósito:	Que el Jefe de Área realice cambios en la asignación de los técnicos por laboratorio.
Resumen:	El caso de uso se inicia cuando el Jefe de Área desea realizar un cambio o una asignación nueva de un técnico a un laboratorio para esto debe escoger el técnico (***Pantalla 1,A)*** y seleccionar el laboratorio (***Pantalla 1, B***) el sistema coge estos datos e inserta el técnico y el laboratorio asignado e informa resultados (***Pantalla 1, C***). Se muestra además: -Una tabla con los laboratorios que están asignados a el técnico seleccionado dentro del sistema (***Pantalla 1, D***)
Referencias:	R9
Precondiciones:	Debe haber laboratorios asignados a una facultad al menos en el docente donde radica el técnico.
Poscondiciones:	
Requisitos Especiales:	
Prototipo	

Pantalla 1

~~Caso de Uso:~~	**Eliminar Recursos.**
Actor(es):	Jefe Área(Inicia)
Propósito:	Que el Jefe de Área elimine cualquier recurso del sistema de la manera más rápida y eficiente.
Resumen:	El caso de uso se inicia cuando el Jefe de Área desea eliminar un medio en específico, al entrar a esta página tiene varias opciones para eliminar distintos medios, ya sea un laboratorio, una facultad o un docente. Además tiene la posibilidad de organizar su búsqueda para agilizar de una manera más confortable la ubicación del medio que quiera eliminar.(***Pantalla 1***)
Referencias:	R10
Precondiciones:	
Poscondiciones:	
Requisitos Especiales:	

Prototipo

Pantalla 1

3.7 Conclusiones

En este capítulo se comenzó a desarrollar la propuesta de solución, teniendo en cuenta los procesos de negocio y los requisitos que debe tener el sistema, llegando a los casos de uso que consideramos suficientes para satisfacer los requisitos anteriormente expuestos. Debido a esto se puede empezar la construcción del sistema, siguiendo lo marcado por los casos de uso.

CAPITULO 4

CONSTRUCCION DE LA SOLUCION PROPUESTA

4.1 Introducción.

En este capítulo hemos modelado los artefactos que ayudan a manejar las complejidades que surgen en la construcción de aplicaciones Web.

Los componentes de la aplicación se tratan como clases, utilizando el lenguaje de modelación UML, presentan a través de diagramas de clases Web. También se presenta el modelo de datos que es la base para construir finalmente la base de datos que soportará el trabajo del sistema. Finalmente se tratan los principios del diseño de la aplicación.

4.2 Diagrama de clases.

Para un mejor entendimiento del diagrama se ha decidido separar este en cuatro paquetes y a su vez uno de ellos en subpaquetes para una mayor organización y entendimiento de los mismos.

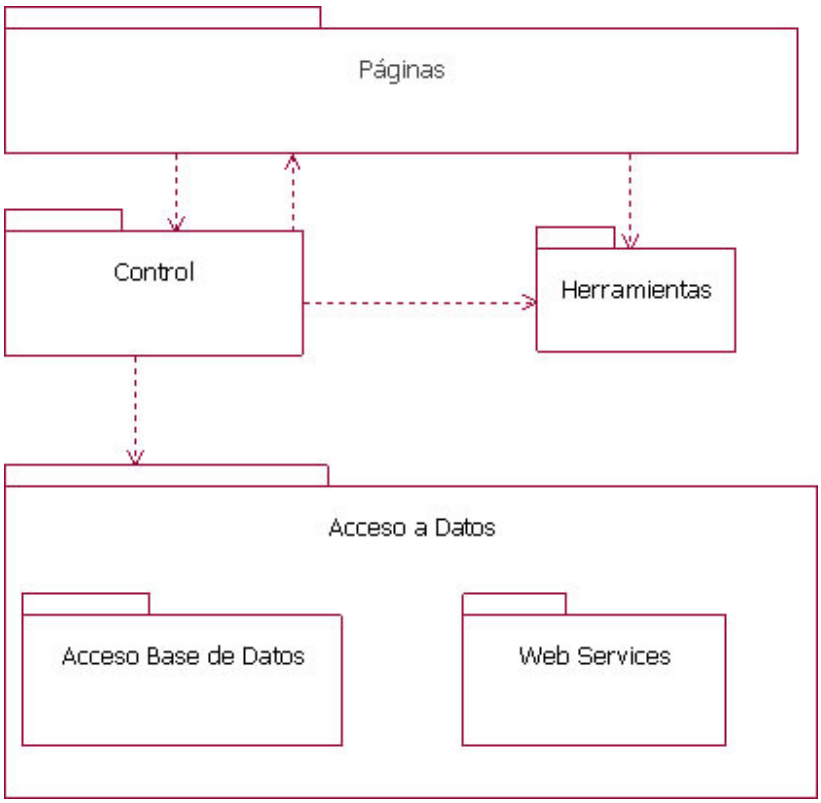


FIGURA 10. *DIAGRAMA DE CLASES DE DISEÑO*

El paquete *Acceso a Datos* contiene las clases que hacen posible la persistencia y la recuperación de objetos. Está dividido en dos subpaquetes:

- *Web Services* que abarca las clases para acceder a Servicios Web de la Intranet, necesarios para obtener diversas informaciones sobre personas y medios de la UCI.
- *Acceso Base de Datos,* contenido también dentro del paquete *Acceso a Datos*, contiene las clases encargadas de acceder a la base de datos para manipular la persistencia de las entidades.

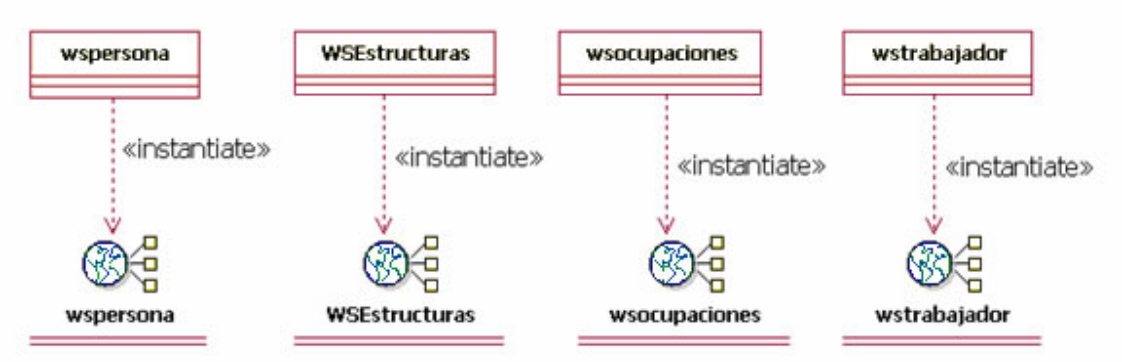
El paquete *Control* contiene como su nombre lo indica las clases controladoras del sistema que en si dan un soporte a la manera de realizar las operaciones específicas del sistema. El paquete *Páginas* contiene las páginas que gestionan las reservaciones, las asignaciones, los cambios etc.

El paquete *Herramientas* contiene las clases que usa el sistema para lograr funcionalidades necesarias que están fuera del sistema por ejemplo validar el usuario en el dominio.

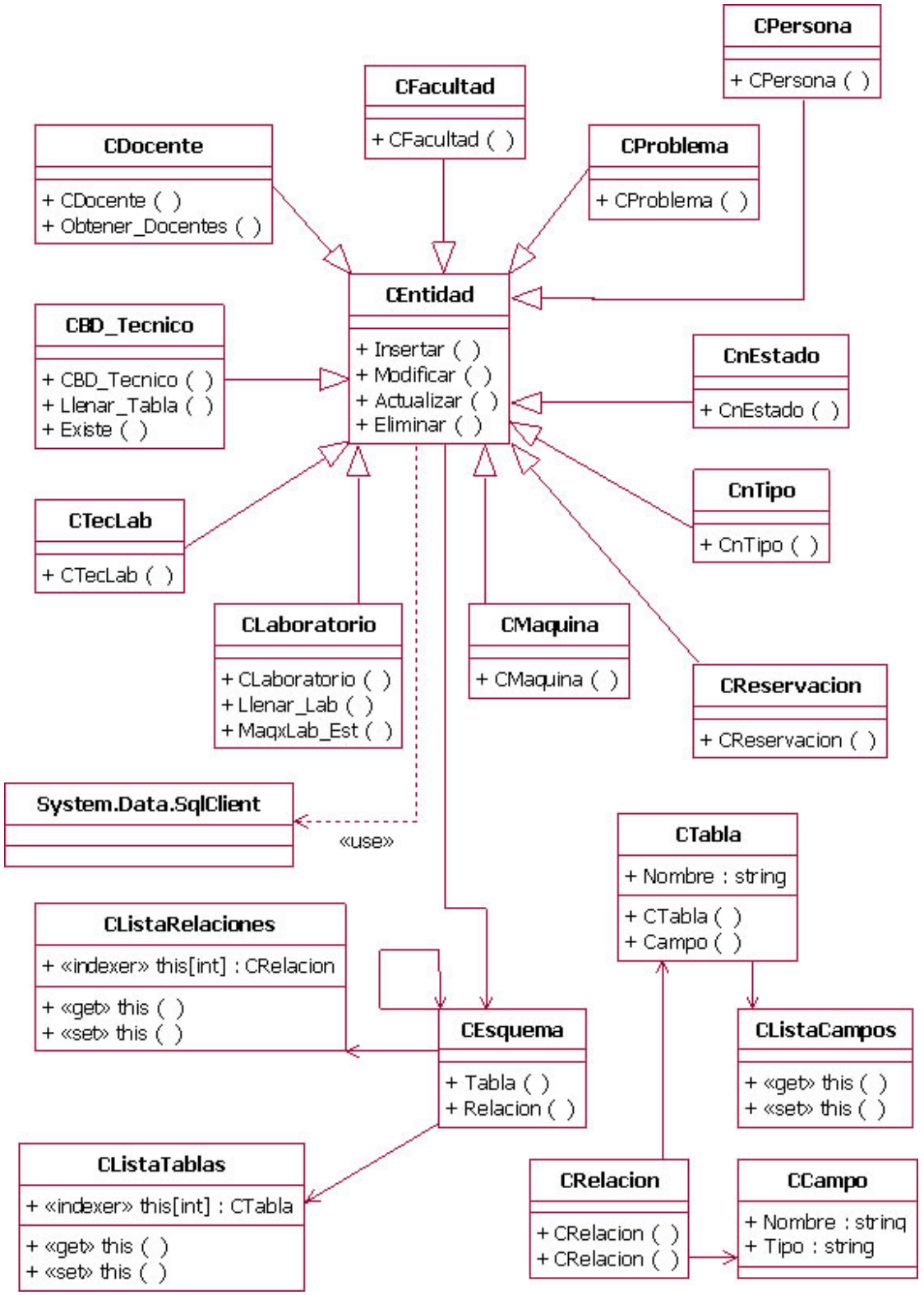
Esta división por paquetes sigue la arquitectura dividida por capas en que se ha basado el diseño de la aplicación y también a la funcionalidad de las clases, las cuales han sido agrupadas de esta forma para lograr mayor desacoplamiento, reutilización y legibilidad de los diagramas.

Paquete Acceso a Datos.

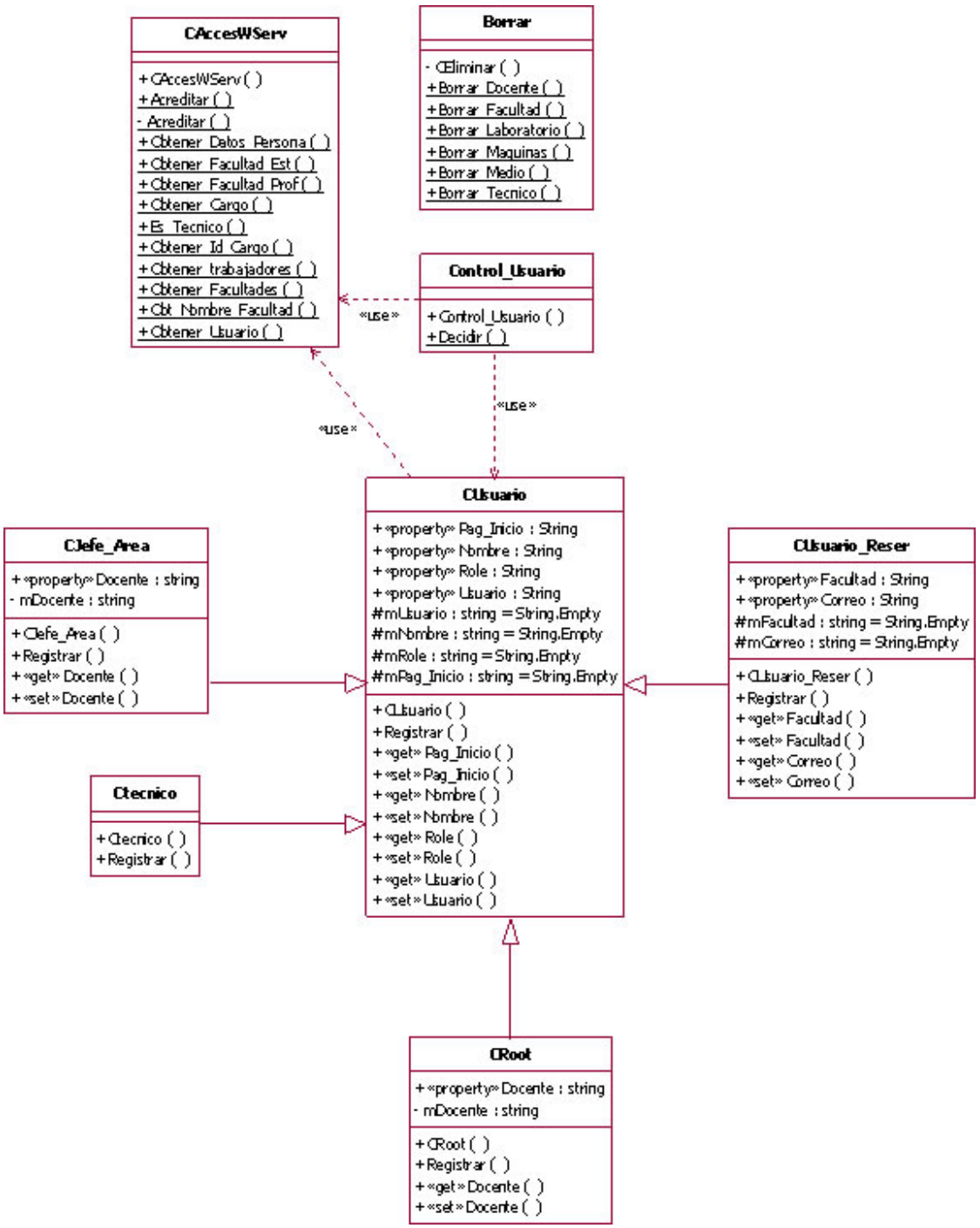
Subpaquete**: Web Services.**

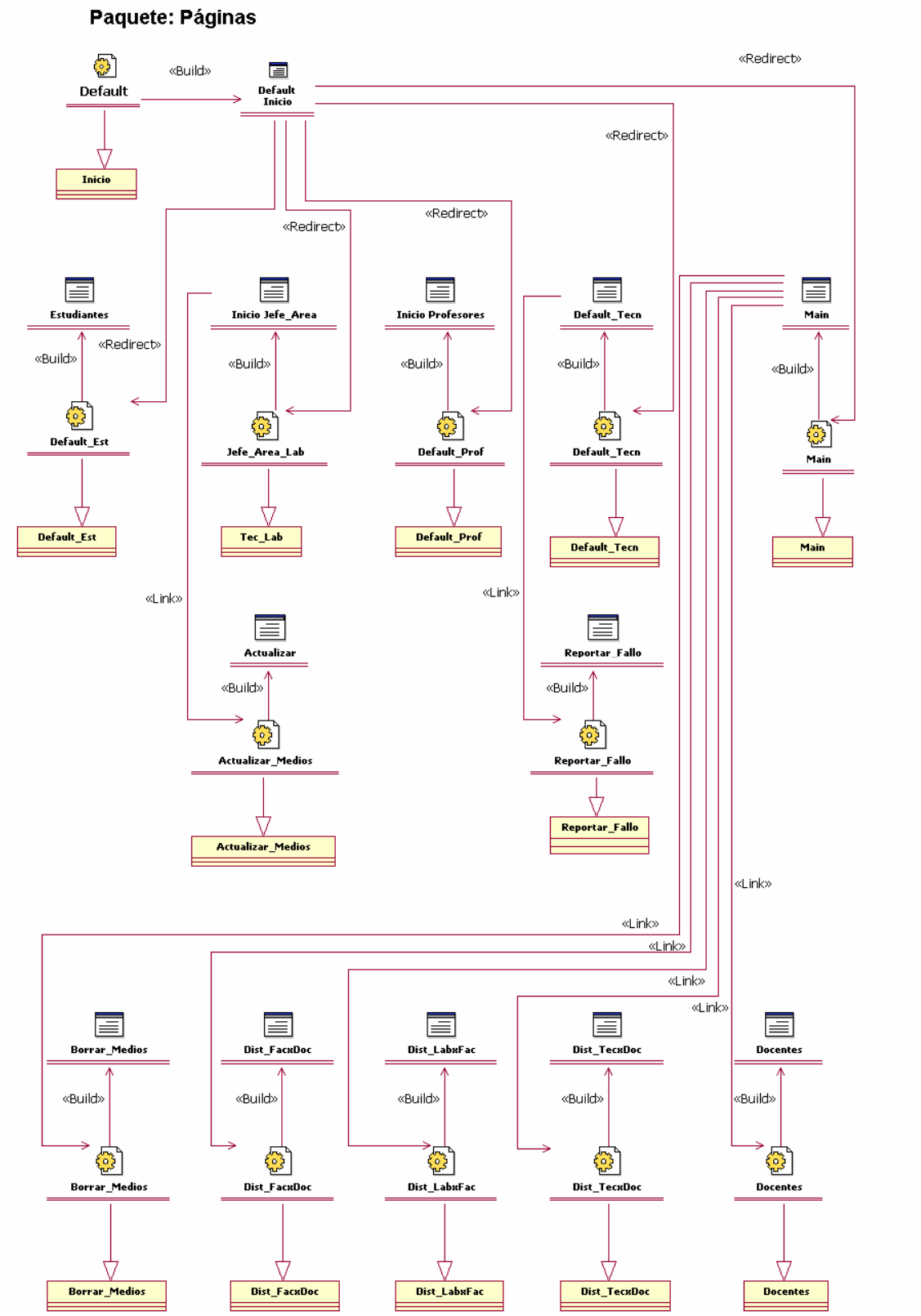


Subpaquete: **Acceso Base de Datos**

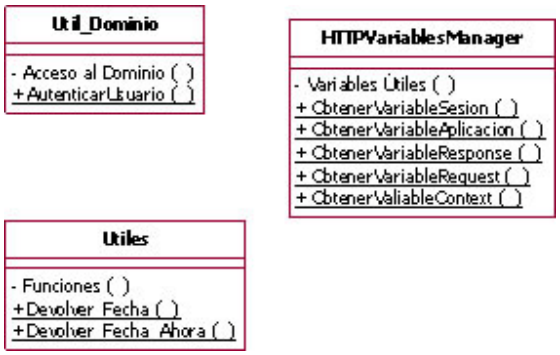


Paquete: **Control**




Paquete: Páginas
Default
«Build»
Default Inicio
«Redirect»
Inicio
«Redirect»
«Redirect»
«Redirect»
Estudiantes
Inicio Jefe_Area
Inicio Profesores
Default_Tecn
Main
«Build»
«Redirect»
«Build»
«Build»
«Build»
«Build»
Default_Est
Jefe_Area_Lab
Default_Prof
Default_Tecn
Main
Default_Est
Tec_Lab
Default_Prof
Default_Tecn
Main
«Link»
«Link»
Actualizar
Reportar_Fallo
«Build»
«Build»
Actualizar_Medios
Reportar_Fallo
Actualizar_Medios
Reportar_Fallo
«Link»
«Link»
«Link»
«Link»
«Link»
Borrar_Medios
Dist_FacxDoc
Dist_LabxFac
Dist_TecxDoc
Docentes
«Build»
«Build»
«Build»
«Build»
«Build»
Borrar_Medios
Dist_FacxDoc
Dist_LabxFac
Dist_TecxDoc
Docentes
Borrar_Medios
Dist_FacxDoc
Dist_LabxFac
Dist_TecxDoc
Docentes

Paquete **Herramientas.**



4.3 Diseño de la base de datos.

En el diseño de la base de datos del sistema, se han tomado como bases el *Diagrama de clases persistente* y el *Modelo de datos.*

4.3.1 Diagrama de clases persistente.

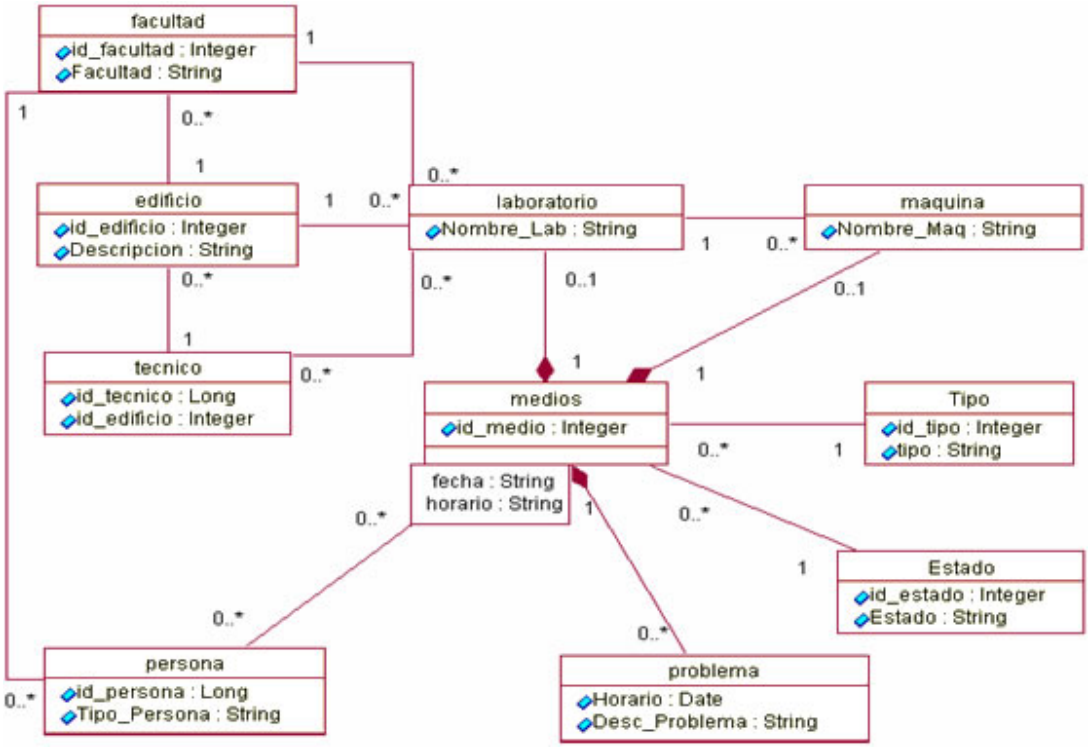


FIGURA 11. *DIAGRAMA DE CLASES PERSISTENTES.*

4.3.2 Modelo de datos.

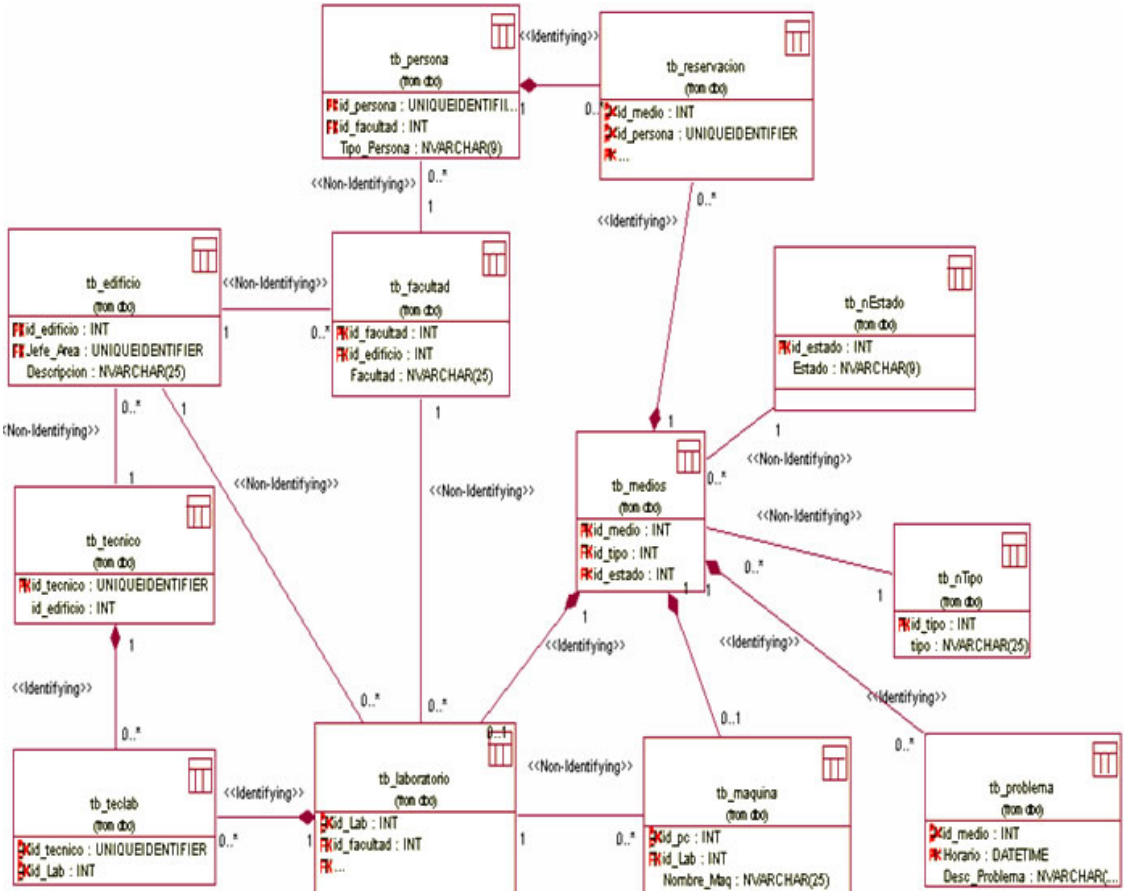


FIGURA 12. *MODELO DE DATOS.*

4.4 Principios de Diseño

El diseño ha sido realizado teniendo presente los usuarios finales, que serán: estudiantes, profesores y trabajadores que de una forma tengan relación con los laboratorios docentes, los cuales, no en todos los casos tienen conocimientos de informática, por lo tanto, se ha seleccionado una interfaz amigable, intuitiva y sencilla sin muchas dificultades para su uso. Además se ha mantenido un diseño consistente en todas las páginas, para lograr que el usuario se sienta cómodo y se acostumbre rápidamente al uso de la aplicación.

4.4.1 Estándares de la Interfaz de la aplicación

Todas las páginas Web están hechas con una estructura similar, contienen una imagen identificativa, además de opciones comunes a todos los usuarios tales como: el nombre del usuario activo. (*Figura 13,A*)

***FIGURA 13**. IMAGEN EN LA PARTE SUPERIOR DE TODAS LAS PÁGINAS*

Se utiliza el rojo para resaltar errores de campos requeridos, con formato incorrecto, o mensajes de operaciones no válidas. (Figura 14)

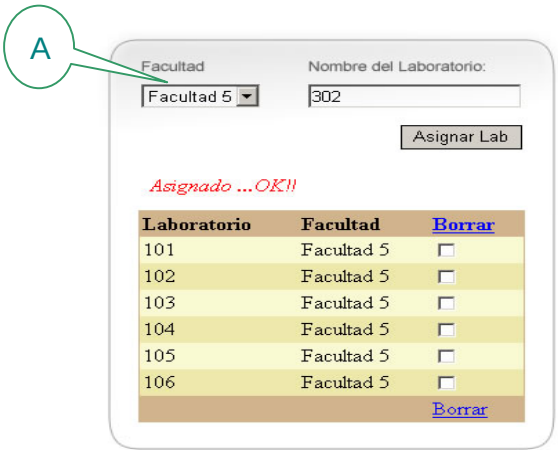


FIGURA 14. USO DEL COLOR ROJO PARA MENSAJES DE ERROR.

En general se realizan múltiples operaciones en cada página, de forma que el usuario no tenga que moverse tanto dentro de la aplicación, para completar una operación.

Se le solicita a los usuarios únicamente los datos completamente necesarios, y no lo que pueda ser calculado o inferido en el sistema, como fechas, por ejemplo. Para minimizar el margen de error.

4.4.2 Formato de Reportes

El sistema brinda varios reportes en forma de tabla. Los reportes permiten paginado, de forma que por cada búsqueda sean visibles solamente un número limitado de registros, permitiendo moverse adelante y hacia atrás. (Figura 15,A)

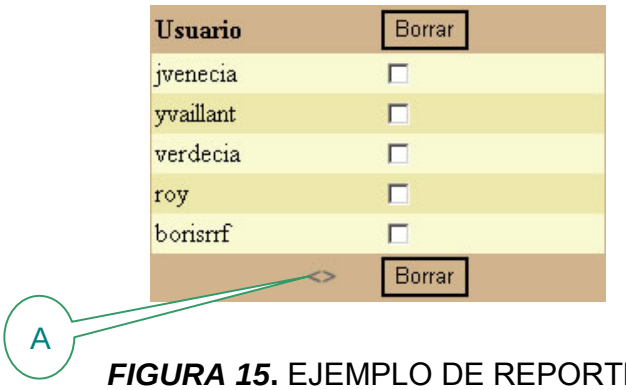


FIGURA 15. EJEMPLO DE REPORTE

4.4.4 Tratamiento de excepciones

Sólo se le brindan las opciones necesarias para prevenir errores por parte del usuario, a la hora de efectuar cualquier operación, se deshabilitan ciertos botones o se ocultan varios controles con una función dentro de un operación particular si el usuario no tiene que utilizarlos en ese momento.

Mediante una combinación de validación en el lado del cliente y en el lado del servidor, se garantiza que los datos suministrados por los usuarios, se almacenen íntegros, y no existan inconsistencias. Se verifican los campos obligatorios.

Todos los mensajes de error se muestran en un área determinada en color rojo para que resalten. Se manipulan las excepciones que podrían ocurrir en tiempo de corrida de la aplicación.

4.5 Estándares de codificación

Resulta de gran ventaja la utilización de un estándar para la escritura de código, entre dichas ventajas, tenemos las siguientes:

- Reducción de errores
- Escribir un código comprensible y fácil de leer
- Garantizar una buena comunicación entre los programadores del equipo
- Facilitar el mantenimiento del software.

En esta aplicación se ha utilizado el estándar de codificación “Camel Case” que principalmente tiene que ver con la capitalización de los caracteres. Así mismo, se ha seguido el estilo de codificación propuesto por Microsoft para programar con C# (*FIGURA 16).*

```
private void DataGrid1_ItemCommand(object source, DataGridCommandEventArgs e)
{
    if (e.Item.ItemType == ListItemType.Footer)
    {
        foreach (DataGridItem item in DataGrid1.Items )
        {
            if ((item.ItemType == ListItemType.Item) || (item.ItemType == ListItemType.AlternatingItem))
            {
                CheckBox chk = (CheckBox)item.FindControl("ChBEliminar");
                if (chk.Checked)
                {
                    string nombre = item.Cells[1].Text;
                    string id= e.Item.Cells[0].Text;

                }
            }
        }
    }
}
```

FIGURA 16. EJEMPLO DE CODIFICACIÓN UTILIZADA

Se ha tenido gran cuidado al nombrar clases, variables, y demás elementos, precediendo cada nombre con un prefijo para su fácil identificación, aquí tenemos los más frecuentes en este sistema:

- Campos de edición: **TextB**Nombre.
- Campos de selección: **ListB**Nombre o **CheB**Nombre
- Botones de acción: **BT**Nombre.
- Datagrids: **DG**Nombre
- Datasets: **ds**Nombre

En el diseño de la base de datos, las tablas se han nombrado igual que la entidad que almacenan precedidos por **tb**_. Los campos de estas tablas están nombrados igual que las propiedades de las entidades.

4.6 Modelo de despliegue

Un diagrama de despliegue contiene los nodos que conforman la topología de hardware sobre la que se ejecuta el sistema.

El sistema ha sido construido siguiendo la arquitectura de tres capas, o sea, la separación de los componentes en Lógica de Negocio, Acceso a Datos, Cliente.

El sistema consta de un servidor SQLServer 2000 que contiene la base de datos. El servidor Web reside en otro servidor y contiene al Internet Information Server, con la Plataforma .NET 1.0 o 1.1 instalada. Los clientes son los distintos usuarios desde sus terminales, accediendo al sistema mediante un navegador.

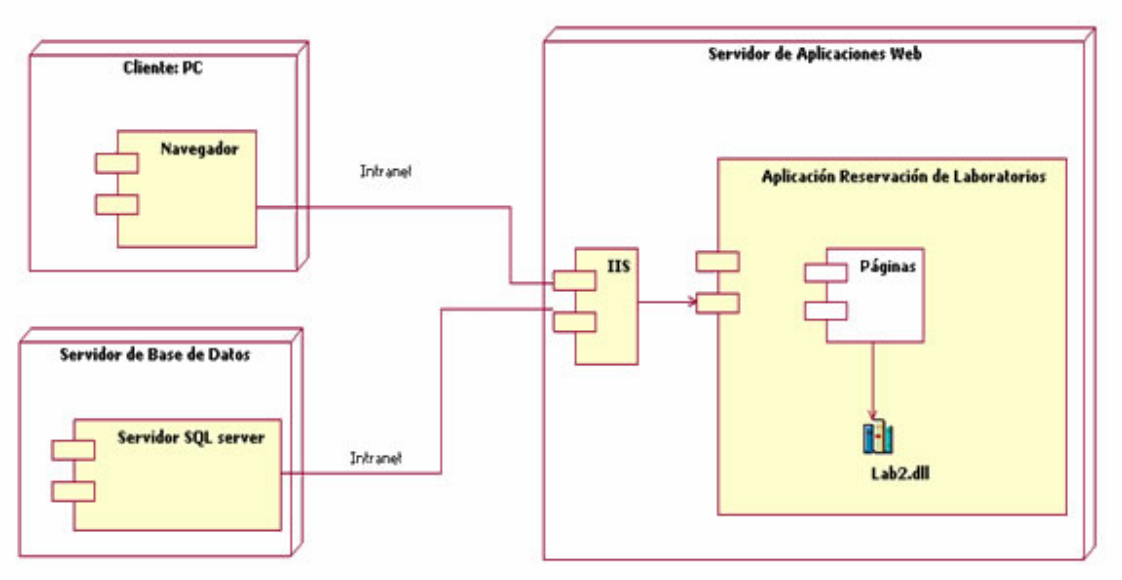


FIGURA 17. DIAGRAMA DE DESPLIEGUE

4.7 Modelo de Implementación

Paquete: **Acceso a Datos.**

Subpaquete: **Acceso Base de Datos.**

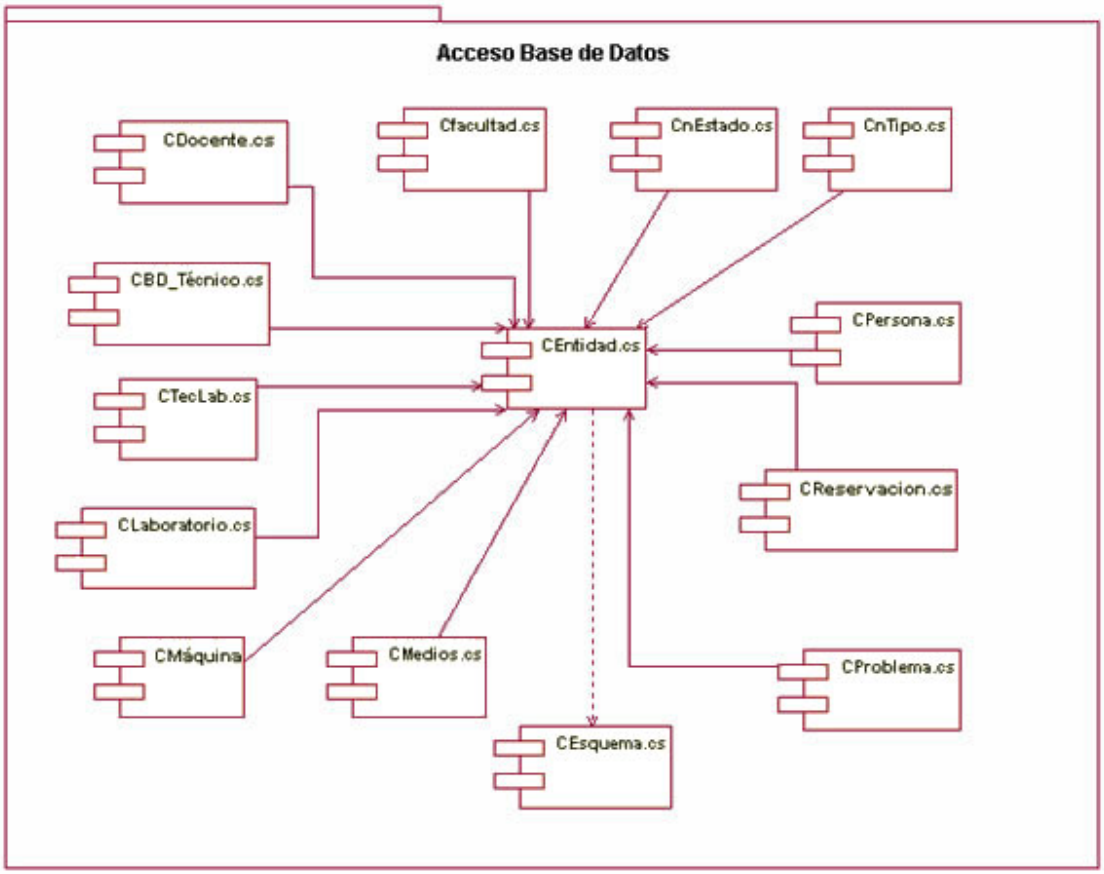


***FIGURA* 18**. DIAGRAMA DE COMPONENTES

Subpaquete: **Web Services.**

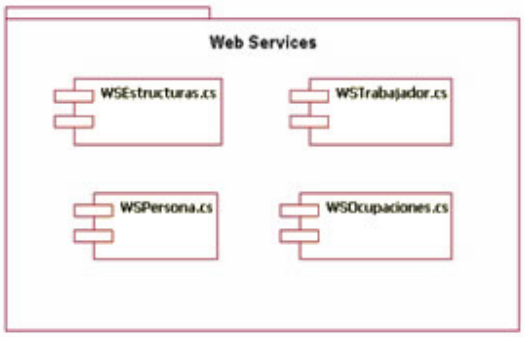


FIGURA 19. DIAGRAMA DE COMPONENTES (CONT)

Paquete: **Control.**

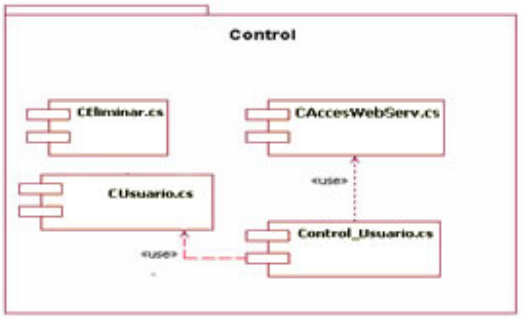


FIGURA 20. DIAGRAMA DE COMPONENTES (CONT)

- Paquete **Herramientas**

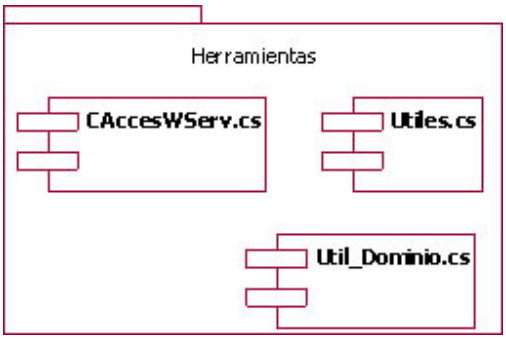


FIGURA 21. DIAGRAMA DE COMPONENTES (CONT)

Paquete: **Páginas.**

Código **HTML**

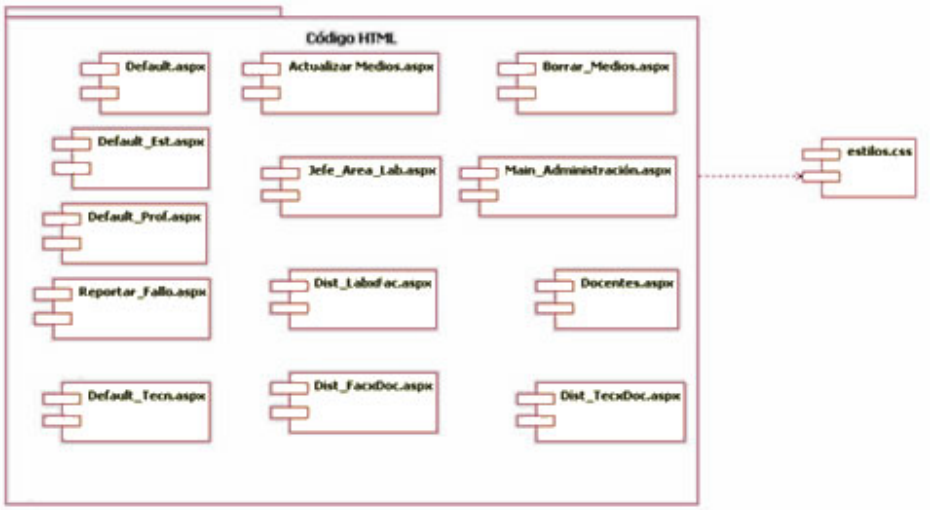


FIGURA 22. DIAGRAMA DE COMPONENTES (CONT)

Código: **CS**

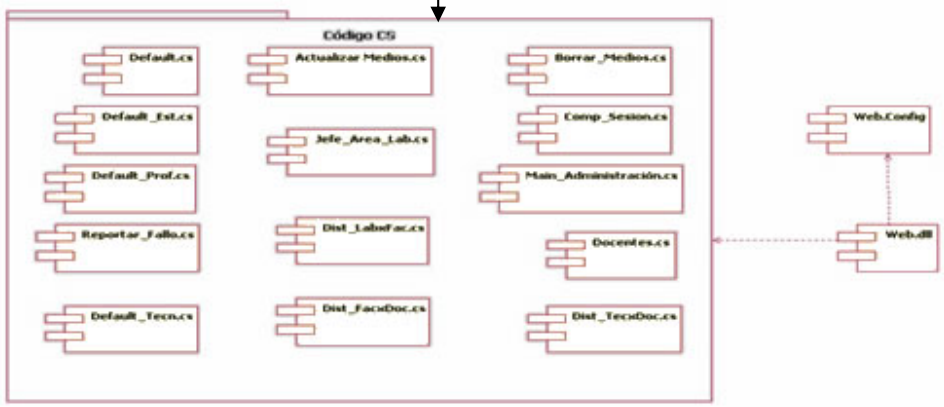


FIGURA 23. DIAGRAMA DE COMPONENTES (CONT)

4.7.1 Explicación de los componentes

Para lograr un mayor nivel de legibilidad en las explicaciones de cada componente, se mantendrá el mismo orden presentado por los anteriores paquetes. Para lograr una mayor limpieza y rapidez al encontrar las clases, se hizo necesario implementar cada una de en un fichero por separado, con el mismo nombre, por tanto, cada componente tiene el mismo nombre de la clase que contiene, y las interfaces que este expone, son los mismos métodos públicos de las clases. Los componentes que cumplen esta regla son los de los siguientes paquetes:

- Subpaquete **Acceso Base de Datos.**

En este paquete se encuentran los componentes que contienen las implementaciones de las clases que se ocupan de la persistencia en la base de datos.

- Subpaquete **WS**

En este paquete se encuentran los componentes que contienen las implementaciones de clases que se ocupan de la recuperación de datos accediendo a WebServices de la Intranet.

- Paquete **Código CS**

Este contiende el código fuente de las clases utilizadas por las páginas ASPX, los componentes se llaman igual que las páginas que dependen de ellos.

- Paquete **Herramientas**

Este contiende el código fuente de las clases con funciones útiles utilizadas por las demás, por ejemplo manejan las operaciones con las fechas, el dominio etc.

Existen otros componentes que no cumplen con la regla de nomenclatura antes mencionada, estos son:

Componente	Propósito	Contenido	Interfaces
Web.dll	Librería necesaria para el funcionamiento de las páginas sitio Web	Contiene las la implementación de las clases del paquete Presentación	Ver Diagrama de Clases del paquete Presentación.
Web.config	Archivo de configuración del sitio web.	Este componente es un XML con valores globales necesarios en diversas páginas.	-
estilos.css	Hoja de estilos del sitio	Contiene los estilos definidos para el sitio Web.	

Páginas Web

Componente	Propósito	Contenido
Default.aspx	Página de Inicio de la aplicación.	Contiene una Bienvenida y breve explicación del sitio, con algunos enlaces de interés.

Default_Est.aspx	Página para la reservación de tiempos de máquina por los estudiantes.	Ver CU Reservación de tiempos de máquina.
Default_Prof.aspx	Página para la reservación de laboratorios por los profesores.	Ver CU Reservación de Laboratorios.
Reportar_Fallo.aspx	Página para reportar fallos en los laboratorios o en un puesto de trabajo.	Ver CU Gestionar Reparaciones.
Default_Tecn.aspx	Página para Visualizar las reservaciones echas por estudiantes y profesores.	Ver CU Visualizar Reservaciones
Actualizar Medios.aspx	Página para reportar en buen estado un medio anteriormente fuera de servicio por parte del jefe de área.	Ver CU Gestionar Reparaciones.
Jefe_Area_Lab.aspx	Página donde el jefe de área actualiza la asignación de laboratorios a los	Ver CU Gestionar distribución de técnicos por laboratorio .

	técnicos de su área.	
Dist_LabxFac.aspx	Página para la distribución de laboratorios por facultad.	Ver CU Gestionar distribución de laboratorios por facultad.
Dist_FacxDoc.aspx	Página para la distribución de facultades por docente.	Ver CU Gestionar distribución de facultades.
Borrar_Medios.aspx	Página que brinda las operciones eliminación de medios del sistema.	Ver CU Eliminar Recursos.
Main_Administración.aspx	Página de inicio para el Jefe de Área principal.	Contiene enlaces a las distintas páginas relacionadas con Administración de los medios.
Docentes.aspx	Página para la actualización de los docentes.	Ver CU Administrar docentes.
Dist_TecxDoc.aspx	Página para la distribución de técnicos por docente.	Ver CU Gestionar distribución de técnicos.

4.8 Conclusiones

En este capitulo se ha llevado a cabo la descripción lo más detallada posible de las clases y otros elementos dentro de la implementación.

Se obtuvo el diagrama de clases del sistema, separado por paquetes. Se definieron, a partir del mismo, cuáles son las clases que serán persistentes, luego, partir de esto, construyó el modelo de datos. Se mostraron las reglas seguidas en el diseño de la interfaz.

Se explicó cómo está estructurada la aplicación físicamente, mediante los modelo de despliegue y de componentes.

CAPÍTULO 5

ESTUDIO DE FACTIBILIDAD.

5.1 Introducción

En el presente capitulo se hace un estudio de factibilidad, beneficios y costo del sistema propuesto. Con estos datos llegaremos a la conclusión de si es o no conveniente llevarlo a cabo teniendo en cuenta que si el riesgo del proyecto es alto la viabilidad de producir software de calidad se reduce.

5.2 Planificación

Nombre de la entrada externa	Cantidad de Ficheros	Cantidad de Elementos de datos	Clasificación (Simple, Media y compleja)
Reservar Tiempo de Máquina	2	7	Medio
Reservar Laboratorio	2	7	Medio
Insertar técnico	1	2	Simple
Eliminar técnico	2	4	Simple
Insertar docente	2	5	Medio
Eliminar docente	8	28	Complejo
Insertar facultad	1	3	Simple
Eliminar facultad	7	25	Complejo
Insertar laboratorio	3	10	Medio
Eliminar laboratorio	5	17	Complejo

Insertar máquina	2	6	Medio
Eliminar máquina	4	13	Complejo
Insertar persona	1	3	Simple
Reportar problema	2	6	Medio
Eliminar problema	2	6	Medio

Tabla1. Entradas externas

Nombre de la salida externa	Cantidad de Ficheros	Cantidad de Elementos de datos	Clasificación (Simple, Media y compleja)
Listado de laboratorios por docente.	1	2	Simple
Listado de laboratorios por facultad	1	2	Simple
Listado de máquinas por laboratorio.	2	2	Simple
Listado de facultades por docente.	1	2	Simple
Listado de laboratorios fuera de servicio	3	2	Simple
Listado de máquinas con problema por laboratorio.	3	2	Simple
Listado de técnicos por laboratorio	2	2	Simple
Listado de técnicos por docente.	2	2	Simple

Tabla 2. Salidas Externas.

Nombre de la petición	Cantidad de Ficheros	Cantidad de Elementos de datos	Clasificación (Simple, Media y compleja)
Visualizar Reservaciones por laboratorio.	3	4	Simple
Visualizar técnicos por docente.	2	4	Simple
Visualizar Jefe Área de cada docente.	2	4	Simple
Visualizar Facultades por docente	2	5	Simple

Tabla 3. Peticiones.

Nombre del fichero interno	Cantidad de records	Cantidad de Elementos de datos	Clasificación (Simple, Media y compleja)
tb_edificio	1	3	Simple
tb_facultad	1	3	Simple
tb_laboratorio	1	4	Simple
tb_maquina	1	3	Simple
tb_medios	1	3	Simple
tb_nEstado	1	2	Simple
tb_nTipo	1	2	Simple
tb_persona	1	3	Simple
tb_problema	1	3	Simple
tb_reservacion	1	4	Simple
tb_teclab	1	2	Simple
tb_tecnico	1	2	Simple

Tabla 4. Ficheros Internos.

Elementos	Simples		Medios		Complejos		
	No	X Peso	No	X Peso	No	X Peso	
Ficheros lógicos internos	12	7	0	10	0	15	84
Entradas externas	4	3	7	4	4	6	64
Salidas externas	8	4	0	5	0	7	32
Peticiones	4	3	0	4	0	6	12
Total							**192**

5.3 Costos

Características	Valor
Puntos de función desajustados	192
Lenguaje	C# (90 %) SQL (10%)
Instrucciones fuentes por puntos de función	(59) (39)
Instrucciones fuentes por lenguaje (miles de instrucciones)	(10,1952) (0.7488)
Instrucciones fuentes (miles de instrucciones)	**10,8**

Tabla 6. Cantidad de Instrucciones Fuentes

Factores	Valor	Justificación
PREC	6,2	Totalmente diferente.
FLEX	2.03	Debe haber buen cumplimiento de los requerimientos del sistema y de las especificaciones de interfaz externa.

TEAM	0	El equipo que va desarrollar el software es altamente cooperativo.
RESL	2.83	Se identifican muchos de los riesgos críticos.
PMAT	4.68	El proceso tiene nivel 2 de madurez.

Cálculo de:	**Valor**	**Justificación**
Esfuerzo	40,9	$\Pi EM_i \approx 1,09$ $E \approx 1,07$ $\sum SF_i = 15,74$ PREC = 6,2 FLEX = 2,03 RESL = 2,83 TEAM = 0 PMAT = 4.68
Tiempo de desarrollo	8 meses	PM ≈ 50 Hombres/Mes F ≈ 0.30
Cantidad de personas	5	Se cuenta con 5 personas para la realización del sistema.
Costo	8098 pesos	C =CHM * PM CHM = 3 * 225 = 675 PM ≈ 26.4
Salario medio	110	El salario mínimo es de 110 pesos.

RCPX	1,33	RELY = NOMINAL DOCU = alto CPLX = alto DATA = NOMINAL
RUSE	1,00	Existe concordancia entre la documentación y las necesidades del ciclo de vida.
PDIF	1,00	El .NET es una plataforma estable, no hay restricciones fuertes de memoria o tiempo nominal
PREX	1.00	Los desarrolladores tienen buena experiencia en la plataforma y herramientas de desarrollo. Lleva 1 año.
FCIL	0,87	se hace un uso notable de herramientas CASE para documentar implementar el sistema.
SCED	1,00	Las exigencias para el cumplimiento de cronograma son normales.
PERS	0,83	ACAP = ALTO pcap = ALTO pcon = alto

Tabla 8. Cálculos Finales de La Estimación de Costos y Esfuerzos.

5.4 Beneficios tangibles e intangibles

El Sistema de Laboratorios de la UCI no es un software con fines comerciales, aunque puede ampliarse para convertirlo en una solución general, capaz de aplicarse a cualquier empresa o institución que tenga características similares.

Su primer objetivo es automatizar el proceso de reservación dentro de los laboratorios docentes tanto de estudiantes como de los profesores, aunque además ,mantiene un control sobre la ubicación de los medios, la distribución de los técnicos a cargo y el control de las reparaciones y en general del estado de los medios dentro de los laboratorios.

Por tanto, los beneficios inmediatos son mayormente intangibles:

1. Ahorro de tiempo en la reservación de los laboratorios.
2. Control de las reservaciones y del estado de los medios.

5.5 Análisis de costos y beneficios

El desarrollo de este sistema no emite grandes gastos de recursos, ni de tiempo; la base de datos que contiene la información, puede ser montada en los servidores existentes en la Universidad. El sistema se ha diseñado pensando en una eventual migración al proyecto Mono por lo que un cambio de plataforma para la implantación del mismo es viable y factible, y no hay que incurrir en muchos cambios.

5.6 Conclusiones

En este capítulo se abarcó lo concerniente al estudio de factibilidad correspondiente al desarrollo del proyecto. Este permitió llegar a la conclusión que resultará factible implementar la aplicación, ya que aunque el existe cierto costo total, los beneficios sociales que se alcanzarán son considerables.

CONCLUCIONES

En este trabajo se propone la solución general al problema de la reservación de tiempos de máquina y laboratorios en la Universidad de Ciencias Informáticas. Se ha implementado un sistema capaz de llevar un registro de los medios existentes en los docentes, por ejemplo los laboratorios con fines docentes, y dentro de estos los puestos de trabajo, con aspectos como el estado, que nos permitirá saber si están en funcionamiento o no, si han sido reservados, cuantas veces han sido reservado etc. Permitiendo además el control del personal que trabaja en ellos, así como la distribución de laboratorios y facultades. El sistema se desarrolló siguiendo la metodología RUP, y se utilizaron representaciones UML para la modelación de todas las fases del proyecto.

El sistema ha sido elaborado cumpliendo los estándares de diseño y siempre realizando la implementación mediante las técnicas de programación orientada a objetos.

De esta manera damos por concluido el presente proyecto, dando por sentado el cumplimiento de los objetivos planteados. A continuación se incluyen recomendaciones que pensamos se deben tener en cuenta para el trabajo futuro.

RECOMENDACIONES

A pesar de haber cumplido los objetivos generales de este trabajo a medida que se ha ido desarrollando el proyecto han surgido ideas que podrían implementarse en un futuro para lograr un sistema más provechoso y efectivo, para lo cual se recomienda:

- Integrar el sistema con los sistemas que faltan como el de inventario, no se pudo integrar anteriormente porque este aún no está terminado.

- Continuar el desarrollo para que brinde un mejor servicio a la universidad.

GLOSARIO DE TÉRMINOS

- CSS: Siglas en Inglés de Cascading Style Sheets (Hojas de Estilo en Cascada).

- SOAP: Protocolo elaborado para facilitar la llamada remota a funciones a través de Internet, permitiendo que dos programas se comuniquen de una manera muy similar técnicamente a la invocación de páginas Web.

- Camel Case: Estilo de escritura de código, se basa en capitalizar las iniciales de las palabras que formen el nombre de un método o variables, menos la primera.

- URL: Un URL es una cadena de caracteres que identifica el tipo de documento, la computadora, el directorio y los subdirectorios en donde se encuentra el documento y su nombre.

- UML: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modelling Language) es el lenguaje de modelado de sistemas de software más difundido en la actualidad.

- Web Services: Aplicación que realiza un cometido y que puede formar parte de otros servicios para formar un servicio más completo. La comunicación hacia y desde el Web Service se realiza con XML.

REFERENCIAS BIBLIOGRÁFICAS

[1] Booch, G., Rumbaugh, J., Jacobson, I. *"El Lenguaje Unificado de Modelado"*. Addison-Wesley. 1999.

[2] Introducción a .NET. Monografías..

http://www.monografias.com/trabajos11/winnet/winnet.shtml (2004)

[3] Desarrollo basado en RUP bajo la herramienta Rational Rose http://lml.ls.fi.upm.es/mdp/si/ (2004)

[4] Introducción al SQL Server
http://www.microsoft.com/sql/ (20/6/2004)

[5] Introducción al .NET FRAMEWORK
http://www.clikear.com/ (10/2/2004)

[6] Introducción al ERWIN Studio
http://www.windowsitpro.com/Article/ArticleID/9155/9155.html?Ad=1
(20/4/2004)

BIBLIOGRAFÍA

1. Universidad .NET

   http://www.microsoft.com/spanish/msdn/comunidad/uni.net/ (4/4/2004)

2. Curso práctico de desarrollo de aplicaciones con Visual Studio .NET

   http://www.microsoft.com/spanish/msdn/comunidad/uni.net/ (4/4/2004)

3. Booch, G., Rumbaugh, J., Jacobson, I. "*The Unified Software Development Process"*. Addison-Wesley. 1999.
4. Larman, C. "*Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design".* Prentice-Hall, 1998.
5. Meyer, B. "*Construcción de software orientado a objetos"*. Prentice-Hall. 1998.
6. Jim Conallen. "*Building Web Applications with UML*." Addison-Wesley. 1999.
7. Curso de ASP.Net
   http://es.gotdotnet.com/quickstart/aspplus/ (12/2/2004)
8. UML Home Page. Object Management Group (OMG).
   http://www.uml.org/ (15/2/2004)
9. Manual de UML
   http://www.clikear.com/manuales/uml/index.asp (12/2/2004)
10. Manual de SQL
    http://www.clikear.com/manuales/sql/ (20/2/2004)
11. Manual de C #
    http://www.clikear.com/manuales/csharp/ (16/2/2004)
12. Manual de XML
    http://www.clikear.com/manuales/xml/ (14/2/2004)
13. Larman C . *"UML y Patrones"* Pearson .1999
14. Rational Rose. *"Using Rose Data Model" PDF. (2002)*
15. Berzal, F. *"Diseño de arquitecturas software",* PDF (2002)
16. Boggs, W; Boggs M. *"Matering UML With Rational Rose 2002" . Sybec 2002.*