

Índice general

1. Descripción y Planificación del Proyecto	1
1.1. Descripción Funcional del Sistema	1
1.2. Metodología de Desarrollo	3
1.3. Requisitos de Alto Nivel del Sistema	5
1.4. Iteraciones	6
2. Descripción de una iteración	9
2.1. Inicio de la prueba	10
2.2. Examen temporizado	10
2.3. Denegar acceso a red	11
2.4. Pruebas	12

Índice de figuras

1.1. Diferencia entre iteración e incremento.	4
1.2. Carga de trabajo en las diferentes fases.	6
1.3. Diagrama de casos de uso del sistema	6
2.1. Aspecto de la GUI del alumno con una prueba temporizada. .	11

Índice de cuadros

1.1. Requisitos del sistema	7
2.1. Objetivos de la iteración	10
2.2. Requisitos ya implementados del sistema	10

Capítulo 1

Descripción y Planificación

El presente capítulo describe en líneas generales el ámbito funcional del proyecto, para delimitar su alcance, la metodología a usar durante el desarrollo del sistema y los requisitos de alto nivel del mismo, extraídos del ámbito funcional descrito.

«a completar»

Contents

1.1. Descripción Funcional del Sistema	1
1.2. Metodología de Desarrollo	3
1.3. Requisitos de Alto Nivel del Sistema	5
1.4. Iteraciones	6

1.1. Descripción Funcional del Sistema

La presente sección describe el ámbito funcional del sistema software que deseamos construir, es decir, del *DrManhattan*, que es un *Sistema de Control de Accesos a Red para los Laboratorios de la Facultad de Ciencias*.

El objetivo del *DrManhattan* es controlar el acceso a la red local, existente en cada laboratorio de la Facultad de Ciencias de la Universidad de Cantabria, de los computadores conectados a ella. Se desea realizar dicho control sobre todo durante la realización de pruebas evaluables, de cara a evitar que se realicen accesos a contenidos no autorizados (e.g., páginas de internet con posibles soluciones a los problemas planteados, el directorio de trabajo de algún compañero, etc.) durante la realización de dichas pruebas. El sistema que se venía utilizando hasta ahora para evitar el mal uso de la red local consistía simplemente en desconectar la alimentación del concentrador de interconexión, deshabilitando la red. Es decir, se aplicaba el principio de muerto el perro, se acabó la rabia.

No obstante, el tener los diversos computadores interconectados mediante una red local, no sólo tiene inconvenientes, sino que también posee varias ventajas. Por ejemplo, ayuda a facilitar la distribución de material electrónico que resulte necesario para la realización del examen. También puede ser de gran utilidad para recoger los ejercicios realizados por los alumnos de una manera rápida y cómoda, ya que el alumno sólo tendría que enviar el material producido durante la prueba al computador o dirección que le indique el profesor. El método utilizado actualmente para realizar esta tarea consiste en que el profesor acude al computador del alumno cuando este desea entregar los ejercicios realizados y copia tales ejercicios en una memoria USB. Este proceso es lento, tedioso y en muchas ocasiones las memorias USB no son reconocidas por los computadores del laboratorio, lo que exige recurrir a otras técnicas. El tiempo empleado por los docentes para recopilar los ejercicios realizados por los alumnas suele oscilar entre la media hora y la hora completa, mientras que usando la red local dicho proceso podría realizarse en cuestión de minutos (cuando no menos).

Por tanto, el acceso de los computadores a la red local (e internet), se deberá controlar desde un computador distinguido, al que llamaremos *Watchman*, que será normalmente el computador asignado al docente. Dicho computador será el encargado de conceder y denegar el acceso a la red al resto de computadores conectados a la red local.

Se desea que el acceso a la red esté habilitado en las siguientes circunstancias:

1. Al comienzo de la realización de las pruebas, de forma que el docente pueda distribuir material electrónico necesario o de interés para la realización de la prueba (e.g., manuales, el enunciado de la prueba, etc.) entre los diferentes computadores de forma cómoda y eficiente.
2. Cuando el alumno haya finalizado la prueba, de forma que se puedan enviar los resultados a través de la red, evitando el penoso proceso de tener que recogerlos de forma individual mediante copia en un memoria USB o dispositivo similar. En estos casos, sería además deseable, con objeto de evitar posteriores problemas, que el sistema comprobase la integridad de los archivos recibidos, es decir, que comprobase que no se han sufrido alteración alguna durante la transmisión.

Durante el periodo de tiempo que un alumno esté realizando una prueba, se debe denegar el acceso a la red del computador que esté utilizando para la realización de la prueba.

Por tanto, la secuencia de realización de una prueba evaluable sería tal como sigue:

1. El profesor y los alumnos acceden al aula y encienden sus correspondientes computadores. El acceso a la red está habilitado para todo el mundo.
2. El profesor envía el material necesario para la realización de la prueba a los computadores de los alumnos.
3. A continuación, una vez que un alumno comienza la prueba, se le deniega el acceso a la red al computador que esté utilizando.
4. Si durante la realización de la prueba un alumno considera que ha acabado y está satisfecho con los resultados producidos, indicará al sistema que ha concluido la prueba. Los ficheros producidos como material evaluable se enviarán al *Watchman*. Se comprobará que se han recibido correctamente y no están corruptos, por lo que se pueden abrir y leer sin problema alguno. Obviamente, para el envío de los ficheros de resultados habrá que habilitar de nuevo la red, pero se ha de evitar que el alumno pueda modificar dichos ficheros.
5. Debe existir también la posibilidad de que el docente decida que el tiempo de realización de la prueba ha concluido, por lo que deberá realizarse todo el proceso descrito en el punto anterior, pero para todos los computadores que se encuentren activos en ese momento y siendo el proceso iniciado desde el *Watchman*, en lugar de desde el propio computador del alumno.

Además, se guardarán registros de los eventos ocurridos en cada prueba, con el objetivo de tener información que permita auditar el transcurso de la misma. Dicho registro debe mantener constancia de eventos tales como hora de inicio de la prueba, hora de entrega de cada ejercicio, número de ficheros que se entregan, etc.

Para comodidad del alumno, en las pruebas con hora de finalización pre-determinada se mostrará en la interfaz de la aplicación los minutos restantes hasta el final.

La siguiente sección detalla la metodología de desarrollo que usaremos para la construcción de este sistema.

1.2. Metodología de Desarrollo

«Describe brevemente la metodología en plan abstracto»

En esta sección se describe la metodología utilizada para desarrollar el sistema.

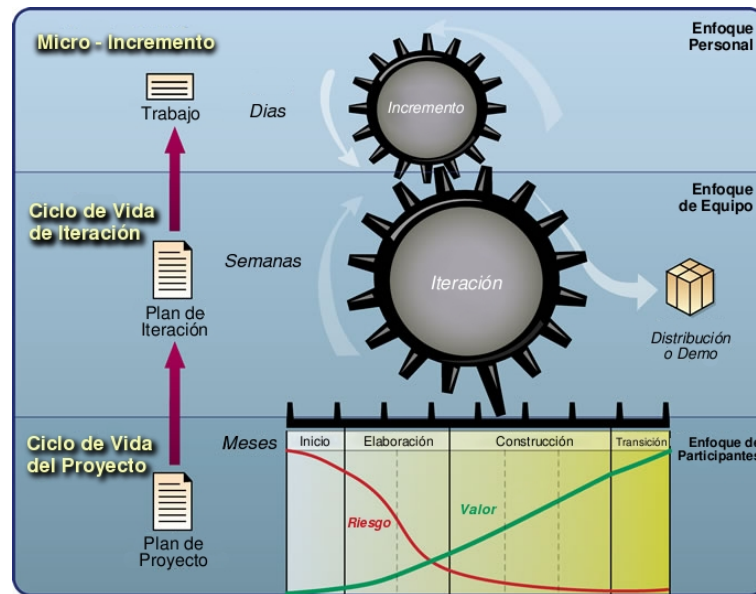


Figura 1.1: Diferencia entre iteración e incremento.

Se ha escogido el proceso de desarrollo *iterativo e incremental* que suele ser parte esencial en las metodologías de desarrollo ágiles.

La idea principal de este proceso se basa, como su nombre, en iteraciones e incrementos, que no son lo mismo. El software se construye mediante tareas repetitivas, *iteraciones*, en las que se añaden nuevas funcionalidades progresivamente, *incrementos*, para crear una versión del sistema que cumple más requisitos que la anterior. Se realizan tantas iteraciones sean necesarias hasta que todos los requisitos se hayan implementado y, por tanto, el sistema este finalizado.

Es decir, una iteración es un mini-proyecto en el que se obtiene una versión de cada una de las piezas del sistema, sean código o no, y un incremento se puede medir como la diferencia entre una iteración y la anterior.

El proceso se divide en cuatro partes fundamentales:

1. **Iniciación:** en esta fase se describe el ámbito funcional del sistema, los requisitos de alto nivel y se identifican posibles riesgos. Hay que comprender el sistema y sus límites.
2. **Elaboración:** el objetivo principal de esta fase es el de crear la arquitectura básica, para tener una visión de cómo será el sistema completo y además, proponer soluciones a los riesgos identificados.
3. **Construcción:** cómo su nombre indica, esta es la fase dónde se cons-

truye y prueba la aplicación. Se realiza un pequeño proceso en cascada en cada iteración de esta fase.

4. **Transición:** se despliega el sistema y finaliza el proceso.

Cada una de estas fases, como es lógico, puede tener más de una iteración, en función del tamaño del proyecto, con tareas propias de la etapa en que se esté, así por ejemplo, en la etapa de construcción, en cada iteración se escoge un grupo de requisitos de alto nivel, se refinan, y partiendo de la versión del sistema obtenida en la iteración anterior, se diseñan, implementan y prueban esos requisitos, creando la versión que se utilizará en la siguiente iteración.

Al finalizar el proceso obtenemos tanto el código de la aplicación, como modelos y documentación de la misma que se van creando a lo largo de las iteraciones.

«añadir imágenes del ciclo de vida»

Las principales razones por las que se ha escogido este tipo de proceso y no otro son:

- ◆ Poca incertidumbre a la hora de diseñar y planificar, ya que al realizarse en cada iteración y no del sistema completo, se tiene mayor comprensión de los riesgos y de las tareas a realizar.
- ◆ Se adapta bien a cambios de requisitos.
- ◆ Si se priorizan los requisitos para implementarlos en las primeras iteraciones de la construcción, se obtiene una aplicación con funcionalidades clave en la que el cliente puede decidir si está o no correcto.
- ◆ Al trabajar con un subconjunto de requisitos en cada iteración la complejidad se reduce, lo que facilita que se reduzca también el número de errores producidos.

De la primera fase, *iniciación*, la descripción funcional detallando el alcance del sistema está descrita en la sección 1.1, a continuación se muestra un diagrama de casos de uso, basado en esa descripción y en la siguiente sección se describen los requisitos de alto nivel extraídos.

1.3. Requisitos de Alto Nivel del Sistema

Esta sección describe el segundo paso en nuestro proceso de desarrollo, de acuerdo a la metodología descrita en la sección anterior, que es la identificación de los requisitos de alto nivel que ha de satisfacer nuestro sistema software, de acuerdo a la descripción del ámbito funcional proporcionada en

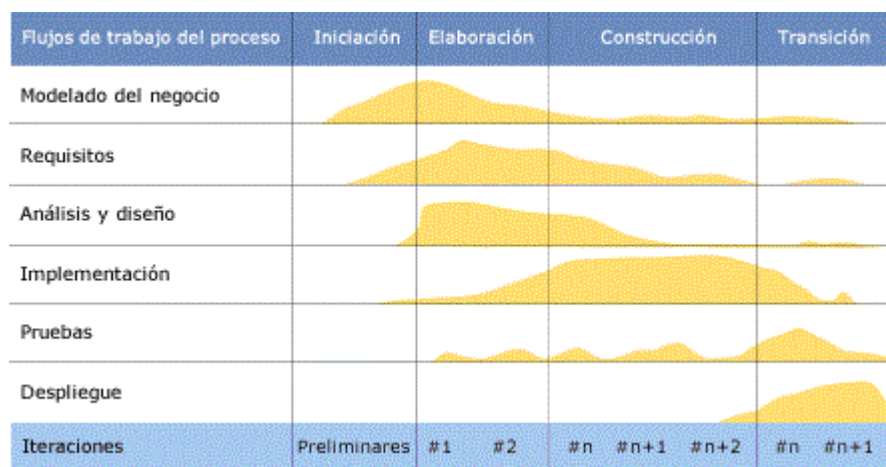


Figura 1.2: Carga de trabajo en las diferentes fases.

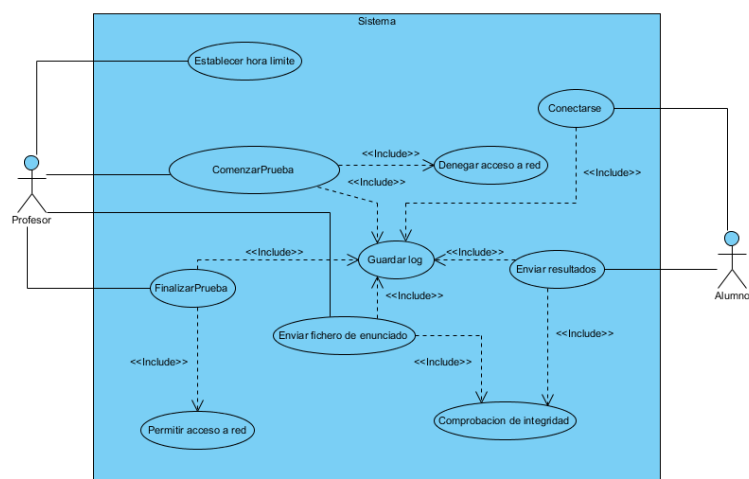


Figura 1.3: Diagrama de casos de uso del sistema

la Sección 1.1.

En concreto, se han identificado los siguientes requisitos de alto nivel para nuestro sistema

«arreglar la tabla»

1.4. Iteraciones

En esta sección se describen las iteraciones previstas para el desarrollo de la aplicación.

Identificador	Descripción
R01	Un computador de la red debe poder ser designado como <i>Watchman</i> .
R02	Todos los computadores que no sean <i>Watchman</i> serán computadores normales, y podrá
R03	El profesor debe ser capaz desde el <i>Watchman</i> de indicar el inicio de la prueba.
R04	El profesor debe ser capaz desde el <i>Watchman</i> de indicar el fin de la prueba.
R05	El profesor ha de ser capaz desde el <i>Watchman</i> de establecer una hora límite para la du
R06	El profesor debe ser capaz desde el <i>Watchman</i> de enviar el fichero de enunciado al resto
R07	El alumno desde su computador debe ser capaz de conectarse al <i>Watchman</i> .
R08	El alumno desde su computador debe ser capaz de enviar sus resultados al profesor.
R09	El alumno desde su computador debe ser capaz de indicar que da por finalizada la prue
R10	El alumno desde su computador debe tener la posibilidad de ver el tiempo restante el p
R11	La aplicación del alumno ha de ser capaz de denegar el acceso a la red al empezar la pr
R12	La aplicación del alumno ha de ser capaz de permitir el acceso a la red al finalizar la pr
R13	La aplicación ha de ser capaz de comprobar que los archivos se han enviado correctame
R14	La aplicación del profesor ha de ser capaz de guardar registros de actividad.

Cuadro 1.1: Requisitos del sistema

■ **Iteración 1: Crear un sistema distribuido.**

- Requisitos implementados: R06, R07
- Descripción: En la primera iteración la aplicación del alumno se podrá conectar al watchman para esperar el inicio de la prueba, envío de ficheros etc... y el profesor es capaz de enviar el enunciado.

■ **Iteración 2: Denegar acceso a red una vez comenzado el examen.**

- Requisitos implementados: R03, R05, R10, R11
- Descripción: En la segunda iteración el profesor puede establecer una hora límite e indicar el inicio de la prueba. La aplicación del alumno es capaz de denegar el acceso a la red cuando se inicia la prueba y en caso de que la duración este prefijada, consultar el tiempo restante

■ **Iteración 3: Habilitar red y enviar resultados**

- Requisitos implementados: R04, R08, R09, R12
- Descripción: En la tercera iteración se permite al profesor finalizar la prueba de modo general. Del mismo modo, cada alumno puede finalizar la prueba por su cuenta y enviar su archivo de resultados al computador del profesor. La aplicación del alumno, cuando este finaliza la prueba, habilita el acceso a la red.

■ Iteración 4: Comprobación de integridad

- Requisitos implementados: R13
- Descripción: En la cuarta iteración se implementa el sistema de comprobación de integridad de los ficheros transmitidos por red para garantizar la correcta recepción de los mismos.

■ Iteración 5: Sistema de logs

- Requisitos implementados: R14
- Descripción: Definir los mensajes necesarios para cuando en la aplicación ocurre algo que se desee guardar.

En cuanto a los requisitos R01 y R02 están implementados en arquitectura del sistema, no son funcionalidades.

En este capítulo se ha descrito el ámbito funcional del sistema y la metodología a utilizar para desarrollarlo. De acuerdo a este plan de desarrollo, se han obtenido los requisitos y planteado una serie de iteraciones a seguir para la construcción de la aplicación.

Capítulo 2

Segunda iteración

Contents

2.1. Inicio de la prueba	10
2.2. Examen temporizado	10
2.3. Denegar acceso a red	11
2.4. Pruebas	12

En el siguiente capítulo se describe lo realizado en la segunda iteración de construcción. Se parte de lo obtenido en las iteraciones previas, que es, la descripción del sistema, la arquitectura del mismo y los diagramas de secuencia y estados, todo esto expuesto a lo largo de los capítulos 1 y ??.

En cuanto al estado de la aplicación, en la primera iteración de construcción se ha desarrollado un sistema distribuido, en el que desde la aplicación del alumno se puede conectar con la aplicación del profesor y transmitir ficheros.

El objetivo de esta iteración es que el profesor pueda establecer el inicio y de la prueba, poder prefijar una hora de finalización de la misma y que la aplicación del alumno sea capaz de denegar el acceso a la red mientras dure la prueba.

En la tabla 2 se pueden ver los requisitos a implementar durante esta iteración y en la 2 la base, o lo que es lo mismo, lo implementado en la iteración anterior. La lista global de requisitos se encuentra en la tabla 1.3.

En las siguientes secciones se describen los incrementos realizados dentro de esta iteración, generalmente coincide el número de incrementos en cada iteración con el número de requisitos a implementar.

Identificador	Descripción
R03	El profesor debe ser capaz desde el <i>Watchman</i> de indicar el inicio de la prueba.
R05	El profesor ha de ser capaz desde el <i>Watchman</i> de establecer una hora límite.
R10	El alumno desde su computador debe tener la posibilidad de ver el tiempo restante.
R11	La aplicación del alumno ha de ser capaz de denegar el acceso a la red al profesor.

Cuadro 2.1: Objetivos de la iteración

Identificador	Descripción
R01	Un computador de la red debe poder ser designado como <i>Watchman</i> .
R02	Todos los computadores que no sean <i>Watchman</i> serán computadores normales.
R06	El profesor debe ser capaz desde el <i>Watchman</i> de enviar el fichero de enunciados.
R07	El alumno desde su computador debe ser capaz de conectarse al <i>Watchman</i> .
R13	La aplicación ha de ser capaz de comprobar que los archivos se han enviado correctamente.

Cuadro 2.2: Requisitos ya implementados del sistema

2.1. Inicio de la prueba

En esta sección se comenta cómo se consigue que el profesor pueda notificar a los alumnos el inicio de la prueba. Como ya hemos comentado, partimos de la iteración anterior en la cual los alumnos eran capaces de conectar a la aplicación del profesor. En la aplicación del alumno, cuando se crea una conexión, se inicia a su vez un hilo de ejecución que se mantiene a la espera de recibir distintas órdenes provenientes del computador del profesor. De este modo se hace muy simple mantener desde la aplicación del profesor una lista de las conexiones abiertas con cada alumno y, al presionar el botón de iniciar la prueba, recorrer esa lista enviando la orden por cada conexión. Cuando cada una de las aplicaciones del alumno recibe esa orden, actúa en consecuencia.

2.2. Examen temporizado

El profesor puede establecer una hora límite, llegada la cual, la prueba terminará automáticamente. Cuando el profesor define este límite en su aplicación y decide comenzar la prueba se envía también si hay una hora de fin y, en caso afirmativo, cual es. De este modo la aplicación del alumno puede mostrar una cuenta atrás con los minutos restantes para la finalización, para facilitar la referencia temporal. Se puede ver el resultado en la siguiente imagen.

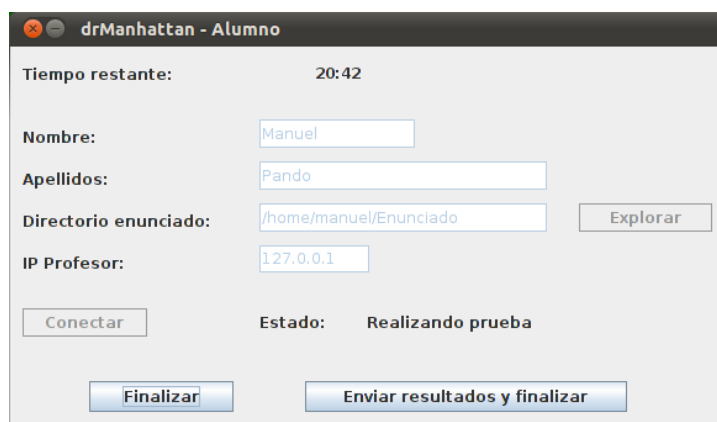


Figura 2.1: Aspecto de la GUI del alumno con una prueba temporizada.

2.3. Denegar acceso a red

Cómo ya hemos visto en las secciones anteriores, una vez que la aplicación del alumno recibe la orden de comenzar la prueba, se ha de denegar el acceso a la red. Para ello se utiliza iptables, por medio de un demonio.

El demonio creado es muy simple, se ejecuta en segundo plano esperando a que la aplicación del alumno conecte, y en función de lo requerido en ese momento, permitir o no el acceso a la red interactuando con iptables. Este demonio se inicia en tiempo de arranque y con los permisos necesarios para poder usar el cortafuegos.

Las órdenes concretas que ejecuta el demonio son, cuándo se desea denegar la red son:

```
iptables -P OUTPUT DROP
iptables -A OUTPUT -s 127.0.0.1 -j ACCEPT
```

Con la primera de ellas se cambia la política del tratamiento a los paquetes salientes del equipo, de modo que los deseche, dicho de otro modo, no se permiten las comunicaciones salientes con otros computadores, de esta forma se evita que se hagan peticiones, por ejemplo, a un servidor web.

La segunda es para permitir los paquetes salientes dirigidos al propio ordenador. Esto es necesario para la comunicación entre la aplicación del alumno y el demonio.

Puede parecer que el equipo queda totalmente aislado, incluso del computador del profesor, pero esto no es así, puesto que ya hay una conexión establecida y dado que la aplicación del alumno espera órdenes de la aplicación

del profesor, puede seguir funcionando perfectamente, puesto que los paquetes de entrada sí que están permitidos, lo que estas instrucciones imposibilitan es la creación de nuevas conexiones y el envío de mensajes a cualquier computador que no sea el propio.

2.4. Pruebas

En esta sección se describen las pruebas realizadas en la iteración.

Debido a la naturaleza del sistema, automatizar pruebas con una herramienta estilo *JUnit* es prácticamente imposible, ya que funciona a base de eventos y es distribuido, por eso, para probar la aplicación utilizo el equipo de sobremesa y un portátil, para intentar recrear el entorno del laboratorio y ejecuto secuencias de acciones para desencadenar los eventos requeridos en cada funcionalidad.

- **R03** - El profesor debe ser capaz desde el *Watchman* de indicar el inicio de la prueba.
 - Pulsar el botón y ver que efectivamente llega la señal al computador del alumno.
- **R05** - El profesor ha de ser capaz desde el *Watchman* de establecer una hora límite para la duración de la prueba.
- **R10** - El alumno desde su computador debe tener la posibilidad de ver el tiempo restante el pruebas de duración prefijada.
 - Establecer horas correctas futuras y comprobar que los minutos calculados restantes son correctos.
 - Establecer horas pasadas y comprobar que no se temporiza la prueba.
 - Establecer horas incorrectas y comprobar que es detectado.
- **R11** - La aplicación del alumno ha de ser capaz de denegar el acceso a la red al empezar la prueba.
 - Después de que de inicio la prueba, intentar abrir páginas web en el navegador, pings a equipos.