Universidad de las Ciencias Informáticas

FACULTAD 2

"Sistema de Monitoreo y Control para los Exámenes Realizados en Laboratorios"

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor(es): Dolores Rodríguez Vega

Anabel Soria Guerra

Tutor(es): Lic. Alain Pérez Balart

Lic. Lester Rodríguez Vallejo

Ciudad de la Habana, Junio 2009

"Al mundo nuevo corresponde la Universidad nueva.

A nuevas ciencias que todo lo invaden, reforman y minan nuevas cátedras.

Es criminal el divorcio entre la educación que se recibe en una época, y la época.

En tiempos teológicos, universidad teológica.

En tiempos científicos, universidad científica"

José Martí

RESUMEN

La Universidad de las Ciencias Informáticas, como centro generador de conocimientos y principal impulsor del uso de las Tecnologías de la Información y las Comunicaciones (TICs), ha venido desarrollando una serie de exámenes digitales en los laboratorios del centro, ya sean prácticos o teóricos. La aplicación de este tipo de evaluación ha ido en incremento, siendo esta una de las vías más factibles para poner en práctica los conocimientos adquiridos por los estudiantes, así como valorar el estado académico de los mismos.

Actualmente resulta muy difícil controlar, por parte del profesor las actividades que realiza cada estudiante en su puesto de trabajo. El proceso de entrega y recogida de los exámenes, debido a su complejidad, requiere de más tiempo del establecido para la realización de los mismos.

El presente trabajo de diploma tiene como objetivo fundamental la implementación de un software encargado de brindar de forma cómoda y ágil las funcionalidades necesarias para el proceso de monitoreo y control de las evaluaciones realizadas en los laboratorios. El sistema está compuesto por dos aplicaciones: Esclavo y Maestro, que estarán en constante comunicación, con el objetivo de controlar las actividades que se realizan en cada una de las computadoras desde el momento en que inicia el examen.

Con el cumplimiento de estos objetivos se garantiza la agilización del tiempo de aplicación del examen, y se logra llevar un control automatizado del mismo.

Palabras claves:

Monitoreo, Control, Aplicación Maestro, Aplicación Esclavo

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Hoy en día la Universidad de las Ciencias Informáticas (UCI) se ha convertido en un centro generador de conocimientos, ciencia y tecnología, actuando como agente impulsor y receptor de las principales Tecnologías de la Información y las Comunicaciones (TICs) desde una perspectiva educativa y constructiva para el desarrollo que se está llevando a cabo. El uso y aplicación de las TICs constituye un proceso continuo de mejora del sistema docente educativo y su marcado avance ha llevado a la creación de nuevas técnicas que permiten mejorar la metodología de la enseñanza en la universidad, siendo ejemplo de estas, las continuas evaluaciones sistemáticas que el centro lleva a cabo en los laboratorios, fomentando así su uso y aprovechando al máximo sus beneficios.

Esta es una de las maneras en que es posible valorar el estado académico de los estudiantes, además de ser una de las vías más factibles para poner en práctica los conocimientos adquiridos por cada uno de ellos.

Actualmente el centro cuenta con una aplicación que facilita la realización de evaluaciones: Moodle; esta tiene como ventajas fundamentales la publicación de exámenes y la descarga de los mismos, sin embargo, cuando se realiza un examen en tiempo real, en ocasiones se hace necesario desconectar la red para evitar cualquier tipo de incidente. La Facultad 1 ha desarrollado una herramienta que realiza funcionalidades similares, posibilitando que varias facultades interactúen con el sitio a la vez, con la característica de que la información de una facultad se torna transparente a la otra. Por cuestiones de seguridad el sitio puede ser accedido por rangos de direcciones IP, lo cual se configura de forma independiente para cada facultad. Consta de una parte administrativa específica para cada facultad, permitiendo a los profesores descargar las pruebas realizadas, las cuales salen organizadas, por facultad, grupo y nombre del estudiante.

Analizando las características de las aplicaciones antes mencionadas se detectaron los siguientes inconvenientes:

- Resulta muy difícil controlar, por parte del profesor, las actividades que realiza cada estudiante en su puesto de trabajo. De la misma forma en que las tecnologías facilitan el trabajo de los profesores con la digitalización de la mayor parte de las pruebas, también se hace inevitable y casi indetectable el hecho de que, durante la realización de una evaluación, se coloquen en la computadora dispositivos externos o accedan a determinada información que puede estar o no ubicada en la red.

- El proceso de entrega y recogida de los exámenes a los estudiantes resulta complejo y demora demasiado tiempo, ya que se debe copiar en cada computadora los elementos necesarios para realizar la evaluación; una vez concluida, esta es recogida por cada uno de los puestos de trabajo, de forma similar a la vía antes expuesta.

Por todo lo antes expuesto ha surgido el siguiente **problema científico:** ¿Cómo entregar, controlar y recoger de forma eficiente los exámenes realizados en los laboratorios docentes?

El **objeto de estudio** a tener en cuenta para el desarrollo de este trabajo es el proceso de comunicación entre aplicaciones a través de la red, y el **campo de acción:** monitoreo de la actividad en una red de computadoras utilizando comunicación vía Socket TCP/IP.

Se plantea como **objetivo general** desarrollar una herramienta que permita monitorear y controlar los exámenes realizados en los laboratorios docentes; la misma estará compuesta por:

1. Desarrollar un software esclavo[1] que brinde las funcionalidades necesarias para el control de la computadora.

2. Desarrollar un software maestro[2] que permita dirigir el proceso de monitoreo y control.

[1] Software esclavo: Es el encargado de procesar los pedidos del software maestro y luego enviar la respuesta, permite chequear las actividades que se realizan en la PC y ejecutar comandos

[2] Software maestro: Es un software diseñado para usarlo en la red, desde una PC determinada, el cual estará en constante comunicación con el software esclavo, es el encargado de realizar solicitudes, funcionando a la vez como un controlador.

Para facilitar la investigación se proponen las siguientes **preguntas científicas**:

1. ¿Qué antecedentes se tienen del proceso de evaluación de las diferentes asignaturas que se imparten en la universidad?
2. ¿Cuál es la situación actual que enfrentan los profesores durante la aplicación de los exámenes en los laboratorios?
3. ¿Qué aspectos deben tenerse en cuenta para la elaboración del software esclavo y el software maestro que faciliten el cumplimiento de los objetivos propuestos?
4. ¿Cuál es la efectividad de la propuesta elaborada, una vez aplicada en la práctica?

Para un mejor entendimiento y logro de los objetivos se trazaron las siguientes **tareas científicas:**

1. Investigar acerca de los métodos que se llevan a cabo actualmente para el control de los exámenes en los laboratorios, tanto en Cuba como en el resto del mundo.
2. Investigar acerca de las tecnologías, lenguajes y herramientas a utilizar
3. Realizar el proceso de análisis y diseño del software.
4. Investigar acerca de las vías existentes para desarrollar la comunicación entre dos aplicaciones a través de la red.
5. Llevar a cabo la implementación de cada uno de los subsistemas y posteriormente del sistema en general.
6. Realizar las pruebas necesarias al software implementado.

Para una mejor organización, el presente trabajo se ha estructurado por capítulos que abordan diferentes temáticas y aspectos de la investigación, así como cada uno de los pasos que se tuvieron en cuenta para el desarrollo del software:

Capítulo 1. Fundamentación Teórica: Se dará una panorámica acerca de las herramientas que antecedieron esta aplicación, especificando las funcionalidades que presentan en común. Se aborda

acerca de la tecnología en la que se apoya, así como la metodología a utilizar en cada una de las etapas de desarrollo del software.

Capítulo 2. Características del sistema: Se realiza una descripción general de la propuesta de sistema teniendo como objetivo fundamental comprender y entender los problemas actuales por lo que es necesaria su realización, así como describir los procesos que serán objeto de automatización para su posterior representación mediante cada uno de los artefactos generados por la metodología seleccionada.

Capítulo 3. Exploración y Planificación: Se realiza la descripción de los requerimientos del software mediante las historias de usuario que describirán en detalle las características del sistema definidas previamente. Se definirá además el tiempo total dedicado a cada iteración del proyecto durante la etapa de implementación.

Capítulo 4. Implementación y prueba: Se definen las tareas que darán cumplimiento a cada Historia de Usuario. Se describen las clases del diseño utilizando las Tarjetas CRC (Clases, Responsabilidades y Colaboración). Por cada historia de usuario se realizan las pruebas necesarias a través de las pruebas de aceptación, artefacto generado por la metodología

Capítulo 5. Estudio de Factibilidad: Se realiza un estudio de factibilidad del sistema propuesto, obteniéndose así una idea aproximada del costo y el esfuerzo empleados en el desarrollo del mismo.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

En este capítulo se presentan los fundamentos teóricos que apoyan el tema de desarrollo, referidos a los procesos que se llevan a cabo para el control de exámenes realizados en laboratorios docentes, se profundiza sobre el proceso de comunicación entre aplicaciones a través de la red, se realiza un análisis de los software existentes en la universidad profundizando además en las tendencias existentes actualmente respecto al desarrollo de software; se describen las tecnologías y metodologías utilizadas exponiendo las razones por las que fueron seleccionadas, así como las herramientas de desarrollo empleadas para el cumplimiento de los objetivos trazados.

1.2. Antecedentes del tema, tendencias y tecnologías actuales.

1.2.1. Sistemas automatizados existentes

En la universidad se han desarrollado aplicaciones encargadas de gestionar los exámenes que se realizan en los laboratorios docentes. Un ejemplo representativo de estas lo constituye la plataforma Moodle, la misma se caracteriza por ser un sitio web, implementado en el lenguaje de programación PHP, que ofrece funcionalidades tales como la gestión de cursos. Sin embargo presenta varias desventajas que provocan que el trabajo con la misma resulte tedioso, ejemplo de esto lo constituye el hecho de que al realizar los exámenes prácticos en los laboratorios, una vez finalizado el tiempo establecido para el desarrollo de los mismos, el estudiante debe subir la prueba para la plataforma, por lo que es necesario establecer la red, esto traería como consecuencia que el mismo pueda acceder a cualquier información que se encuentre disponible dentro de la misma. En caso que el servidor presente algún fallo en el momento que se esté efectuando una evaluación, puede provocar pérdida de información y datos necesarios para el trabajo que esté realizando el estudiante en ese instante.

La Facultad 1 cuenta con una herramienta implementada, al igual que en el caso anterior, en el lenguaje de programación PHP, con MySQL como Gestor de Base de Datos, permite que varias facultades interactúen con el sitio a la vez; a modo de seguridad permite ser accesible por rangos de direcciones IP, lo cual es configurable de forma independiente para cada facultad. El proceso de recogida del examen, se realiza de la siguiente manera: después de estar desconectada la red, el profesor debe ir por cada puesto de trabajo, es conectada y da paso a que el estudiante suba hacia el sitio la prueba, de forma similar al ejemplo expuesto anteriormente, esto provoca que se requiera de más tiempo del establecido para el examen.

En general, estas aplicaciones contribuyen a la gestión de exámenes, pero no han sido diseñadas para controlar las actividades que realiza cada estudiante desde su computadora en el momento activo de la prueba.

En el país actualmente no existe un software que solucione el problema actual. En el mundo existen herramientas encargadas de monitorear recursos, sitios Web, procesos, pero son productos propietarios que requieren el pago de una licencia para su uso, y ninguna de ellas satisface las necesidades existentes.

Luego de un profundo análisis de la situación actual se hace necesario el desarrollo de un software capaz de dar solución al problema en cuestión.

1.2.2. Tendencias y tecnologías actuales

Cuba tiene como una de sus tendencias principales la utilización del software libre, debido a la situación existente y a muchas de las restricciones que existen actualmente en el campo de la informática, en la que se ve mayormente afectada, por lo que se hace necesaria la migración desde los sistemas de licencia comercial a aquellos cuyo uso por parte de cualquiera que esté interesado en hacerlo, está exento de pago o paga muy poco por el derecho de usarlo.

Actualmente la UCI se propone como objetivo fundamental la migración al software libre, sin embargo, por diversas razones esto no se ha logrado en su totalidad. Teniendo en cuenta estas tendencias la herramienta en cuestión se desarrolla siguiendo estas premisas, brindando la posibilidad de su utilización en la universidad.

1.3. Comunicación entre aplicaciones.

Una de las características más importantes del sistema a implementar es que ambas aplicaciones serán capaces de conectarse entre sí aun encontrándose en equipos totalmente diferentes, además de que el envío de datos entre ellas es fundamental para el funcionamiento del software en general. Por este motivo fue necesario realizar un estudio de algunas formas de comunicación entre aplicaciones a través de la red, llegando a la conclusión de que el mecanismo más apropiado para dar solución a los objetivos propuestos es utilizando la Comunicación vía Sockets TCP/IP.

1.3.1. Comunicación a través de Socket TCP/IP.

Un socket (enchufe), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red, se define, por tanto, como el punto final en una conexión (1).

Este mecanismo surge a principios de los 80 con el sistema UNIX de Berkeley, para proporcionar un medio de comunicación entre procesos (2) y presentan la misma funcionalidad que tiene la comunicación por correo o por teléfono (de un buzón se extraen mensajes completos, mientras que el teléfono permite el envío de flujos de información que no tienen una estructura claramente definida), es decir permiten que un proceso hable (emita o reciba información) con otro incluso estando estos en distintas máquinas. Esta característica de ínter conectividad hace que el concepto de socket sea de gran utilidad (3).

Cuando un cliente conecta con el servidor se crea un nuevo socket, de esta forma, el servidor puede seguir esperando conexiones en el socket principal y comunicarse con el cliente conectado, de igual manera se establece un socket en el cliente en un puerto local (3).

Una aplicación servidor normalmente escucha por un puerto específico esperando una petición de conexión de un cliente, una vez que se recibe, el cliente y el servidor se conectan de forma que les sea posible comunicarse entre ambos. Durante este proceso, el cliente es asignado a un número de puerto, mediante el cual envía peticiones al servidor y recibe de este las respuestas correspondientes. Similarmente, el servidor obtiene un nuevo número de puerto local que le servirá para poder continuar escuchando cada petición de conexión del puerto original. De igual forma une un socket a este puerto local (4).

Generalmente la comunicación con sockets se realiza mediante un protocolo de la familia TCP/IP (Protocolo de control de transmisión/Protocolo de Internet). Los dos más utilizados son: TCP (Protocolo de Control de Transmisión) y UDP (Protocolo de Datagrama de Usuario), el primero es orientado a la conexión, permite a dos anfitriones establecer una conexión e intercambiar datos y garantiza la entrega de datos, es decir, que los mismos no se pierdan durante la transmisión y que los paquetes sean entregados en el mismo orden en el cual fueron enviados; el segundo ofrece ventajas como rapidez y facilidad de recibir datos de varios anfitriones usando un solo socket (5).

En la actualidad existen varios tipos de socket y cada uno por lo regular se asocia a un tipo de protocolo, por ejemplo:

SOCK_STREAM: está asociado al protocolo TCP. El cual brinda seguridad en la transmisión de datos, seguridad en la recepción, en la integridad y en la secuencia, entre otros.

SOCK_DGRAM: está asociado al protocolo UDP, e indica que los paquetes viajarán en tipo datagramas, el cual tiene una comunicación asíncrona (6).

Su principal ventaja radica en que son muy eficientes a la hora de enviar muchos mensajes y datos, tales como transmisión de ficheros, lo cual es fundamental para el envío y recogida de exámenes, funcionalidades de gran importancia en el sistema.

1.4. Lenguajes de programación

Los lenguajes de programación son herramientas que permiten crear programas y software. Disponen de formas adecuadas que posibilitan ser leídas y escritas por personas; a su vez resultan independientes del modelo de la computadora a utilizar (7). En la actualidad existen variedades de lenguajes, cada uno con características que los hacen diferentes de los demás. En la universidad son utilizados un gran número de ellos, debido a las necesidades y requisitos de las que disponga el software a desarrollar, un ejemplo de estos es C++, el cual es un lenguaje orientado a objetos derivado del C, que nació para añadirle cualidades y aspectos de los que carecía, entre sus principales características se encuentran portabilidad, brevedad y velocidad; y C Sharp (C#) que es un lenguaje de propósito general orientado a objetos creado por Microsoft para la plataforma .NET. Se caracteriza por la sencillez de uso, modernidad, orientado a componentes y seguridad de tipos.

1.4.1. Java

Teniendo en cuenta el estudio realizado sobre estos y otros lenguajes de programación orientada a objetos, se llega a la conclusión de que el lenguaje a utilizar para la implementación del sistema es Java, debido fundamentalmente a que es un lenguaje multiplataforma que, a medida que pasan los años, toma mayor importancia en el mundo de la informática, además, no requiere de mucho tiempo para producir. Los programas desarrollados en Java presentan diversas ventajas frente a otros implementados en lenguajes como C/C++. Permite fácilmente el desarrollo tanto de arquitecturas cliente - servidor como de aplicaciones distribuidas, consistentes en crear sistemas capaces de conectarse a otros ordenadores y ejecutar tareas en los mismos simultáneamente, repartiendo por lo tanto el trabajo. Aunque también otros lenguajes de programación permiten crear aplicaciones de este tipo, Java incorpora en su propio API (Interfaz de Programación de Aplicación) estas funcionalidades.

Java es un lenguaje de programación lo bastante potente para desarrollar aplicaciones en cualquier ámbito, o sea, prácticamente todo aquello que se puede hacer en cualquier lenguaje se puede hacer también en Java y muchas veces con grandes ventajas (8).

En la actualidad es un lenguaje muy extendido enfocado fundamentalmente a cubrir las necesidades tecnológicas más punteras, por lo que cada vez cobra importancia tanto en el ámbito de Internet como en la informática en general. Es además un lenguaje de propósito general con el que se puede escribir desde un Applet para una página Web, hasta una aplicación financiera en modo texto sin ninguna conexión a Internet (9). Todo esto, unido a la portabilidad, potencia, seguridad y estabilidad que presenta lo ha hecho convertirse en un lenguaje utilizado por millones de programadores de todo el mundo.

Es un lenguaje dinámico, la máquina virtual de java, enlaza los programas java en tiempo de ejecución, eliminando la necesidad de enlazarlos con las librerías en tiempo de compilación (10).

Es un lenguaje preparado para la red, puesto que fue creado desde el principio como un lenguaje pensado para esta. La máquina virtual de Java evita que los programas transferidos al ordenador lo destruyan, facilita la rápida transferencia de dichos programas y los ejecuta de modo que no dependan del sistema operativo subyacente (10).

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas (11).

Hoy en día ya se ven limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario, mientras otro presenta una animación en pantalla y otro realiza cálculos (11).

1.4.1.1. Máquina Virtual de Java

Un programa Java no tiene contacto con el ordenador en el que se ejecuta, sino que se relaciona con la máquina virtual. Esta configuración tiene una serie de implicaciones de gran importancia, técnicamente esta decide lo que los programas Java pueden hacer y lo que no (12).

Esta máquina virtual actúa como un cortafuego entre el ordenador host (anfitrión) y el programa Java. Este último nunca accede a los mecanismos de entrada y salida del ordenador, al sistema de archivos, ni a la memoria, simplemente solicita a la máquina virtual que lo haga (12).

1.4.2. JNI (Interfaz Nativa de Java)

La plataforma Java es relativamente nueva, lo que significa que algunas veces se necesita integrar programas escritos en Java con servicios, programas o APIs existentes desarrollados en lenguajes distintos. La plataforma Java proporciona la Interfaz Nativa de Java (JNI) para contribuir con dicha integración (13).

1.4.2.1. Características

El JNI define una convención de nombres y llamadas para que la Máquina Virtual Java pueda localizar e invocar a los métodos nativos, está construido dentro de la máquina virtual Java, por lo que puede llamar a sistemas locales para realizar entrada / salida, gráficos, trabajos de red y operaciones de hilos de ejecución sobre el host del sistema operativo (13).

Posibilita modificar programas existentes escritos en algún otro lenguaje, permitiéndoles ser accesibles desde aplicaciones Java. Muchas de las clases de la API estándar de Java dependen del JNI para proporcionar funcionalidad al desarrollador y al usuario, por ejemplo las funcionalidades de sonido o lectura/escritura de ficheros. El desarrollador debe asegurarse que la API estándar de Java no proporciona una determinada funcionalidad antes de recurrir al JNI, ya que la primera ofrece una implementación segura e independiente de la plataforma (14).

Posibilita realizar llamadas a código nativo desarrollado en lenguajes como C y C++, permitiendo la utilización de los objetos Java de la misma forma en que el propio código lo hace (14).

1.4.3. C

El lenguaje de programación C, es un lenguaje conocido como de alto nivel. Una de sus características fundamentales es que es un lenguaje estructurado, lo que permite generar código claro y sencillo, ya que está basado en la modularidad (15).

La popularidad, eficacia y potencia de C se ha producido porque este lenguaje no está prácticamente asociado a ningún sistema operativo, ni a ninguna máquina en especial. Esta es la razón fundamental por la que C es conocido como el lenguaje de programación de sistemas por excelencia (16).

1.4.3.1. Características

Hay numerosas características que diferencian al lenguaje C de otros, y lo hacen eficiente, potente, eficaz, rápido e indispensable para todos los programas. Algunas son:

1. Una nueva sintaxis para declarar funciones: Una declaración de función puede añadir una descripción de los argumentos de la función. Esta información adicional sirve para que los compiladores detecten más fácilmente lo errores causados por argumentos que no coinciden.
2. Asignación de estructuras (registros) y enumeraciones.
3. Preprocesador más sofisticado.

4. Una nueva definición de la biblioteca que acompaña a C. Entre otras funciones se incluyen: acceso al sistema operativo (por ejemplo, lectura / escritura de archivos), entrada y salida con formato, asignación dinámica de memoria, manejo de cadenas de caracteres.

5. Una colección de cabeceras estándar que proporciona acceso uniforme a las declaraciones de funciones y tipos de datos (16).

6. Permite la creación de DLLs (Librería de Enlace Dinámico), ofreciendo la posibilidad de implementar métodos nativos en las mismas, siendo de gran utilidad ya que brinda a otros lenguajes funcionalidades de las que carecen.

1.4.4. DLL (Librería de Enlace Dinámico)

Una DLL es un archivo con código ejecutable que se carga bajo demanda del programa por parte del sistema operativo. Contiene funciones que se pueden llamar desde aplicaciones u otras DLLs y son utilizadas para poder reciclar el código y aislar las diferentes tareas (17)

1.4.4.1. Características

1. **Reducen el tamaño de los archivos ejecutables:** Gran parte del código puede estar almacenado en bibliotecas y no en el propio ejecutable lo que redunda en una mejor modularización.

2. **Pueden estar compartidas entre varias aplicaciones:** Si el código es suficientemente genérico, puede resultar de utilidad para múltiples aplicaciones

3. **Facilitan la gestión y aprovechamiento de la memoria del sistema**: La carga dinámica permite al sistema operativo aplicar algoritmos que mejoren el rendimiento del sistema cuando se carguen estas bibliotecas. Además, al estar compartidas, basta con mantener una copia en memoria para todos los programas que la utilicen.

4. **Brindan mayor flexibilidad frente a cambios**: Es posible mejorar el rendimiento o solucionar pequeños errores distribuyendo únicamente una nueva versión de la biblioteca dinámica. Nuevamente, está corrección o mejora será aprovechada por todas las aplicaciones que compartan la biblioteca siempre será verdad (18).

1.5. Entornos de desarrollo de Java

En la actualidad existen distintos programas que permiten desarrollar el código Java, dada a la distribución gratuita del *Java(tm) Development Kit* (JDK), el cual lleva implícito un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en Java. Incorpora además la posibilidad de ejecutar parcialmente el programa, deteniendo la ejecución en el punto deseado y estudiando en cada momento el valor de cada una de las variables (con el denominado Debugger). Cuenta además con una versión reducida del JDK, denominada JRE (*Java Runtime Environment*) destinada únicamente a ejecutar código Java (19).

Un IDE (Entorno de Desarrollo Integrado), tal y como su nombre indica, es un entorno de desarrollo integrado. En un mismo programa es posible escribir el código Java, compilarlo y ejecutarlo sin tener que cambiar de aplicación. Estos entornos integrados permiten desarrollar las aplicaciones de forma mucho más rápida, incorporando en muchos casos librerías con componentes ya desarrollados, los cuales se incorporan al proyecto o programa, proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación como C++, Java, C#, Delphi, Visual Basic, Object Pascal, Velneo y entre otros (19).

1.5.1. NetBeans

NetBeans IDE es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Ofrece a los desarrolladores numerosas ventajas, en la creación de nuevas aplicaciones multiplataforma. Es un producto de código abierto tanto para su uso comercial como no comercial, con todos los beneficios del software disponible en forma gratuita y sin restricciones (20).

En su núcleo, es una herramienta de desarrollo Java, escrita puramente sobre la base de dicha tecnología, de modo que puede ejecutarse en cualquier ambiente que ejecute la misma. Las amplias posibilidades de desarrollo multiplataforma de NetBeans atraen a muchos desarrolladores que han trabajado con otras herramientas. Posee numerosas características que hacen que el IDE sea atractivo para cualquier desarrollador, incluyendo la amplia integración de las características específicas de la tecnología Java que no se encuentran disponibles en otros conjuntos de herramientas de aplicaciones multiplataforma (21).

Entre sus principales características se encuentran facilidad de uso, cumplimiento de regulaciones, perfiles de rendimiento, flexibilidad entre plataformas y la capacidad de desarrollar de manera eficiente las aplicaciones Java, lo que permite que el código fuente se modifique automáticamente junto con los cambios de modelo y elimine la necesidad de los desarrolladores de tener que referirse constantemente a los comentarios de este (21).

1.6. Metodologías de desarrollo de software

Una metodología para el desarrollo de un proceso de software, es un conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de Sistemas Informáticos (22).

En los últimos tiempos la cantidad y variedad de metodologías de desarrollo de software ha aumentado de forma impresionante, se podría decir que en estos últimos años se han desarrollado dos corrientes en lo referente a estas metodologías, las llamadas metodologías pesadas y metodologías ágiles. La diferencia fundamental entre ambas es que mientras las pesadas intentan conseguir el objetivo común por medio del orden y documentación, las ágiles (o ligeras como también se le llaman) tratan de mejorar la calidad del software por medio de una comunicación directa e inmediata entre las personas que intervienen en el proceso.

Entre las metodologías más utilizadas en la actualidad se encuentra: RUP (Proceso Unificado de Software), el mismo es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye una metodología estándar muy utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos que son los productos tangibles del proceso como el modelo de casos de uso, el código fuente, etc., y roles, que no es mas que el papel que desempeña una persona en un determinado momento (23). La metodología RUP es más apropiada para proyectos grandes ya que requiere de un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios (24).

1.7. Programación Extrema (XP)

XP es la metodología de desarrollo de software seleccionada para el desarrollo de este trabajo ya que está diseñada principalmente para proyectos a corto plazo y requiere de un equipo pequeño de trabajo. Se caracteriza por potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y simplicidad en las soluciones implementadas. XP se define como especialmente adecuada para proyectos con requisitos imprecisos, muy cambiantes y con alto riesgo técnico. Exige, dentro del equipo de desarrollo, la existencia permanente de un representante competente del cliente, con lo que intenta minimizar las probabilidades de fallo del proceso y garantizando, además, un mínimo de inconformidades de este último con el producto final.

Los roles que define esta metodología son:

1. **Programador**: Escribe las pruebas unitarias y produce el código del sistema.
2. **Cliente:** Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
3. **Encargado de pruebas (Tester):** Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
4. **Encargado de seguimiento (Tracker):** Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración
5. **Entrenador (Coach):** Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
6. **Consultor:** Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.

7. **Gestor (Big boss):** Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

El ciclo de vida ideal de XP consiste de cuatro fases:

1. **Exploración:** En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.

2. **Planificación de la Entrega (*Release*):** En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.

3. **Implementación:** El desarrollo es la parte más importante en el proceso de la programación extrema. Todos los trabajos tienen como objetivo que se programen lo más rápidamente posible, sin interrupciones y en dirección correcta. También es muy importante el diseño, y se establecen los mecanismos, para que éste sea revisado y mejorado de manera continuada a lo largo del proyecto, según se van añadiendo funcionalidades al mismo.

4. **Prueba:** Las pruebas son realizadas mediante test para comprobar el funcionamiento del código que será implementado. Se deben crear las aplicaciones que realizarán los test con un entorno de desarrollo específico. Un punto importante es crear test que no tengan ninguna dependencia del código que en un futuro evaluará.

Entre sus principales características se encuentran:

1. **Pruebas unitarias** continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión: Generalmente se escribe el código de la prueba antes de la codificación.

2. **Programación en parejas:** Las tareas de desarrollo generalmente son llevadas a cabo por equipos conformados por dos personas, garantizando así mayor calidad del código escrito y, como consecuencia, del producto en general.

3. **Frecuente integración del equipo de programación con el cliente o usuario:** Un representante del cliente trabaja con el equipo de desarrollo.

4. **Corrección de todos los errores antes de añadir nueva funcionalidad:** Conlleva a hacer entregas frecuentes.

5. **Refactorización del código:** Se reescriben ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento.

6. **Propiedad del código compartida:** Promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto.

1.8. Conclusiones

En este capítulo se han abordado todos los puntos fundamentales que constituirán la base para el desarrollo del trabajo, por lo que se puede concluir que:

1. Las aplicaciones existentes aunque no constituyen la solución al problema brindan una visión optimista del software final que se desea obtener.

2. Las tecnologías y metodologías seleccionadas contribuyen a lograr la solución del problema de forma eficiente, posibilitando así, un mejor entendimiento del mismo.

3. La herramienta de desarrollo seleccionada se destaca por la ventaja de crear aplicaciones multiplataforma, aspecto fundamental por la migración hacia el software libre, además de proporcionar los elementos necesarios para establecer una excelente comunicación entre aplicaciones, la cual se realizará a través de Socket TCP/IP.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA. EXPLORACION Y PLANIFICACION

2.1. Introducción

En el presente capítulo se realiza una descripción general de la propuesta de sistema, teniendo como objetivo fundamental comprender y entender los problemas actuales por lo que es necesaria su realización. Partiendo de este análisis se da paso a las dos primeras fases de la metodología XP: Exploración y Planificación, en las que quedarán plasmadas las funcionalidades del sistema mediante las historias de usuario, dando una idea de cuáles son los requisitos que el cliente desea cumpla el producto una vez finalizado. Se definirá además el tiempo total dedicado al proyecto con el objetivo de que la entrega de este se realice sin retrasos en la fecha solicitada.

2.2. Objeto de Estudio.

2.2.1. Problema y situación problémica.

La universidad fomenta, en cada una de las actividades docentes que desarrolla, el uso de las tecnologías, apoyando así el desarrollo tecnológico evidenciado en el país en los últimos años. Es por ello que se desarrollan diariamente métodos que de una forma u otra mejoran el proceso docente - educativo del centro, siendo un ejemplo crucial la aplicación de exámenes en los laboratorios.

En este caso, la metodología que se ha estado llevando a cabo ha tenido gran aceptación, puesto que esta representa una forma mucho más eficiente de evaluar el nivel de aprendizaje de los estudiantes. Sin embargo, al realizar un examen, los profesores encargados de evitar que el estudiante realice una actividad fuera de lo establecido, se enfrentan a una serie de problemas dificultando su trabajo debido, fundamentalmente, al gran número de computadoras existentes en el laboratorio.

Otro aspecto a tener en cuenta es la entrega y recogida de estos exámenes, procesos que actualmente son llevados a cabo por el profesor, quien se encarga de ir por cada puesto copiando, en el primer caso hacia cada computadora y en el segundo, hacia un dispositivo de almacenamiento externo, dirección IP o sitio web destinado para ello, las pruebas, lo cual ocupa tiempo no establecido para la evaluación.

El centro no cuenta con un sistema capaz de solucionar todo lo antes expuesto, es por ello que se hace necesario implementar una herramienta que contenga las funcionalidades requeridas para controlar, eficientemente, estas evaluaciones.

2.2.2. Flujo actual de los procesos.

Actualmente el proceso de aplicación de los exámenes en los laboratorios se realiza a través de la plataforma Moodle, la cual es una herramienta muy efectiva fundamentalmente en el desarrollo de exámenes teóricos. En el caso particular de las pruebas prácticas permite la descarga del archivo correspondiente a la misma, esto presenta varias desventajas, siendo ejemplo de estas el hecho de que una vez abierto el sitio puede ocurrir que el examen no se encuentre disponible, siendo necesario que el estudiante espere hasta su publicación provocando que el horario de inicio del mismo no coincida con el establecido.

Estas evaluaciones son gestionadas por un servidor central por lo que, en ocasiones, debido al gran número de estudiantes conectados al sitio, provoca que el proceso de navegación y descarga desde el mismo se torne demasiado lento.

Generalmente por cada laboratorio son ubicados a lo sumo dos profesores, que son los encargados de controlar las actividades que realizan los estudiantes en su puesto de trabajo. Cuando el examen no se encuentra publicado en el Moodle, en el mejor de los casos se cuenta con una dirección IP que los contiene, en caso contrario se requiere de un dispositivo externo que se irá conectando en cada computadora. El mismo proceso es necesario llevarlo a cabo a la hora de recoger las evaluaciones, tornándose tedioso y dificultando el trabajo de las personas responsables en cada laboratorio.

2.2.3. Objetivos estratégicos.

El Moodle es una herramienta cuyo objetivo fundamental es la gestión y publicación de las evaluaciones que se realizan en la universidad. Entre sus funcionalidades más importantes podemos encontrar la descarga de los archivos contenedores del examen, control del tiempo de duración del mismo y la facilidad de la aplicación de pruebas teóricas directamente desde la red.

A pesar de las muchas ventajas y facilidades que brinda, esta plataforma posee también desventajas notables, entre las que podemos encontrar la necesidad que existe de desconectar la red a la hora de

realizar exámenes prácticos, puesto que es imposible controlar la actividad de cada estudiante en su puesto de trabajo, siendo esta una forma efectiva aunque no eficiente de evitar el fraude.

El sistema a automatizar se propone como objetivos fundamentales monitorear y controlar estas actividades sin que se requiera la previa desconexión de la red; distribuir los exámenes a realizar en cada una de las computadoras sin necesidad de descargarlos y agilizar el proceso de recogida de los mismos una vez finalizado el tiempo señalado.

2.2.4. Análisis crítico del funcionamiento del sistema actual.

El funcionamiento de los procesos a automatizar debe verse desde las siguientes vertientes:

- Actualmente no existe ningún sistema capaz de monitorear y controlar los exámenes que se realizan en los laboratorios de la universidad, siendo este un proceso no eficiente llevado a cabo por los profesores.
- No existe una vía que posibilite la recogida de estos exámenes de forma rápida puesto que actualmente esta acción se realiza manualmente.

Un sistema automatizado, capaz de garantizar la distribución y recogida de las evaluaciones, además de realizar todo el proceso de monitoreo y control de las mismas, mejoraría el trabajo de los profesores encargados de estos exámenes, siendo esta una vía factible para chequear eficientemente lo que realiza cada estudiante en su computadora, además de garantizar que se cumpla con el tiempo destinado al mismo.

2.2.5. Objeto de automatización.

Inicialmente se tiene en cuenta la necesidad de llevar a cabo un control exhaustivo de los exámenes desarrollados en los laboratorios de la universidad. Para ello es necesario contar con una aplicación que permita monitorear cada una de las computadoras durante el tiempo de duración de los mismos. De esta forma se garantizará el mantener informado al profesor encargado, de las actividades que realiza cada estudiante desde su puesto de trabajo.

El sistema permitirá además entregar y recoger dichos exámenes a través de la red y que el tiempo de demora de cada estudiante durante el desarrollo de los mismos sea exactamente el establecido para esta actividad.

2.2.6. Propuesta de Sistema

El sistema a automatizar está compuesto por dos aplicaciones, encargadas de controlar las actividades que se realizan en la computadora desde el momento en que inicia el examen. Esto se logra mediante la comunicación entre ambas herramientas, la cual se realiza a través de Socket TCP\IP, la siguiente figura muestra el funcionamiento del sistema.

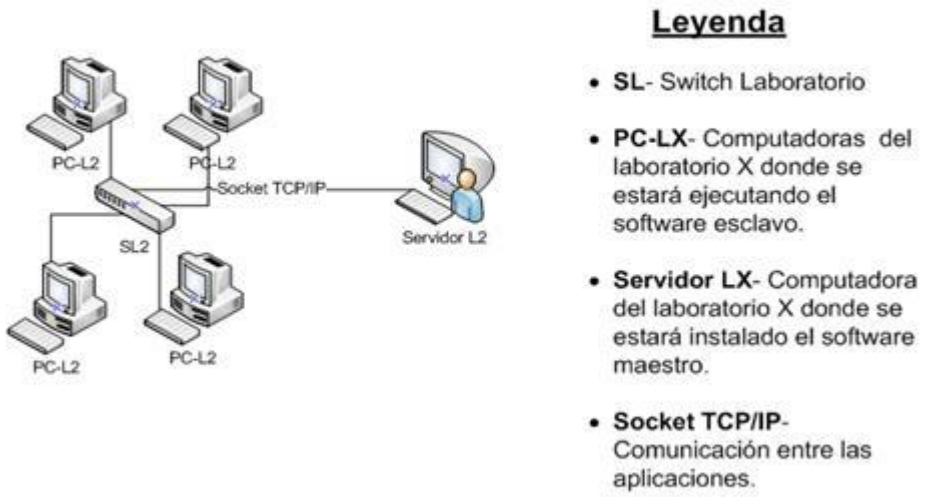

Figura 2. 1. Funcionamiento del sistema.

El software maestro es el encargado de dirigir todo el proceso de monitoreo y control. Cuenta con una interfaz gráfica, flexible y amigable para que pueda ser utilizada fácilmente por la persona autorizada, brindando las funcionalidades necesarias que el subsistema deberá realizar. Es la parte del software con la que interactuaría el usuario. Su mayor responsabilidad es la de enviar las peticiones al software esclavo, el cual se caracteriza por ser un proceso invisible, de forma tal que nadie pueda acceder a él. Su

objetivo fundamental es monitorear las actividades que realiza el estudiante en su computadora, garantizándole al software maestro poder visualizar lo referente a las operaciones que realiza el mismo desde su puesto de trabajo, como conectar un dispositivo externo.

2.2.7. Características de la propuesta de sistema

En el presente trabajo se propone la implementación de un sistema que permita fundamentalmente monitorear y controlar la actividad de cada estudiante en su puesto de trabajo en un examen determinado. El mismo estará programado en java y será multiplataforma, por lo que podrá ser visualizado en sistemas operativos tales como versiones de Windows y Linux.

El software estará disponible mientras exista conexión entre ambos subsistemas, dígase aplicación maestro y aplicaciones esclavos, la misma se realizará a través de la comunicación Sockets TCP/IP y será gestionada por un *switch* ubicado en cada uno de los laboratorios. Contará con una interfaz fácil de usar y amigable para que pueda ser utilizada fácilmente por el usuario; podrá ser utilizado solo por las personas autorizadas para ello y contendrá un menú en donde estarán visibles las funcionalidades del mismo, facilitando así la distribución de las mismas y la localización de estas por el usuario.

Para lograr eficiencia en la distribución de los exámenes hacia las computadoras determinadas se utilizará la sincronización de múltiples hilos de ejecución, que son útiles, especialmente, en la creación de aplicaciones de red distribuidas. Se requiere la instalación en cada computadora de la máquina virtual de java.

2.3. Fase de exploración

Esta es la fase en la que se especifica el alcance general del proyecto. Se definen las funcionalidades necesarias que debe cumplir el sistema mediante la redacción de sencillas “historias de usuarios”. En base a esta información es estimado el tiempo de desarrollo del software (25). El equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo (26).

2.3.1. Historias de Usuario

Uno de los artefactos más importantes que genera la metodología XP son las Historias de Usuario. Estas tienen el mismo propósito que los casos de uso y son escritas por el propio cliente, tal y como ven ellos las necesidades del sistema, por tanto son descripciones cortas y escritas en el lenguaje del usuario sin terminología técnica.

Conducen al proceso de creación de los test de aceptación, los cuales servirán para verificar que estas historias se han implementado correctamente. Otra de sus características es que solamente proporcionan los detalles sobre la estimación del riesgo y cuánto tiempo conllevará su implementación. Su nivel de detalle debe ser el mínimo posible, de manera que permita hacerse una ligera idea de cuánto costará implementar el sistema.

El software en cuestión cuenta con un total de 7 historias de usuario, cada una de ellas respondiendo a las diferentes funcionalidades solicitadas por el cliente y dando una idea al resto del equipo de desarrollo de cómo debe ser su posterior implementación.

Historia de Usuario	
Número: 1	**Usuario:** Profesor
Nombre historia: Conexión entre la aplicación maestro y las aplicaciones esclavos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos Estimados: 2	**Iteración asignada:** 1
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: La aplicación maestro se conectará a las aplicaciones esclavos a través de la comunicación vía Sockets TCP/IP.	
Observaciones:	

Tabla 2. 1.Historia de usuario "Conexión entre la aplicación maestro y las aplicaciones esclavos".

Historia de Usuario	
Número: 2	**Usuario:** Profesor
Nombre historia: Eliminar archivos.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos Estimados: 0.5	**Iteración asignada:** 4
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Dada una petición de la aplicación maestro a las aplicaciones esclavas, estas últimas se encargarán de eliminar los archivos existentes en cada uno de los escritorios de las computadoras conectadas.	
Observaciones:	

Tabla 2. 2. Historia de usuario "Eliminar Archivos".

Historia de Usuario	
Número: 3	**Usuario:** Profesor
Nombre historia: Distribuir examen	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Puntos Estimados: 2	**Iteración asignada:** 3
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: La aplicación maestro se encargará, apoyándose en las aplicaciones esclavos, de distribuir los exámenes en cada una de las computadoras conectadas. Debe ser especificado el archivo que se desea distribuir.	
Observaciones:	

Tabla 2. 3. Historia de Usuario "Distribuir Examen".

Historia de Usuario	
Número: 4	**Usuario:** Profesor
Nombre historia: Establecer tiempo de duración del examen	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.5	**Iteración asignada:** 4
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Después de ser distribuidos los exámenes se establece el tiempo destinado para su desarrollo. Este se mostrará en la pantalla de cada una de las computadoras.	
Observaciones:	

Tabla 2. 4. Historia de Usuario "Establecer tiempo de duración del examen".

Historia de Usuario	
Número: 5	**Usuario:**
Nombre historia: Monitoreo de la actividad en cada una de las computadoras	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Iteración asignada: 2**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Las aplicaciones esclavos se mantendrán constantemente monitoreando la actividad de la computadora correspondiente y notificará a la aplicación maestro en caso de que un usuario conecte cualquier dispositivo de almacenamiento externo.	
Observaciones:	

Tabla 2. 5. Historia de usuario "Monitoreo de la actividad en cada una de las computadoras".

Historia de Usuario	
Número: 6	**Usuario:**
Nombre historia: Deshabilitar teclado	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada: 5**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Una vez culminado el tiempo establecido para el examen, cada aplicación Esclavo se encargará de deshabilitar el teclado de la computadora correspondiente.	
Observaciones:	

Tabla 2. 6. Historia de usuario "Deshabilitar teclado".

Historia de Usuario	
Número: 7	**Usuario:** Profesor
Nombre historia: Recoger exámenes	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimado: 1	**Iteración asignada:** 3
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Una vez culminado el examen, las aplicaciones esclavas enviarán los mismos a la aplicación maestro y este se encargará de copiarlos a la dirección especificada.	
Observaciones:	

Tabla 2. 7. Historia de Usuario "Recoger Exámenes". Fase de Planificación

La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. Es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas (25).

2.3.2. Estimación de esfuerzo por historia de usuario

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Este se expresa utilizado como medida el punto. Un punto se considera como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción (27). Los resultados obtenidos en esta estimación se exponen en la siguiente tabla:

Historias de Usuario	Puntos estimados
Conexión entre la aplicación maestro y las aplicaciones esclavos.	2
Eliminar archivos	0.5
Distribuir examen	2
Establecer tiempo de duración del examen	0.5
Monitoreo de la actividad en cada una de las computadoras.	3
Deshabilitar teclado	3
Recoger exámenes	1

Tabla 2. 8. Estimación de esfuerzo por historias de usuario

2.3.3. Plan de Iteraciones

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada historia de usuario se establecen las pruebas de aceptación.

Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores. Las

pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir (25).

Iteración 1

En esta primera iteración se implementan las historias de usuario con mayor prioridad, obteniendo al final de la misma una primera versión de prueba y dando al sistema las primeras funcionalidades, centrándose en la conexión entre la aplicación maestro con las esclavas.

Iteración 2

En la segunda iteración se realiza la implementación de las historia de usuario con prioridad alta, relacionadas con el proceso de monitoreo de las actividades que se realizan en la computadora, debido a que la misma requiere de mucho tiempo de trabajo y cada iteración no debe sobrepasar 3 semanas para su desarrollo. De esta forma se obtiene la segunda versión de pruebas del software.

Iteración 3

La tercera iteración está basada en las historias de usuario con prioridad media, obteniendo al final de la misma las funcionalidades referidas a la gestión de exámenes que, agregadas a las obtenidas en la iteración anterior, constituyen los requerimientos fundamentales que debe tener el sistema. Se obtiene además la versión número tres de pruebas.

Iteración 4

En la cuarta iteración se desarrollan las historias de usuario con menor prioridad, se agregan a las funcionalidades ya implementadas las concernientes a la eliminación de archivos y el establecimiento del tiempo de duración del examen.

Iteración 5

En la quinta iteración ya implementadas las funcionalidades especificadas se realiza el desarrollo la última historia de usuario correspondiente a deshabilitar teclado obteniendo así la versión 1.0 del producto final.

2.3.4. Plan de duración de las iteraciones

Este plan se realiza con el objetivo de reflejar cuales serán las historias de usuario que serán implementadas en cada una de las iteraciones, así como el tiempo destinado a cada una de ellas y e orden en que se implementaran, lo que ayuda a obtener una idea general del tiempo que durara la confección total del sistema.

Iteración	Orden de la historias de usuario a implementar	Duración total de la iteración
Iteración 1	1. Conexión entre la aplicación maestro y las aplicaciones esclavos.	2 semanas
Iteración 2	1. Monitoreo de la actividad en cada una de las computadoras	3 semanas
Iteración 3	1. Distribuir examen. 2. Recoger exámenes	3 semanas
Iteración 4	1. Eliminar archivos 2. Establecer tiempo de duración del examen	1 semana
Iteración 5	1. Deshabilitar teclado	3 semanas

Tabla 2. 9. Plan de Iteraciones

2.3.5. Plan de entregas

El cronograma de entregas establece las fechas para las cuales se planifica serán entregadas cada una de las iteraciones y no es más que el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.); XP denomina a esta reunión “Juego de planeamiento”.

Este cronograma se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores. Luego de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario.

Historias de Usuarios	Final 1ra Iteración 3ra semana de febrero	Final 2da Iteración 2da semana de marzo	Final 3ra Iteración 1ra semana de abril	Final 4ta Iteración 2da semana de abril	Final 5ta Iteración 1ra semana de mayo
1	0.1				
5		0.2			
3,7			0.3		
2,4				0.4	
6					1.0

Tabla 2. 10. Plan de entregas

2.4. Conclusiones

El presente capítulo ha sentado las bases para el desarrollo de la propuesta de sistema a implementar, el estudio minucioso, objetivo y crítico del flujo actual y ejecución de los procesos, permitió que se lograra obtener una acertada descripción de cómo funciona en la actualidad los procesos a automatizar y las deficiencias existentes.

CAPÍTULO III: DISEÑO

3.1. Introducción

Luego de las fases de Exploración y Planificación, abordadas en el capítulo anterior, la metodología XP, define la fase de diseño, cuyo objetivo fundamental es dejar plasmadas las descripciones de cada una de las clases que será implementada posteriormente, además de diseñar las interfaces que conformaran el software.

El desarrollo de este capítulo se basa fundamentalmente en la construcción tanto de la interfaz gráfica de la aplicación como de las tarjetas CRC, artefacto fundamental generado por esta fase y que constituyen la guía para la posterior codificación de la aplicación.

3.2. Interfaz de usuario

La creación de las interfaces de usuario ha sido un área del desarrollo de software que ha evolucionado dramáticamente a partir de la década de los setentas. La interfaz de usuario es el vínculo entre el usuario y el programa de computadora. Una interfaz es un conjunto de comandos o menús a través de los cuales el usuario se comunica con el programa (28).

La elaboración de una interfaz de usuario, bien diseñada, exige una gran dedicación pues generalmente las interfaces son grandes, complejas y difíciles de implementar, depurar y modificar. Hoy en día las interfaces de manipulación directa (también llamadas interfaces gráficas de usuario, GUI por sus siglas en inglés) son prácticamente universales. Las interfaces que utilizan ventanas, íconos y menús se han convertido en estándar en las aplicaciones que se desarrollan actualmente (29).

La interfaz representa el punto de encuentro entre el usuario y la computadora. En esta interacción, el usuario juzga la utilidad de la interfaz; el hardware y el software se convierten en simples herramientas sobre los cuales fue construida la interfaz. La definición de interfaz en sí misma es un tanto arbitraria, aunque esto depende de la naturaleza de la tarea que se tiene enfrente.

3.2.1. SMCPrincipal

SMCPrincipal constituye la interfaz principal del software y pertenece, específicamente, a la aplicación Maestro. Su diseño comprende todas las funcionalidades que será capaz de brindar el software y a través de ella, según sea el caso, se irán mostrando todas las interfaces que conforman el proyecto en general.

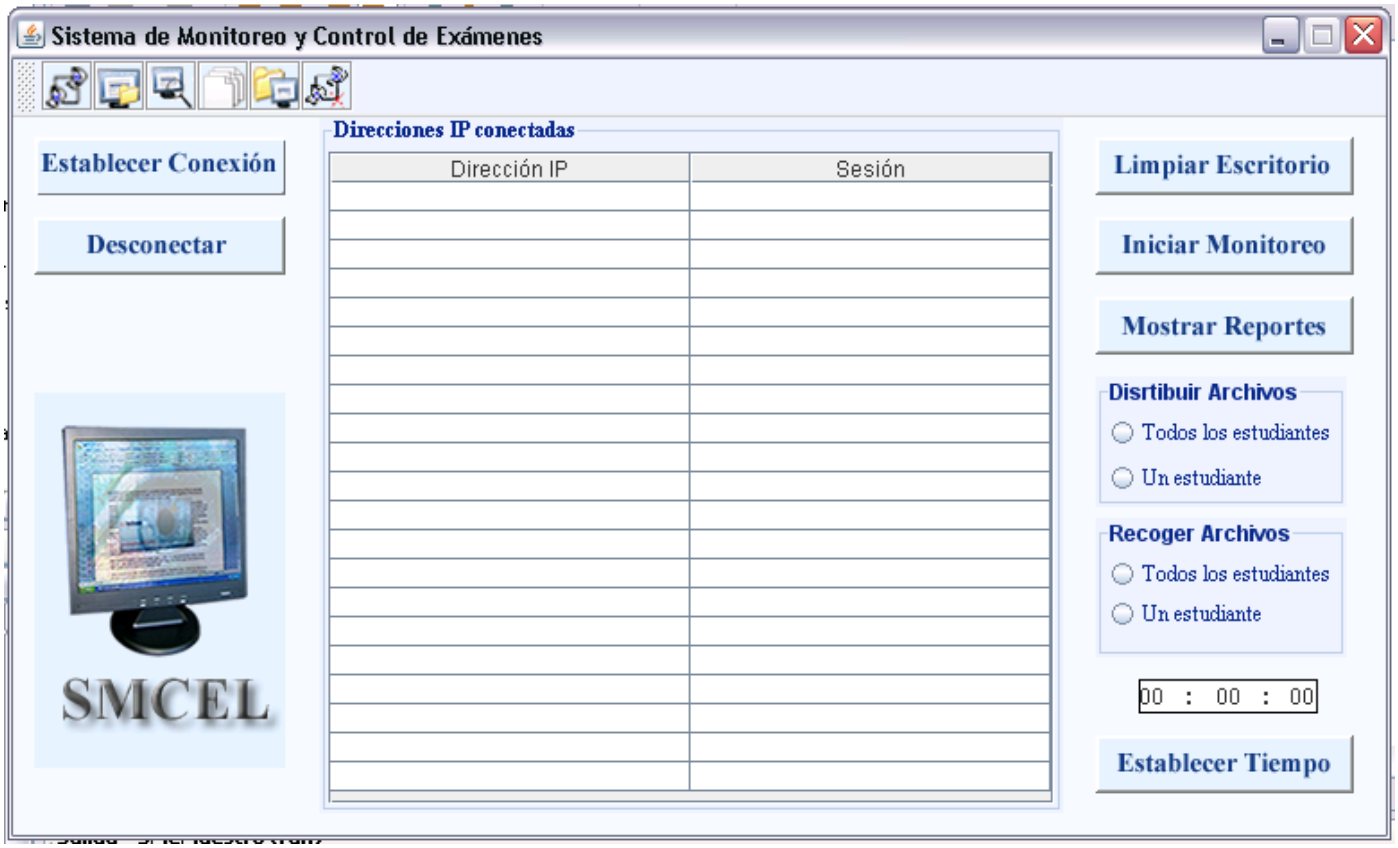

Figura 3. 1. Interfaz principal del software: "SMCPrincipal". Aplicación Maestro

3.2.1.1. Establecer Conexión

Este botón responde a la funcionalidad: Conexión entre la aplicación maestro y las aplicaciones esclavos. Realiza las llamadas a los métodos encargados de conectar ambas aplicaciones, realizando tantas conexiones como direcciones IP existan en el laboratorio donde se estará realizando el examen.

Las computadoras que resulten conectadas serán mostradas en la tabla que se visualiza en el centro de la interfaz. Una vez que la aplicación Esclavo recibe la petición de conexión, muestra el siguiente formulario mediante el cual, la persona que hará uso de la PC, insertará su usuario, dato que será mostrado también en la aplicación Maestro.

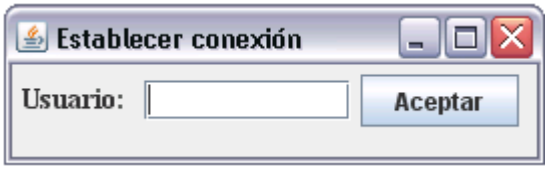

Figura 3. 2. Interfaz Establecer Conexión. Aplicación Esclavo

3.2.1.2. Limpiar Escritorio

Este botón realiza las llamadas al método que se encargara de eliminar todos los archivos que existan en las computadoras que se encuentren conectadas, respondiendo a la funcionalidad: Eliminar archivos.

3.2.1.3. Distribuir Archivos

En este caso, tenemos dos opciones, el usuario puede escoger entre distribuir los archivos a todas las computadoras (Todos los estudiantes), o solo a uno (Un estudiante), en cuyo caso deberá marcar en la tabla la dirección IP y usuario al que se le desea enviar el fichero. Para ambos casos es mostrada una interfaz cuyo diseño es el siguiente:

Figura 3. 3. Interfaz SMCDistribuirExamen. Aplicación Maestro

El campo “Archivo” no es más que la especificación del fichero que se desea distribuir, la misma es hecha por el usuario al pulsar el botón “Examinar”, “Distribuir” se encarga de llamar a los métodos correspondientes a la funcionalidad, que en este caso es: Distribuir examen.

3.2.1.4. Recoger Archivos

Al igual que en el caso anterior, a la hora de recoger el usuario puede escoger cualquiera de las dos opciones que se especifican en la Interfaz principal. En este caso se responde a la funcionalidad: Recoger exámenes y requiere del siguiente formulario:

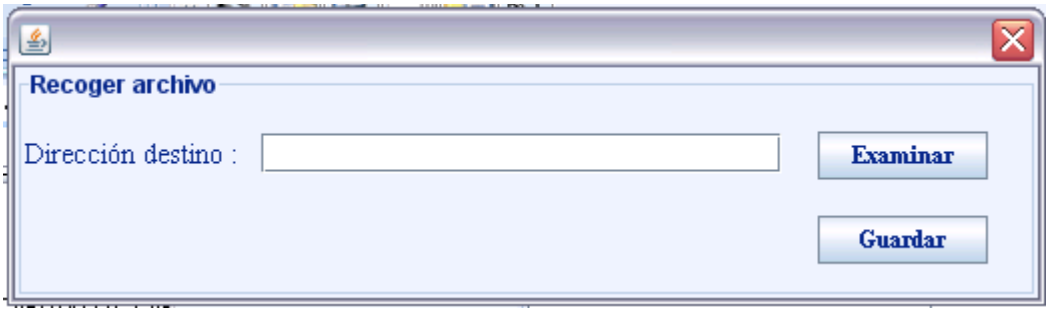

Figura 3. 4. Interfaz SMCRecogerArchivos. Aplicación Maestro

En el mismo se especifica la dirección donde deben ser copiados los archivos que el usuario desea recibir, el botón “Guardar” se encarga de realizar las llamadas a los métodos correspondientes, según sea el caso.

3.2.1.5. Iniciar Monitoreo

Este botón responde a la funcionalidad: Monitoreo de la actividad en cada una de las computadoras y, como su nombre lo indica, es el encargado de iniciar este proceso. El resultado de monitorear los dispositivos de almacenamiento externo que son insertados en las computadoras se muestra a través de la Interfaz SMCReportes, la misma se visualiza una vez pulsado el botón “Mostrar Reportes”. Su diseño es el siguiente:

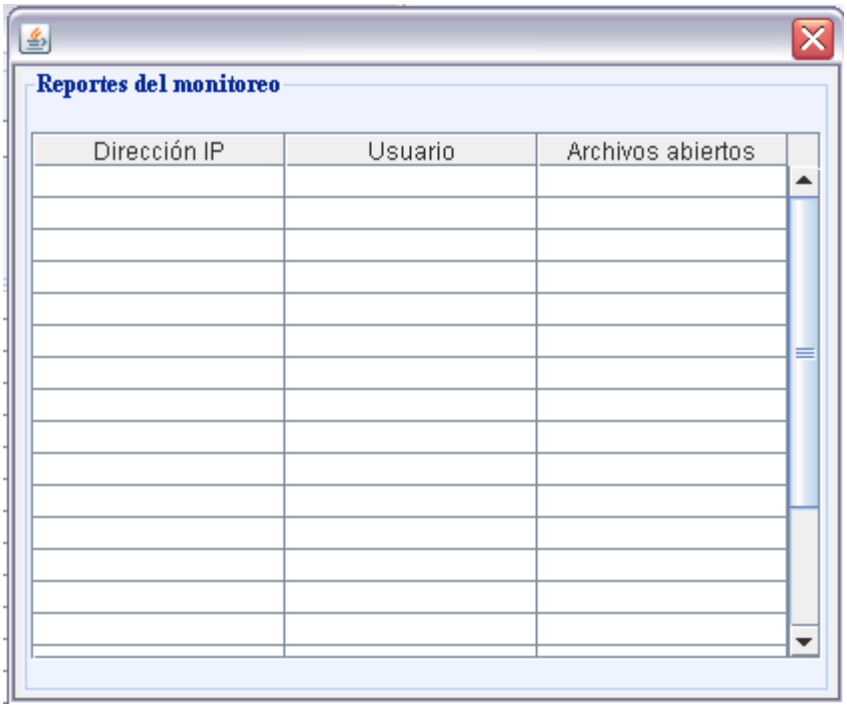

Figura 3. 5. Interfaz SMCReportes. Aplicación Maestro

Esta interfaz muestra una tabla que contendrá la dirección IP de la computadora a la que corresponda el reporte, el usuario que se encuentra trabajando en la misma y el dispositivo colocado.

3.2.1.6. Establecer Tiempo

Este botón responde a la funcionalidad: Establecer tiempo de duración del examen y se encarga de, una vez recibidos los datos correspondientes al tiempo que debe durar el examen, iniciar el conteo regresivo del mismo e ir mostrando esta información en pantalla. Una vez que la aplicación Esclavo reciba la orden correspondiente, mostrara en la esquina inferior derecha de la pantalla el siguiente formulario:

Figura 3. 6. Formulario Tiempo. Aplicación Esclavo

3.2.1.7. Desconectar

Une vez pulsado este botón se procede a eliminar la comunicación entre ambas aplicaciones.

3.3. Descripción de las clases del diseño.

La metodología XP sugiere que se deben realizar diseños simples y sencillos, donde se debe procurar hacerlo todo, lo menos complicado posible, para conseguir un diseño fácil de entender e implementar (30).

Para el diseño de las aplicaciones esta metodología se basa en el uso de las tarjetas CRC (Clases, Responsabilidades y Colaboración), cuyo objetivo es facilitar la comunicación y documentar los resultados. Estas contienen la información del nombre de la clase, sus responsabilidades, que es todo lo que una clase realiza como sus métodos y atributos, y las colaboraciones, son aquellas clases que se utilizan para obtener información.

3.3.1. Tarjetas CRC de la aplicación Maestro.

Nombre de la Clase	SMCMaestro	
Tipo de Clase	Controladora	
Descripción		
Es la encargada de llevar el control de la aplicación Maestro, contiene y manipula los atributos que hacen persistir los datos en memoria, mientras esta se encuentra en ejecución. Es responsable de iniciar todas las operaciones que deben ser ejecutadas en el momento que el usuario se lo solicita a través de los controles gráficos de las clases interfaz.		
Colaboración		
Nombre de las Clases	RegConexion, Archivos, ControlMonitoreo, Reporte, Comandos	
Atributo		**Tipo**
listaRegistro listaArchivo		Vector<RegConexion> Vector<Archivo>
Responsabilidad		
Nombre	Vector<String> genDireccionIP()	
Descripción	Teniendo en cuenta una máscara determinada, genera todas las direcciones IP que puedan encontrarse dentro de la misma, devolviendo un listado con	

	cada una de ellas.
Nombre	void estConexion()
Descripción	Obtiene la lista de direcciones IP generadas y una vez creado un objeto de la clase RegConexion, se encarga de ejecutar el hilo correspondiente a la misma. Las conexiones establecidas se guardan en la lista de registros junto con todos sus datos.
Nombre	void distArchivo (Archivo archivo)
Descripción	Crea un objeto de la clase Archivo y se encarga de, una vez ejecutado el hilo del mismo, añadir el archivo pasado como parámetro a la lista correspondiente.
Nombre	void estTiempo(int horas,int minutos, int segundos)
Descripción	Se encarga de enviar a cada una de las aplicaciones esclavos los datos correspondientes al tiempo de duración del examen.
Nombre	void inicMonitoreo()
Descripción	Ejecuta el hilo correspondiente a la clase ControlMonitoreo y guarda cada uno de los objetos creados en la lista correspondiente.
Nombre	void elimArchivo()
Descripción	Obtiene cada uno de los clientes de la lista de registros y envía a cada una de las aplicaciones Esclavos la orden de limpiar el escritorio.
Nombre	Vector<Reporte> obtReportes()
Descripción	Obtiene la lista de reportes correspondiente a cada uno de los clientes conectados
Nombre	void desconectar(String dirIP)
Descripción	Dada una dirección IP determinada, se encarga de desconectar el cliente correspondiente a la misma.

Tabla 3. 1. Tarjeta CRC de la clase "SMCMaestro".

Nombre	RegConexion
Tipo de Clase	Entidad
Descripción	
Hereda de la clase Thread y es la encargada de establecer la conexión dada una dirección IP especificada. Contiene los datos de la dirección IP, usuario y cliente de la conexión establecida.	

Colaboración	
Nombre de las Clases	InetAddress, Socket
Atributo	**Tipo**
dirIP usuario cliente comando	String String Socket Comandos
Responsabilidad	
Nombre	void run()
Descripción	Es un método abstracto de la clase Thread, que es redefinido. Crea un Socket cliente, especificándole la dirección IP y puerto, con los que se requiere establecer la conexión.

Tabla 3. 2. Tarjeta CRC de la clase "RegConexion".

Nombre	ControlMonitoreo
Tipo de Clase	Entidad
Descripción	
Hereda de la clase Thread y es la encargada de mantenerse esperando los reportes enviados por la aplicación Esclavo y agregarla a la lista correspondiente.	
Colaboración	
Nombre de las Clases	Socket
Atributo	**Tipo**
dirIP usuario cliente listaReportes unidNuevas	String String Socket Vector<Reporte> Vector<String>
Responsabilidad	
Nombre	int buscarReportes(String unidad)
Descripción	Se encarga de buscar en la lista de reportes, la unidad recibida desde la aplicación Esclavo.
Nombre	void run()

Descripción	Es un método abstracto de la clase Thread, que es redefinido. Es el encargado de esperar los reportes y lo agrega a la lista de reportes en caso d que no se encuentre ya en la misma.

Tabla 3. 3. Tarjeta CRC de la clase "ControlMonitoreo".

Nombre	Archivo	
Tipo de Clase	Entidad	
Descripción		
Hereda de la clase Thread y es la encargada de enviar el examen a la aplicación Esclavo.		
Colaboración		
Nombre de las Clases	Socket	
Atributo		**Tipo**
nomArchivo dirArchivo cliente		String String Socket
Responsabilidad		
Nombre	void run()	
Descripción	Es un método abstracto de la clase Thread, el cual es redefinido. Es el responsable de enviar el archivo al cliente determinado.	

Tabla 3. 4. Tarjeta CRC de la clase "Archivo".

Nombre	Reporte
Tipo de Clase	Entidad
Descripción	
Contiene los datos de los reportes enviados por la aplicación Esclavo	
Colaboración	
Nombre de las Clases	

Atributo	Tipo
dirIP	String
usuario	String
resultado	String

Tabla 3. 5. Tarjeta CRC de la clase "Reporte".

Nombre	Comandos
Tipo de Clase	
Descripción	
Contiene los diferentes comandos que representan las posibles peticiones que envía la aplicación Maestro a las aplicaciones Esclavos	

Tabla 3. 6. Tarjeta CRC de la clase "Comandos".

3.3.2. Tarjetas CRC de la aplicación Esclavo.

Nombre	SMCEsclavo
Tipo de Clase	Controladora
Descripción	
Es la encargada de llevar el control de la aplicación Esclavo mientras esta se encuentra en ejecución. Es responsable de iniciar todas las operaciones que deben ser ejecutadas en el momento que la aplicación Maestro lo solicita a través los Stream de entradas y salidas.	
Colaboración	
Nombre de las Clases	Tiempo, ContTiempo, Monitoreo, IntMonitoreo, LibTiempo, LibMonitoreo, Main
Atributo	**Tipo**
cliente	Socket
servidor	ServerSocket
usuario	String
comando	String
Responsabilidad	

Nombre	void elimArchivo(File archivo)
Descripción	Se encarga de eliminar los archivos existentes en el escritorio de la computadora correspondiente
Nombre	void distArchivos(String nomArchivo)
Descripción	Una vez recibido el archivo de la aplicación Maestro, esta se encarga de colocarlo en el escritorio de la computadora.
Nombre	void recArchivos(String nomArchivo)
Descripción	Una vez obtenido el nombre del archivo que se desea recoger, se encarga de enviarlo a la aplicación Maestro.
Nombre	void desconectar()
Descripción	Se encarga de desconectar la aplicación Esclavo
Nombre	void run()
Descripción	Se mantiene esperando a que lleguen las peticiones de la aplicación Maestro y dependiendo de esta, ejecuta el código correspondiente.

Tabla 3. 7. Tarjeta CRC de la clase "SMCEsclavo".

Nombre	Tiempo
Tipo de Clase	Entidad
Descripción	
Hereda de la clase TimerTask. Mantiene los datos del tiempo establecido para el desarrollo del examen y se encarga de mostrarlo en pantalla.	
Colaboración	
Nombre de las Clases	JFrame, JLabel, Timer, LibTiempo

Atributo	**Tipo**
segundos	int
minutos	int
horas	int
timer	Timer
label	JLabel
frame	JFrame

Responsabilidad	
Nombre	void run()
Descripción	Muestra, como un texto, un tiempo determinado sobre una ventana que crea esta misma clase, destinada para esto

Tabla 3. 8. Tarjeta CRC de la clase "Tiempo".

Nombre	ContTiempo	
Tipo de Clase	Entidad	
Descripción		
Manda ejecutar, cada 1 segundo, el método run() de la clase Tiempo.		
Colaboración		
Nombre de las Clases	Tiempo, Timer	
Atributo		**Tipo**
segundos, minutos, horas timer		int Timer

Tabla 3. 9. Tarjeta CRC de la clase "ContTiempo".

Nombre	LibTiempo
Tipo de Clase	
Descripción	
Carga la librería que obtiene las dimensiones de la pantalla de la computadora correspondiente.	
Colaboración	
Nombre de las Clases	JFrame, JLabel, Timer, LibTiempo
Responsabilidad	
Nombre	native int getInt();
Descripción	Método nativo implementado en la librería Tiempo_DLL que retorna las dimensiones de la pantalla

Tabla 3. 10. Tarjeta CRC de la clase "LibTiempo".

Nombre	IntMonitoreo	
Tipo de Clase		
Descripción		
Manda ejecutar, cada 60 segundos el método run() de la clase Monitoreo		
Colaboración		
Nombre de las Clases	Timer, Monitoreo	
Atributo		**Tipo**
Timer		Timer

Tabla 3. 11. Tarjeta CRC de la clase "IntMonitoreo".

Nombre	Monitoreo	
Tipo de Clase		
Descripción		
Hereda de la clase TimerTask. Es la que controla todo el monitoreo a la computadora correspondiente.		
Colaboración		
Nombre de las Clases	TimerTask, Timer, LibMonitoreo	
Atributo		**Tipo**
unidIniciales cliente timer monitoreo		char[] Socket Timer LibMonitoreo
Responsabilidad		
Nombre	boolean buscarCaracter(char car)	
Descripción	Dado un caracter determinado, se encarga de buscarlo en la lista de unidades iniciales, retornando true en caso de encontrarlo, y false en caso contrario.	
Nombre	Vector<String> compArrays(char[] nuevo)	

Descripción	Dado un arreglo de caracteres, se encarga de compararlo con el arreglo inicial que contiene las unidades ocupadas inicialmente en la computadora correspondiente. Retorna una lista con los caracteres no comunes en ambos arreglos.
Nombre	void run()
Descripción	Obtiene una lista con el resultado de la comparación de los arreglos, y si esta tiene longitud mayor que cero, se encarga de enviar cada carácter a la aplicación maestro, junto con el usuario y la dirección IP correspondientes.

Tabla 3. 12. Tarjeta CRC de la clase "Monitoreo".

Nombre	LibMonitoreo	
Tipo de Clase		
Descripción		
Carga la librería que obtiene las unidades ocupadas en la computadora correspondiente.		
Colaboración		
Nombre de las Clases		
Atributo	**Tipo**	
Responsabilidad		
Nombre	native char[] obtUnidExistentes();	
Descripción	Método nativo implementado en la librería Monitoreo que retorna las unidades ocupadas en la computadora correspondiente.	

Tabla 3. 13. Tarjeta CRC de la clase "LibMonitoreo".

Nombre	Main
Tipo de Clase	
Descripción	
Clase principal de la aplicación Esclavo	

Tabla 3. 14. Tarjeta CRC de la clase "Main".

3.4. Patrones de diseño

Los desarrolladores orientados a objetos con experiencia (y otros desarrolladores de software) acumulan un repertorio tanto de principios generales como de soluciones basadas en aplicar ciertos estilos que les guían en la creación de software. Estos principios y estilos, si se codifican con un formato estructurado que describa el problema y la solución, se les denominan patrones.

De manera más simple se puede decir que un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos.

Durante el desarrollo del sistema se usaron los patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades) los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (31). A continuación se hace una breve descripción de los mismos.

Experto: Este patrón se basa en asignar una responsabilidad al experto en información, la clase que cuenta con la información necesaria para cualquier responsabilidad.

Con su puesta en práctica se logra que se conserve el encapsulamiento, pues los objetos se valen de su propia información para hacer lo que se les pide. Las clases se hacen más fáciles de mantener y comprender. Además soporta un bajo acoplamiento, lo que favorece que el sistema sea mucho más robusto y fácil de comprender.

Creador: Con este patrón se resuelve el problema de saber quién debería ser responsable de crear una nueva instancia de alguna clase, pues se encarga de asignarle la responsabilidad a una clase de crear instancias de otra, ya sea agregando, utilizando, conteniendo, registrando o creando objetos de la otra clase.

Su uso permite un bajo acoplamiento y mejores oportunidades de reutilización.

Alta Cohesión: Con el uso de este patrón, se resuelve el problema de saber cómo mantener la complejidad dentro de límites manejables, pues es el encargado de asignar una responsabilidad de modo que la cohesión siga siendo alta.

Su puesta en práctica permite lograr un diseño claro y fácil de comprender. Además de que aumenta la capacidad de reutilización.

Bajo Acoplamiento: Este patrón resuelve la problemática de saber cómo dar soporte a una dependencia escasa y a un aumento de la reutilización, mediante la asignación de responsabilidades a las clases de manera que la comunicación entre ellas sea la menor posible

Su uso permite, aumentar la reutilización, que el diseño sea fácil de entender, además de que no se afecte el diseño por cambios en otros componentes.

También se hizo uso de los patrones GOF (del inglés: Gang of Four). A continuación se describen los mismos:

Observer: Este patrón es de comportamiento, define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y se actualicen automáticamente todos los objetos que dependen de él.

Su uso permite lograr un acoplamiento mínimo entre sujetos y observadores, pueden pertenecer a diferentes capas de abstracción de un sistema. Además permite tener una capacidad de comunicación mediante difusión, es decir, libertad de añadir y quitar observadores en cualquier momento, lo cual constituye una ventaja.

3.5. Conclusiones

La simplicidad y la sencillez a la hora de elaborar tanto las clases como las interfaces de una aplicación determinada, logra que se consiga un diseño fácil de entender e implementar que a la larga, costará menos tiempo y esfuerzo desarrollar. El software en cuestión cuenta con un total de 6 interfaces y 14 clases, descritas en detalle a lo largo de todo el capitulo, las mismas tienen como objetivo ser la base de las fases posteriores que define la metodología XP.

CAPÍTULO IV: IMPLEMENTACION Y PRUEBAS

4.1. Introducción

XP plantea que la implementación de un software debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para retroalimentar a los desarrolladores con la opinión de este (27). El presente capítulo se basa en las dos últimas fases generadas por la metodología XP, Implementación y Pruebas, detallando, por cada iteración, las tareas generadas por las Historias de Usuario correspondientes, así como las Pruebas de aceptación efectuadas al software.

4.2. Implementación

Esta fase se caracteriza por el constante intercambio entre el equipo de desarrollo y el cliente, puesto que este último debe estar presente durante todo el tiempo que dure la codificación del software logrando con esto que cada una de las historias de usuario cumplan con todas las especificaciones hechas inicialmente, o sea, que el producto, una vez finalizado, sea exactamente lo que el cliente desea.

4.2.1. Iteración 1

La primera iteración está centrada en las Historias de Usuario de mayor prioridad, construyéndose de esta forma la base para la realización de las posteriores funcionalidades.

Historia de Usuario	Estimación	Real
Conexión entre la aplicación Maestro y las aplicaciones Esclavos.	2	2
Total	2	2

Tabla 4. 1. Historias de Usuario Iteración 1

4.2.1.1. Tareas de las Historias de Usuario desarrolladas en la primera iteración

Historia de usuario: Conexión entre la aplicación Maestro y las aplicaciones Esclavos.

	Tarea
Número tarea: 1	**Número historia: 1**
Nombre tarea: Diseño de la interfaz de la aplicación Maestro.	
Tipo de tarea : Desarrollo	**Puntos estimados: 1**
Fecha inicio: 03/02/09	**Fecha fin:** 04/02/03
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Creación de una interfaz que contenga las funcionalidades requeridas por el usuario. Se visualizará el listado de direcciones IP y los procesos que se encuentran en ejecución de los equipos conectados, permitirá mostrar el tiempo establecido para el examen.	

Tabla 4.1. 1. Tarea #1 de la Historia de Usuario "Conexión entre la aplicación Maestro y las aplicaciones Esclavos".

	Tarea
Número tarea: 2	**Número historia: 1**
Nombre tarea: Diseño interfaz *"Establecer conexión"*.	
Tipo de tarea : Desarrollo	**Puntos estimados: 1**
Fecha inicio: 03/02/09	**Fecha fin:** 04/02/03
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Creación de una interfaz que le permita al profesor especificar el rango de direcciones IP, con las que se necesita establecer la conexión.	

Tabla 4.1. 2. Tarea #2 de la Historia de Usuario "Conexión entre la aplicación Maestro y las aplicaciones Esclavos".

Tarea	
Número tarea: 3	**Número historia: 1**
Nombre tarea: Establecer conexión entre la aplicación Maestro y Esclavo.	
Tipo de tarea : Desarrollo	**Puntos estimados: 1**
Fecha inicio: 03/02/09	**Fecha fin:** 04/02/03
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: La aplicación Maestro será capaz de conectarse después de ser especificados las direcciones IP donde se encuentra ejecutándose el Esclavo.	

Tabla 4.1. 3. Tarea #3 de la Historia de Usuario "Conexión entre la aplicación Maestro y las aplicaciones Esclavos".

4.2.2. Iteración 2

En esta iteración se realiza la implementación de las historias de usuario con prioridad alta, es este caso relacionadas con el proceso de monitoreo de las actividades que se realizan en las computadoras.

Historias de Usuarios	Estimación	Real
Monitoreo de la actividad en cada una de las computadoras	3	3
Total	3	3

Tabla 4. 2. Historias de Usuario Iteración 2

4.2.2.1. Tareas de las Historias de Usuarios desarrolladas en la segunda iteración.

Historia de usuario: Monitoreo de la actividad en cada una de las computadoras

	Tarea
Número tarea: 1	**Número historia: 5**
Nombre tarea: Implementación de la librería "Monitoreo"	
Tipo de tarea : Desarrollo	**Puntos Estimados : 1**
Fecha inicio:	**Fecha fin:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Esta librería estará implementada en el lenguaje de programación C, y contendrá la funcionalidad encargada de obtener las unidades que se encuentran ocupadas en un momento determinado.	

Tabla 4.2. 1. Tarea #1 de la Historia de Usuario "Monitoreo de la actividad en cada una de las computadoras".

	Tarea
Número tarea: 2	**Número historia: 5**
Nombre tarea: Monitorear dispositivos externos colocados en una computadora determinada	
Tipo de tarea : Desarrollo	**Puntos Estimados : 1**
Fecha inicio:	**Fecha fin:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: La aplicación será capaz de mantenerse monitoreando las diferentes torres que estén ocupadas en las computadoras que se encuentran conectadas, siendo capaz de detectar en caso de que sea colocado algún dispositivo externo en las mismas.	

Tabla 4.2. 2. Tarea #2 de la Historia de Usuario "Monitoreo de la actividad en cada una de las computadoras".

Tarea	
Número tarea: 3	**Número historia: 5**
Nombre tarea: Diseño de la Interfaz “Reportes”	
Tipo de tarea : Desarrollo	**Puntos Estimados : 1**
Fecha inicio:	**Fecha fin:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Diseño de la interfaz que se encargará de mostrar dirección IP, usuario y dispositivo externo colocado en una computadora determinada.	

Tabla 4.2. 3. Tarea #3 de la Historia de Usuario “Monitoreo de la actividad en cada una de las computadoras”.

Tarea	
Número tarea: 4	**Número historia: 5**
Nombre tarea: Mostrar reportes	
Tipo de tarea : Desarrollo	**Puntos Estimados : 1**
Fecha inicio:	**Fecha fin:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: La aplicación debe ser capaz de mostrar, si el usuario lo desea, todos los reportes recibidos desde las aplicaciones Esclavos una vez que estas detectan que un dispositivo externo ha sido colocado en la computadora correspondiente.	

Tabla 4.2. 4. Tarea #4 de la Historia de Usuario “Monitoreo de la actividad en cada una de las computadoras”.

4.2.3. Iteración 3

En la presente iteración se realiza la implementación de las Historias de Usuario de prioridad media, referentes a todo el proceso de distribución y recogida del examen, siendo estas unas de las funcionalidades de mayor importancia en el desarrollo del software.

Historias de Usuarios	Estimación	Real
Distribuir Examen	2	2
Recoger Examen	1	1
Total	3	3

Tabla 4. 3. Historias de Usuario Iteración 3

4.2.3.1. Tareas de las Historias de Usuarios desarrolladas en la tercera iteración.

Historia de usuario: Distribuir Examen

Tarea	
Número tarea: 1	**Número historia: 1**
Nombre tarea: Diseño de la interfaz "*Distribuir Examen*"	
Tipo de tarea : Desarrollo	**Puntos Estimados: 1**
Fecha inicio:	**Fecha inicio:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Creación de una interfaz que le permita al profesor cargar el examen, y a la ves mostrarlo para su posterior distribución.	

Tabla 4.3. 1. Tarea #1 de la Historia de usuario Distribuir Examen.

Tarea	
Número tarea: 2	**Número tarea: 2**
Nombre tarea: Enviar examen a las computadoras conectadas.	
Tipo de tarea : Desarrollo	**Tipo de tarea :** Desarrollo
Fecha inicio:	**Fecha inicio:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Debe ser implementado un mecanismo que después de ser cargado el examen, desde la aplicación Maestro sea enviado un mensaje hacia la aplicación Esclavo, especificándole el envío del mismo, y seguidamente se realice su distribución.	

Tabla 4.3. 2. Tarea #2 de la Historia de Usuario Distribuir Examen.

Historia de usuario: Recoger examen.

Tarea	
Número tarea: 1	**Número historia: 7**
Nombre tarea: Recoger el examen.	
Tipo de tarea : Desarrollo	**Puntos estimados: 1**
Fecha inicio:	**Fecha fin:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Debe ser implementado un mecanismo que permita a través de la aplicación Esclavo obtener el examen ya terminado y luego ser enviado hacia el Maestro.	

Tabla 4.3. 3. Tarea #1 de la Historia de Usuario Recoger Examen.

Tarea	
Número tarea: 2	**Número historia: 7**
Nombre tarea: Diseño de la interfaz "*Recoger Examen*".	
Tipo de tarea : Desarrollo	**Puntos estimados: 1**
Fecha inicio:	**Fecha fin:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Creación de una interfaz que muestre el examen después de ser finalizado, ofreciendo la opción de guardarlo en la dirección deseada.	

Tabla 4.3. 4. Tarea #2 de la Historia de Usuario Recoger Examen.

4.2.4. Iteración 4

En la cuarta iteración se implementan las Historias de Usuarios de menor prioridad, relacionadas con la eliminación de los archivos y el establecimiento del tiempo de duración del examen a desarrollar.

Historias de Usuarios	Estimación	Real
Eliminar archivos	0.5	0.5
Establecer tiempo de duración del examen	0.5	0.5
Total	1	1

Tabla 4. 4. Historias de Usuario Iteración 4

4.2.4.1. Tareas de las Historias de Usuarios desarrolladas en la cuarta iteración.

Historia de usuario: Eliminar archivos

	Tarea
Número tarea: 1	**Número historia: 2**
Nombre tarea: Eliminar archivos del escritorio de las computadoras conectadas.	
Tipo de tarea : Desarrollo	**Puntos estimados: 1**
Fecha inicio:	**Fecha fin:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Debe ser implementado un mecanismo que desde la aplicación Maestro sea enviado un mensaje hacia la aplicación Esclavo, especificándole la eliminación de los archivos existentes en el escritorio, y esta proceda a su realización automática.	

Tabla 4.4. 1. Tarea #1 de la Historia de Usuario Eliminar archivos.

Historia de usuario: Establecer tiempo de duración del examen

	Tarea
Número tarea: 1	**Número historia: 4**
Nombre tarea: Creación de un reloj.	
Tipo de tarea : Desarrollo	**Puntos estimados: 1**
Fecha inicio:	**Fecha fin:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Debe ser implementado un reloj, que dado la duración establecida para el examen, inicie un conteo regresivo hasta finalizar el tiempo especificado.	

Tabla 4.4. 2. Tarea #1 de la Historia de Usuario Establecer tiempo de duración del examen.

Tarea	
Número tarea: 2	**Número historia: 4**
Nombre tarea: Establecer duración del examen.	
Tipo de tarea : Desarrollo	**Puntos estimados: 1**
Fecha inicio:	**Fecha fin:**
Programador responsable: Dolores Rodríguez, Anabel Soria	
Descripción: Debe ser implementada una función donde la aplicación Maestro después de ser especificado el tiempo establecido para el examen, lo debe enviar al Esclavo, quien se encargará de mostrar en pantalla, a partir de la duración especificada, lo que resta para finalizar el examen.	

Tabla 4.4. 3. Tarea #2 de la Historia de Usuario Establecer tiempo de duración del examen.

4.3. Pruebas

Uno de los pilares de la Programación Extrema es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones (32).

4.3.1. Pruebas de Aceptación

Las pruebas de aceptación constituyen el artefacto más importante que genera la fase de pruebas en la metodología XP. Las mismas son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a testear cuando una historia de usuario ha sido correctamente implementada.

Caso de Prueba de Aceptación	
Código: HU #1_P1	**Historia de Usuario 1:** Conexión entre la aplicación Maestro y la aplicación Esclavo
Nombre: Establecer conexión	
Descripción: Prueba para realizar la conexión entre la aplicación Maestro con las aplicaciones Esclavos.	
Condiciones de ejecución: La aplicación Esclavo debe estar ejecutándose inicialmente en dos computadoras como mínimo. La aplicación Maestro debe ser ejecutada en un PC del local donde se encuentran las aplicaciones Esclavos.	
Entrada/Pasos de ejecución:	
Resultado Esperado: Al finalizar la ejecución de la funcionalidad, ambas aplicaciones deben haberse conectadas satisfactoriamente.	

Tabla 4. 5. Pruebas de Aceptación Historia de Usuario: "Conexión entre la aplicación Maestro y la aplicación Esclavo" .

Caso de Prueba de Aceptación	
Código: HU #2_P1	**Historia de Usuario 2:** Eliminar archivos.
Nombre: Eliminar Archivos	
Descripción: Prueba para verificar que una vez ejecutada esta funcionalidad, se eliminan correctamente los archivos del escritorio de cada una de las computadoras	
Condiciones de ejecución: Ambas aplicaciones, maestro y esclavo, deben estar ejecutándose y conectadas entre si.	
Entrada/Pasos de ejecución:	
Resultado Esperado: Una vez culminada la prueba deben haber sido eliminados todos los archivos de cada uno de los escritorios	

Tabla 4. 6. Pruebas de Aceptación Historia de Usuario: "Eliminar archivos".

Caso de Prueba de Aceptación	
Código: HU #3_P1	**Historia de Usuario 3:** Distribuir examen
Nombre: Envío de exámenes	
Descripción: Prueba para verificar la copia correcta y sin fallos de los exámenes en las computadoras	
Condiciones de ejecución: Ambas aplicaciones, maestro y esclavo, deben estar ejecutándose y conectadas entre sí.	
Entrada/Pasos de ejecución: Deben ser especificados en la aplicación maestro los archivos a enviar	
Resultado Esperado: Una vez culminada la prueba deben estar copiados cada uno de los archivos que se enviaron en todas las computadoras	

Tabla 4. 7. Pruebas de Aceptación Historia de Usuario: "Distribuir examen".

Caso de Prueba de Aceptación	
Código: HU #4_P1	**Historia de Usuario 4:** Establecer tiempo de duración del examen
Nombre: Establecer tiempo	
Descripción: Prueba para verificar que se muestre en cada computadora, el tiempo asignado para la realización de los exámenes, y una vez establecido este, que disminuya acorde al tiempo real transcurrido	
Condiciones de ejecución: Ambas aplicaciones, maestro y esclavo, deben estar ejecutándose y conectadas entre sí.	
Entrada/Pasos de ejecución: Debe ser especificado en la aplicación maestro el tiempo destinado para el examen	
Resultado Esperado: Una vez ejecutada la funcionalidad, el tiempo debe ser mostrado y seguidamente comenzar a disminuir	

Tabla 4. 8. Pruebas de Aceptación Historia de Usuario: "Establecer tiempo de duración del examen".

Caso de Prueba de Aceptación	
Código: HU #7_P1	**Historia de Usuario 7:** Recoger exámenes
Nombre: Recoger exámenes	
Descripción: Prueba para verificar que una vez concluido el examen, estos sean enviados a una dirección especificada en la computadora donde esta ejecutándose la aplicación maestro	
Condiciones de ejecución: Ambas aplicaciones, maestro y esclavo, deben estar ejecutándose y conectadas entre sí.	
Entrada/Pasos de ejecución: Debe ser especificado en la aplicación maestro la dirección destino de los archivos y la dirección IP de la computadora de la cual se recogerán los archivos	
Resultado Esperado: Una vez ejecutada la funcionalidad, debe encontrarse en la dirección especificada el archivo correspondiente	

Tabla 4. 9. Pruebas de Aceptación Historia de Usuario: "Recoger exámenes".

4.4. Conclusiones

El principal objetivo de estas dos iteraciones es que al concluir el desarrollo del producto, este sea entregado con la mayor calidad posible, para ello es importante detectar el número máximo de errores posibles durante la etapa de pruebas, así como también es fundamental la interacción con el cliente durante todo el tiempo que dure la fase implementación.

CAPÍTULO V: ESTUDIO DE FACTIBILIDAD

5.1. Introducción

La medición del software está adquiriendo una gran importancia debido a que cada vez se hace más patente la necesidad de obtener datos objetivos que permitan evaluar, predecir y mejorar la calidad del software así como el tiempo y coste de desarrollo del mismo (33). El objetivo de este capítulo es realizar un estudio de factibilidad del sistema propuesto, obteniendo así una idea aproximada del costo y el esfuerzo empleados en el desarrollo del mismo.

5.2. COCOMO II

COCOMO, propuesto y desarrollado por Barry Boehm, es uno de los modelos de estimación de costos mejor documentados y utilizados. El modelo permite determinar el esfuerzo y tiempo que se requiere en un proyecto de software a partir de una medida del tamaño del mismo expresada en el número de líneas de código que se estimen generar para la creación del producto software (34).

Consiste básicamente en la aplicación de ecuaciones matemáticas sobre los Puntos de Función sin ajustar o la cantidad de líneas de código (SLOC, Source Lines of Code) estimados para un proyecto. Estas ecuaciones se encuentran ponderadas por ciertos factores de costo (cost drivers) que influyen en el esfuerzo requerido para el desarrollo del software (33).

Está compuesto por tres modelos que se adaptan tanto a las necesidades de los diferentes sectores descritos, como al tipo y cantidad de información disponible en cada etapa del ciclo de vida de desarrollo, lo que se conoce por granularidad de la información (34).

El estudio de factibilidad de este sistema se realizo mediante el modelo de diseño temprano que se utiliza en las primeras etapas del desarrollo en las cuales se evalúan las alternativas de hardware y software de un proyecto. En estas etapas se tiene poca información, se conoce muy poco del tamaño del producto a ser desarrollado, de la naturaleza de la plataforma, del personal a ser incorporado al proyecto o aspectos específicos del proceso a utilizar. Este nivel de detalle en este modelo es consistente con el nivel

general de información disponible y con el nivel general de estimación detallada que es necesaria en esta etapa, lo que concuerda con el uso de Puntos de Función, para estimar tamaño usa Puntos de Función No Ajustados como métrica de medida y el uso de un número reducido de factores de costo (34).

5.3. Características del proyecto

El primer paso a llevar a cabo para la estimación del proyecto consiste en la obtención de los Puntos de Función desajustados, los cuales están dados por la suma de cada una de las entradas, las salidas y las consultas externas del sistema, así como los archivos lógicos internos y de interfaz externos. A continuación se muestran cada una de estas características aplicadas al software en cuestión.

5.3.1. Entradas externas

Se definen como un proceso elemental mediante el cual ciertos datos cruzan la frontera del sistema desde afuera hacia adentro (33). En el caso particular de este software se cuenta con 5 entradas externas, especificadas en la siguiente tabla:

Nombre de la entrada externa	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Establecer conexión	0	2	Simple
Establecer tiempo de duración del examen	0	1	Simple
Distribuir examen	1	2	Simple
Recoger examen	1	2	Simple
Validar procesos	0	6	Simple
Total		**13**	

Tabla 5. 1. Entradas Externas

5.3.2. Salidas externas

Se definen como un proceso elemental con componentes de entrada y de salida mediante el cual datos simples y datos derivados (esto es, datos que se calculan a partir de otros datos) cruzan la frontera del sistema desde adentro hacia afuera) (33). Las salidas externas vinculadas al proyecto se describen a continuación.

Nombre de la entrada externa	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Tiempo de duración del examen	0	1	Simple
Procesos detectados durante el monitoreo	0	1	Simple
Examen	1	1	Simple
Total		**3**	

Tabla 5. 2. Salidas Externas

5.3.3. Consultas Externas

Se definen como un proceso elemental con componentes de entrada y de salida donde un Actor del sistema rescata datos de uno o más Archivos Lógicos Internos o Archivos de Interfaz Externos. Los datos de entrada no actualizan ni mantienen ningún archivo (lógico interno o de interfaz externo) y los datos de salida no contienen datos derivados (es decir, los datos de salida son básicamente los mismos que se obtienen de los archivos) (33).

Nombre de la petición	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Total		**0**	

Tabla 5. 3. Consultas Externas

5.3.4. Archivos Lógicos Internos

Constituyen un grupo de datos relacionados lógicamente e identificables por el usuario, que residen enteramente dentro de los límites del sistema y se mantienen a través de entradas externas (33). En este caso no se cuentan con archivos lógicos internos, como se muestra en la siguiente tabla.

Nombre del fichero interno	Cantidad de records	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Total		**0**	

Tabla 5. 4. Archivos Lógicos Internos

5.3.5. Archivos de Interfaz Externos

Son un grupo de datos relacionados lógicamente e identificables por el usuario, que se utilizan solamente para fines de referencia. Los datos residen enteramente fuera de los límites del sistema y se mantienen por las Entradas Externas de otras aplicaciones, es decir, cada Archivo de Interfaz Externo es un Archivo Lógico Interno de otra aplicación (33). En este caso fueron identificados dos Archivos de interfaz externos, como se detalla en la siguiente tabla:

Nombre de la entrada externa	**Cantidad de records**	**Cantidad de elementos de datos**	**Clasificación (simple, media y compleja)**
Examen finalizado	1	1	Simple
Procesos a controlar	0	0	Simple
Total		**1**	

Tabla 5. 5. Archivos de Interfaz Externos

5.3.6. Puntos de función desajustados

La siguiente tabla está basada en las características del sistema anteriormente expuestas, el producto de la cantidad existente de cada una de ellas y el peso correspondiente a las mismas dan como resultado final, los puntos de función desajustados pertenecientes al proyecto.

Elementos	Simple		Medio		Complejo		Subtotal
	No.	Peso	No.	Peso	No.	Peso	
Entrada Externa	5	3	0	4	0	6	15
Salida Externa	3	4	0	5	0	7	12
Petición	0	3	0	4	0	6	0
Fichero Lógico Interno	0	7	0	10	0	15	0
Fichero interfaz Externa	2	5	0	7	0	15	10
Subtotal (UFP)							**37**

Tabla 5. 6. Puntos de Función desajustados

5.4. Cálculo de instrucciones fuentes, esfuerzo, tiempo de desarrollo, cantidad de hombres y costo.

5.4.1. Cálculo de instrucciones fuentes.

Luego de obtener la cantidad total de puntos de función desajustados pertenecientes al proyecto, se procede a calcular la cantidad de instrucciones fuentes del mismo, para lo cual se utiliza la siguiente ecuación:

$$SLOC = UFP \times ratio$$

Donde

SLOC: Cantidad de instrucciones fuentes.

UFP: Puntos de función desajustados.

ratio: Conversión de puntos de función desajustados a líneas de código para el lenguaje java.

Partiendo de la ecuación anterior se obtiene el siguiente resultado:

$$SLOC = 37 \times 53$$

$$SLOC = 1961 = 1.91\ KSLOC$$

Las características y valores obtenidos anteriormente han sido plasmados en la siguiente tabla:

Características	Valor
Puntos de función desajustados	37
Lenguaje (Java)	53
Instrucciones fuentes por puntos de función	1961 SLOC
Instrucciones fuentes	1.91 SLOC

Tabla 5. 7. Características generales del proyecto

5.4.2. Cálculo del esfuerzo nominal

Posteriormente se procede al cálculo del esfuerzo nominal, ecuación que se toma como base tanto en el método de diseño preliminar, al cual se hacía alusión anteriormente, como en el modelo Post arquitectura, ambos definidos por COCOMO II.

$$PM_{nominal} = A \times (Size)^{B} \textbf{ (2)}$$

$$PM_{nominal} = 2.94 \times 1.91^{1.09}$$

$$PM_{nominal} = 5.95\ meses/hombre$$

Donde

$PM_{nominal}$: Esfuerzo nominal requerido en meses – hombre

size: Tamaño estimado del software, en Puntos de Función sin ajustar (KSLOC)

A: constante que se utiliza para capturar los efectos multiplicativos en el esfuerzo requerido de acuerdo al crecimiento del tamaño del software. (Valor: 2.94).

B: Constante denominada Factor escalar y su valor está dado por la resultante de los aspectos positivos sobre los negativos que presenta el proyecto (33).

$$B = 0.91 + 0.01 \times \sum(W_i) \textbf{ (3)}$$

$$B = 0.91 + 0.01 \times 17.78$$

$$B = 1.09$$

Donde

W_i**:** Variables escalares que indican las características que el proyecto presenta en lo que a su complejidad y entorno de desarrollo se refiere. La siguiente tabla muestra los valores asignados a cada una de estas variables:

Nombre	Valor	Justificación
PREC	6.2	Actualmente no existen sistemas que cumplan con las funcionalidades del software a desarrollar.
FLEX	2.03	Cuenta con una alta flexibilidad con los requerimientos pre – establecidos.
TEAM	2.19	El equipo de desarrollo presenta una alta cohesión.
RESL	4.24	Se identificaron algunos de los riesgos críticos.
PMAT	3.12	Se cuenta con la experiencia necesaria como para que el software cumpla con las funcionalidades requeridas
Total (SF)	**17.78**	

Tabla 5. 8. Factores de Escala

5.4.2.1. Ajuste del esfuerzo nominal

El esfuerzo calculado anteriormente es un valor nominal y debe ser ajustado en base a las características del proyecto para lo cual se tiene un conjunto de Multiplicadores de esfuerzo (ME_i) que representan las características del proyecto y expresan su impacto en el desarrollo total del producto de software (33).

Nombre	Valor	Justificación
RCPX	1.74	La complejidad del sistema es alta
RUSE	1.00	Pretende reutilizase solo una parte del código
PDIF	1.00	Uso de memoria y almacenamiento normal, plataforma estable
PREX	1.12	Baja experiencia del personal en la utilización de lenguaje, así como de la plataforma
PERS	1.00	La capacidad del personal es alta
FCIL	1.00	La utilización de entornos de desarrollo integrados, así como de herramientas CASE simples facilitan en gran medida el trabajo.
SCED	1.00	El sistema se desarrolló en el tiempo establecido
Total (EM)	**1.95**	

Tabla 5. 9. Multiplicadores de esfuerzo

$$PM_{ajustado} = PM_{nominal} \times \prod(ME_i)$$

$$PM_{ajustado} = 5.95\ meses/hombre \times 1.95$$

$$PM_{ajustado} = 11.6\ hombres/mes$$

5.4.3. Cálculo del tiempo de desarrollo del software

El tiempo requerido para el desarrollo del proyecto está dado por la siguiente ecuación:

$$TDEV = C \times (PM_{ajustado})^F$$

$$TDEV = 3.67 \times 11.6^{0.28}$$

$$TDEV = 7.29$$

Donde

C: constante con valor 3.64

$$F = D + 0.2 \times 0.01 \times \sum SF$$

$$F = 0.24 + 0.2 \times 0.01 \times 17.78$$

$$F = 0.28$$

Donde

D: constante cuyo valor es 0.24

SF: valor de los factores de escala

5.4.4. Cálculo del costo total del proyecto.

Para el cálculo del costo total correspondiente al proyecto en cuestión, COCOMO II define la siguiente ecuación:

$$C = CHM \times PM$$

$$C = 200 \times 11.6$$

$$C = 2320$$

Donde

C: costo total

CHM: costo teniendo en cuenta salario de todos los obreros, el cual es calculado por la ecuación

$$CHM = CH \times sal$$

$$CHM = 2 \times 100$$

$$CHM = 200$$

Donde

Sal: Salario medio por cada trabajador

CH : Cantidad de personas destinadas al proyecto, lo que es calculado a través de la ecuación:

$$CH = {}^{PM}/_{TDEV}$$

$$CH = {}^{11.6}/_{7.29}$$

$$CH \approx 2$$

5.5. Resultados

La siguiente tabla muestra los resultados obtenidos luego de haber realizado todo el análisis correspondiente y haber efectuado todos los cálculos para determinar el costo y esfuerzo requeridos por el proyecto.

Calculo de:	Valor
Esfuerzo	11.6 hombres/mes
Tiempo de desarrollo	7 meses
Cantidad de hombres	2 hombres
Salario medio	100 pesos
Costo	2320 pesos

Tabla 5. 10. Resultados

5.6. Conclusiones

Entre los objetivos fundamentales de COCOMO II se encuentra proporcionar un marco analítico cuantitativo y un conjunto de herramientas y técnicas para la evaluación de los efectos de la mejora tecnológica del software en costes y tiempo del ciclo de vida del mismo. Aplicando este modelo se pudo obtener un aproximado del coste total del proyecto, dato que toma el valor de 2320 pesos, siendo necesarios 2 hombres para el desarrollo del mismo.

CONCLUSIONES

- A partir de la investigación y estudio del funcionamiento del proceso de comunicación a través de Socket TCP/IP, se logró implementar una herramienta capaz de monitorear y controlar los exámenes realizados en laboratorios docentes.
- Se obtuvo un software Maestro que permite dirigir y controlar el proceso de aplicación de exámenes en cada uno de los laboratorios en los que se estén efectuando los mismos.
- Se obtuvo un software Esclavo encargado de mantener el control de la computadora correspondiente y brindar información referente a los resultados obtenidos durante la ejecución de cada una de las peticiones enviadas por el software Maestro.
- Se obtuvo un sistema, primero de su tipo en Cuba, con el que se logra, fundamentalmente, agilizar el proceso de aplicación de exámenes, además de tener, en gran medida, un control sobre la actividad realizada por los estudiantes en el momento en que se desarrollan los mismos.

RECOMENDACIONES

- Actualmente la aplicación, durante el monitoreo, solo permite llevar el control de los dispositivos de almacenamiento externo que se colocan en una computadora determinada, por lo que se recomienda continuar la implementación de esta funcionalidad con el objetivo de lograr que el software pueda detectar todas las aplicaciones que se encuentran abiertas en cada uno de los escritorios.
- En la UCI, una de las principales tendencias es la migración hacia el software libre, razón por la cual se recomienda añadir las funcionalidades necesarias para que el sistema pueda ser utilizado en el sistema operativo Linux.

REFERENCIAS BIBLIOGRAFICAS

1. masadelante.com. [Online] [Cited: Enero 16, 2009.] http://www.masadelante.com/faqs/socket.

2. Lenguajes y Ciencias de la Computación. [Online] 2008. [Cited: Enero 16, 2009.] http://www.lcc.uma.es/~eat/services/i_socket/i_socket.html#link2.

3. **Barranco, Miguel Rueda.** TLDP-ES/LuCAS. *Doctorado en informatica Sockets: Comunicacion entre procesos distribuidos.* [Online] Junio 1996. [Cited: Enero 16, 2009.] http://es.tldp.org/Universitarios/seminario-2-sockets.html.

4. **Sun.** Programación en castellano. *Java en castellano.* [Online] [Cited: Enero 16, 2009.] http://programacion.com/java/tutorial/red/11/.

5. Ciencia e Ingeniería de la Computación. *Programación en redes.* [Online] Septiembre 20, 1998. [Cited: Enero 16, 2009.] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-1/resumen2.html#TCP.

6. Departamento de Arquitectura y Tecnologías de Computadores. Universidad de Granada. *Comunicación de Socket. Tipo SOCKET_STREAM.* [Online] [Cited: Enero 20, 2009.] http://atc.ugr.es/~hcamacho/socket.pdf.

7. Lenguajes de Programación. *Lenguajes de Programación.* [Online] 2009. [Cited: Enero 17, 2009.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

8. Nexos Software. *Un Enfoque al Leguaje de Programación Java .* [Online] [Cited: Enero 17, 2009.] http://www.nexos-software.com.co/Articulo_25.htm.

9. **Rocha, Jorge Luis Aréchiga.** arechiga.50megs.com. *Java.* [Online] [Cited: Enero 17, 2009.] http://arechiga.50megs.com/tpoo2/2dept/informacionjava2.html.

10. —. arechiga.50megs.com. *Java.* [Online] [Cited: Enero 17, 2009.] http://arechiga.50megs.com/tpoo2/javah1.html#Ventajas.

11. **Marañón, Gonzalo Álvarez.** Departamento de Tratamiento de la Información y Codificación. [Online] 2009. [Cited: Enero 17, 2009.] http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.

12. **Rocha, Jorge Luis Aréchiga.** arechiga.50megs.com. [Online] [Cited: Enero 17, 2009.] http://arechiga.50megs.com/tpoo2/javah1.html#Maquina%20Virtual%20de%20Java.

13. **Morales, Edgar Iván Durán.** Modulo de conexión para el CU Communicator vía WEB. Apéndice B. [Online] enero 14, 2003. [Cited: Enero 19, 2009.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/duran_m_ei/apendiceB.pdf.

14. wapedia. [Online] [Cited: Enero 19, 2009.] http://wapedia.mobi/es/Java_Native_Interface.

15. Lenguajes de Programación. *Lenguajes de Programación.* [Online] 2009. [Cited: Enero 17, 2009.] http://www.lenguajes-de-programacion.com/programacion-en-c.shtml.

16. **Vergara, Kervin.** Blog Informático. [Online] Julio 06, 2007. [Cited: Enero 17, 2009.] http://www.bloginformatico.com/lenguaje-de-programacion-c.php.

17. msdn . [Online] Noviembre 2007. [Cited: Enero 18, 2009.] http://msdn.microsoft.com/es-es/library/dtba4t8b.aspx.

18. DLLsDnld. [Online] Septiembre 14, 2007. [Cited: Enero 19, 2009.] http://es.dll-download-system.com/.

19. **García de Jalón, Javier, et al.** *Aprenda Java como si estuviera en primero.* Ciudad de la Habana : s.n., 2000.

20. NetBeans. [Online] [Cited: Enero 19, 2009.] http://www.netbeans.org/index_es.html.

21. Sun Microsystems. [Online] [Cited: Enero 19, 2009.] http://www.sun.com/emrkt/innercircle/newsletter/latam/0207latam_feature.html.

22. SlideShare. *SlideShare.* [Online] [Cited: Enero 20, 2009.] http://www.slideshare.net/juliopari/4-clase-metodologia-de-desarrolo-de-software.

23. Metodologia XP Vs. Metodologia Rup . *Metodologia XP Vs. Metodologia Rup .* [Online] [Cited: Enero 20, 2009.] http://metodologiaxpvsmetodologiarup.blogspot.com/.

24. www.concytec.gob.pe. *Ministerio de Educación. Consejo Nacional de Ciencias, Tecnologías e Innovación Tecnológica.* [Online] [Cited: Enero 20, 2009.] http://www.concytec.gob.pe/clubciencias/index.php?option=com_content&task=view&id=677&Itemid=375.

25. **Joskowicz, Ing. José.** *Reglas y Prácticas en eXtreme Programming.* 2005.

26. **Letelier, Patricio and Penadés, Mª Carmen.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).*

27. **Plá Rodríguez, José Antonio and Ilizastegui Arriba, Damián.** *Tesis: "Sistema para la integración continua de proyectos y el control de builds en la empresa Procyon Soluciones".* Ciudad de la Habana : s.n., 2007.

28. **Wilson, Andy.** *User interface software technology.* s.l. : ACM Computing surveys, 1996.

29. **Shneiderman, Ben.** *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Menlo Park, CA : Addison Wesley., 1998.

30. **Castillo, Oswaldo, Figueroa, Daniel and Sevilla, Hector.** Programación Extrema. [Online] [Cited: Abril 22, 2009.] http://programacionextrema.tripod.com/fases.htm.

31. **Larman, Craig.** *UML y Patrones.* La Habana : Félix Varela, 2004.

32. **Rodríguez Corbea, Maite and Ordóñez Pérez, Meylin.** *Tesis: "La metodología XP aplicable al desarrollo del software educativo en Cuba:.* Ciudad de la Habana : s.n., 2007.

33. unab. Universidad Autónoma de Bucaramanga. *Obtención y Validación de Modelos de Estimación deSoftware Mediante Técnicas de Minería de Datos.* [Online] [Cited: Marzo 3, 2009.] http://www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r31_art3_c.pdf.

34. **Rodríguez, Raymari Reyes Chirino y Ariel Díaz.** *Evaluación de posibles elementos a considerar en la integración de los métodos de Boehm y Humphrey para la estimación de la duración de un proyecto de software para aplicaciones Multimedia.* Ciudad de La Habana : s.n., 2008.

GLOSARIO DE TERMINOS

A

API (Interfaz de Programación de Aplicaciones): Lenguaje y formato de mensaje utilizados por un programa para activar e interactuar con las funciones de otro programa o de un equipo físico.

Aplicación distribuida: aplicación con distintos componentes que se ejecutan en entornos separados, normalmente en diferentes plataformas conectadas a través de una red.

C

C: Lenguaje de programación estructurado, de aplicación general.

C++: Lenguaje C orientado a objetos.

Código nativo: Seudónimo de lenguaje de máquina, este puede ser creado directamente para micro controladores extremadamente sencillos o código fuente ya compilado, que puede ser interpretado por la maquina.

D

Dirección IP: Número único e irrepetible con el cual se identifica una computadora conectada a una red que corre el protocolo IP.

Dispositivo de almacenamiento externo: Dispositivos periféricos del sistema, que actúan como medio de soporte para la grabación de los programas de usuario, y de los datos y ficheros que son manejados por las aplicaciones que se ejecutan en estos sistemas.

F

Filosofía CLIENTE-SERVIDOR: Arquitectura de sistemas de información en la que los procesos de una aplicación se dividen en componentes que se pueden ejecutar en máquinas diferentes. Modo de funcionamiento de una aplicación en la que se diferencian dos tipos de procesos y su soporte se asigna a plataformas diferentes.

H

Hilo de ejecución: Característica que permite a una aplicación realizar varias tareas concurrentemente.

J

Java: Lenguaje desarrollado por Sun Microsystems para la elaboración de aplicaciones exportables a la red y capaces de operar sobre cualquier plataforma.

JNI (*Java Native Interface*): Framework de programación que permite que un programa escrito en Java ejecutado en la máquina virtual java (JVM) pueda interactuar con programas escritos en otros lenguajes como C, C++ y ensamblador.

L

Librerías. Conjunto de programas que cumplen una funcionalidad determinada y que se ponen a disposición de los programadores para simplificar tareas complejas.

M

Metodología Ágil: Marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto que intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados.

Metodología de desarrollo: Conjunto de normas y principios a aplicar en el proceso de construcción de un software.

Multiplataforma: término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

MySQL: Sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

P

PHP (Pre-procesador de Hipertexto): Lenguaje de programación del lado del servidor usado principalmente para la creación de contenido para sitios web.

Puerto: Número entero pequeño usado para identificar un programa de aplicación en una computadora remota. Los protocolos de transportación, como el TCP, asignan un número de puerto único a cada servicio (por ejemplo, el correo electrónico usa el puerto 25).

R

RUP (Proceso Unificado de Software): Marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental.

S

Servidor: Ordenador que ofrece sus prestaciones a varios ordenadores clientes conectados a una red.

Sistema gestor de Base de Datos: Tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Sitio Web: Es un sitio (localización) que contiene documentos (páginas web) organizados jerárquicamente. Cada documento contiene texto y o gráficos que aparecen como información digital en la pantalla de un ordenador. Un sitio puede contener una combinación de gráficos, texto, audio, vídeo, y otros materiales dinámicos o estáticos.

Socket: Número de identificación compuesto por dos números: La dirección IP y el número de puerto TCP. En la misma red, el nº IP es el mismo, mientras el nº de puerto es el que varia.

Software: Programa del sistema, de aplicación, de utilidades, procedimientos, reglas y su documentación asociada, relacionados con la operación de un ordenador. Conjunto de instrucciones y datos que un ordenador es capaz de entender.

Software libre: Es la denominación del software que brinda libertad a los usuarios sobre un producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

Software esclavo: Es el encargado de procesar los pedidos del software maestro y luego enviar la respuesta, permite chequear las actividades que se realizan en la PC y ejecutar comandos.

Software maestro: Es un software diseñado para usarlo en la red, desde una PC determinada, el cual estará en constante comunicación con el software esclavo, es el encargado de realizar solicitudes, funcionando a la vez como un controlador.

T

TCP/IP ("Protocolo de control de transmisión/Protocolo de Internet"): Conjunto de protocolos de red en la que se basa Internet y que permiten la transmisión de datos entre redes de computadoras.

Tecnologías de la Información y las Comunicaciones (TIC): Conjunto de tecnologías que permiten la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones contenidas en señales de naturaleza acústica (sonidos), óptica (imágenes) o electromagnética (datos alfanuméricos).