uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática
2020-2021

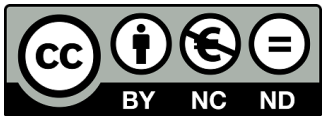
Trabajo Fin de Grado

"Análisis, diseño e implementación de una aplicación de escritorio remoto"

Antonio Manuel Hernández De León

Tutor/es
Alejandro Calderón Mateos
Javier López Gómez

Leganés, Octubre 2021



"Es el esfuerzo constante y decidido lo que rompe toda resistencia y barre con todos los obstáculos."

Claude M. Bristol

Agradecimientos

Quiero agradecerles desde lo más profundo de mi corazón a mis padres, Marcial y Rosa Delia, todo lo que me han enseñado a lo largo de mi vida; sin ustedes no habría podido llegar nunca donde estoy ahora. Muchísimas gracias por transmitirme siempre que con esfuerzo y dedicación se puede conseguir todo lo que uno se propone, por apoyarme en los momentos más complicados de mi vida y de la etapa universitaria, y por siempre confiar en mí.

A mi abuela, Nieves, por darme ánimos y transmitirme fuerza en todo momento, e inculcarme que el que quiere algo tiene que hacer un sacrificio y que después del esfuerzo se recogen los frutos.

A mi abuelo, Antonio, y a mi tío, Juan Antonio, que pese a que no estén físicamente conmigo, sé que lo están espiritualmente; y siento que siempre estarán orgullosos de mí y de ver donde he sido capaz de llegar.

A Javier López, por su vital ayuda, sin obligación, en todos estos meses en los que sin él no hubiera podido hacer un proyecto tan apasionante como este, por atenderme cada vez que necesitaba ayuda, por las explicaciones tan buenas que he recibido y todo lo que he aprendido, y por transmitir con tanta pasión todo lo que sabes. Nunca olvidaré lo que has hecho por mí, eres una maravilla de persona, te lo mereces todo. Mil gracias.

A mi tutor, Alejandro Calderón, por transmitirme su sabiduría, explicar tan bien cada cosa que transmite, y disfrutar tanto enseñando a los alumnos haciendo que a ellos les apasione también cualquier cosa que les explique. Muchas gracias por toda tu ayuda y tus consejos, y también por haberme dado clase tan magníficamente en tres asignaturas de la carrera.

Por último, también agradecer a todos mis amigos y a todas las personas que han coincidido conmigo y me han dado felicidad en alguna parte del camino de mi vida, marcando como una de las más especiales mi etapa en la universidad, la cual me ha hecho crecer y evolucionar como nunca antes habría podido imaginar.

¡Muchas gracias a todos, siempre los llevaré en mi corazón!

Resumen

Desde la aparición de la informática hasta los últimos años, los tipos de software que se han podido desarrollar han ido variando mucho. En sus inicios se ejecutaban en un solo ordenador, después pudieron hacerlo en varios, físicamente juntos, conectados a través de una misma red local; hasta la llegada de Internet, cuando fueron capaces de dar el salto a ejecutarse en varios ordenadores a miles de kilómetros de distancia. Estos programas reciben el nombre de programas distribuidos.

Los programas de control por escritorio remoto son unos de los programas distribuidos más populares. Típicamente, consisten en una parte servidora que se ejecuta en el equipo que se quiere controlar y una parte cliente que se ejecuta en la máquina desde la cual se va a controlar el equipo anterior, permitiendo de esta manera el uso completo del mismo como si se estuviese físicamente delante.
En los últimos tiempos, el uso de este tipo de software ha aumentado considerablemente por su gran utilidad. Para las empresas, es imprescindible su existencia, permitiéndoles ofrecer asistencia a distancia a sus trabajadores y clientes, o también facilitando el teletrabajo.

El objetivo de este Trabajo de Fin de Grado es el desarrollo de una aplicación de escritorio remoto que además de ofrecer las características que brindan las existentes, permita a cualquier interesado aprender cómo funciona esta tecnología examinando su código fuente.
Para cumplir dicho objetivo, la simplicidad del código de la aplicación y su compatibilidad con un entorno familiar para la mayor parte de la población es fundamental. Por este motivo, se ha hecho su desarrollo para el sistema operativo de Windows, utilizando la plataforma .NET en la mayor parte de la implementación, facilitando y abstrayendo de esta manera la interacción con el sistema operativo, consiguiendo que el código sea más legible y sencillo.

Palabras clave: Control remoto, escritorio remoto, sistema distribuido, cliente-servidor.

Abstract

Since the early years of computing, the kind of software that has been developed has been changing a lot. In the beginning, software ran in a single computer; afterward, it could do on several computers that were physically co-located (connected to the same local network) until, with the arrival of the internet, they were able to run in many computers being hundreds of miles away from each other. These programs are called distributed programs.

The remote desktop control programs are some of the most popular distributed programs. Typically, they consist of one server part that runs on the machine that is to be remotely controlled, and the client part which runs on the machine that is going to control the server. This way, it allows the complete use of the desktop environment as if the user were physically at the system console. The use of this kind of software has increased during recent years because of its usability. In the industry, its existence is essential, not only because they are able to offer remote assistance to their clients and employees, but also for facilitating telework.

The objective of this Bachelor's Degree Final Project is the development of a remote desktop application that in addition to the features typically offered by other well-known remote desktop applications, it allows anybody to learn how this technology works by reading its source code. In order to achieve this objective, the simplicity of the application's code and its compatibility with the familiar environment to the majority of users is fundamental. For this reason, the developed application was created for the Windows' operating system, using the .NET Framework in the vast majority of the implementation, facilitating and abstracting part of the interaction with the operating system, resulting in easier and more readable code.

Keywords: Remote control, remote desktop, distributed system, client-server.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Desde la aparición del ser humano, la humanidad siempre ha intentado construir artilugios que le faciliten cualquier tipo de tarea que pueda realizar. Descubrimientos como los del fuego y la rueda marcaron los inicios de sus progresos, llegando hasta los época actual con la creación del ordenador y la llegada de Internet, marcando una verdadera revolución tecnológica, dando la bienvenida a la era digital en la sociedad.

La aparición de Internet supuso un enorme cambio tanto social como económico, permitiendo la ejecución de multitud de tareas a distancia que antes eran impensables, como el comercio y la banca online, videoconferencias, teletrabajo, etc.

Labores como el teletrabajo o la asistencia remota se pueden hacer de muchas maneras, una de las opciones más innovadoras es realizarlas con un software de control remoto, el cual permite acceder a una máquina a distancia y manejarla como si se estuviese físicamente delante de ella.

1.1. Motivación

Con el avance de la tecnología y la expansión masiva de Internet en los últimos años se ha visto cómo las necesidades tecnológicas de las personas han ido cambiando. Hace cuarenta años no era necesario tener un ordenador en los hogares, sin embargo, en la actualidad este dispositivo es indispensable para poder trabajar. Esta necesidad se está viendo incrementada conforme pasa el tiempo debido a que cada vez más procesos se han ido digitalizando (comercio online, cita previa por Internet, educación a distancia telemática, etc.).

Las herramientas de comunicaciones digitales tales como las de videoconferencia, control remoto, etc. han tomado mucho protagonismo en los últimos años, más aún después de la pandemia de la COVID-19. Entre todas estas herramientas, destacan sobre todo las que ofrecen control remoto, debido a que permiten la asistencia a distancia cuando hay problemas en un equipo determinado o simplemente se necesita ayuda.

Una aplicación de control por escritorio remoto ofrece la posibilidad de controlar una máquina a distancia tal y como si se estuviese delante de ese dispositivo. Este proyecto nace de la necesidad de poder enseñar cómo funciona esta tecnología (imprescindible actualmente) en el ámbito de la docencia académica. Para ello se propone el desarrollo de una aplicación que ofrezca asistencia

de escritorio remoto de una manera sencilla y lo más eficiente posible, garantizando que sea apta para su uso en la docencia de prácticas de ciertas asignaturas del grado como, por ejemplo, Sistemas Distribuidos.

En el mercado existen varias soluciones que ofrecen esta funcionalidad. Cada una de ellas tiene diferentes ventajas e inconvenientes, pero todas tienen en común que son sumamente complejas y que es muy complicado entender su funcionamiento, por lo que dichas soluciones están lejos de ser aptas para propósitos docentes.

Para este Trabajo de Fin de Grado se propone una solución sencilla, de código abierto, que sea fácilmente entendible y también apta para la docencia. Lo más importante es que cumple con los requisitos mínimos que debe tener una aplicación de escritorio remoto y que se consigue esta funcionalidad con una complejidad muy inferior, de varias órdenes de magnitud en cuanto a líneas de código, en contraste con las soluciones existentes.

1.2. Objetivos

A continuación, se enumeran los objetivos que se pretenden alcanzar con la realización de este proyecto:

- **Poder controlar un equipo remotamente a través de una conexión de escritorio remoto**
  El usuario podrá ver todo lo que ocurra y manejar el escritorio del ordenador que quiere controlar remotamente, como si estuviera físicamente en el mismo lugar.

- **Hacer un uso lo más eficiente posible de los recursos utilizados, adaptándose a las capacidades del equipo donde se ejecute**
  Se pretende gestionar de la mejor manera posible los recursos que utilice el programa, tales como la CPU, la memoria RAM o el ancho de banda de red consumido. También se tratará de conseguir la adaptación a las capacidades del equipo donde se ejecute, minimizando en lo posible los requisitos mínimos de ejecución necesarios.

- **Realizar la transmisión de la información con conexiones seguras**
  El usuario tendrá la posibilidad de elegir si quiere que sus datos se transmitan en claro o de manera cifrada.

1.3. Estructura del documento

En esta sección se expone una descripción del contenido de los capítulos que constituyen este documento. Dichos capítulos son los siguientes:

- **Estado del arte**. Se expone la historia de la tecnología de control remoto diferenciando cada uno de sus hitos. También se analizan, detenidamente, las soluciones existentes más populares de control remoto para posteriormente hacer una comparativa de las mismas.

- **Análisis y diseño**. Consta de un estudio inicial en el que se toman decisiones cruciales para las siguientes fases del proyecto. Asimismo, se definen los requisitos de usuario, casos de uso y también los requisitos de software derivados de los de usuario. Por último, se incorporan los detalles de la arquitectura del software desarrollado.
- **Implementación**. Incluye los aspectos de más importancia de la implementación de los componentes del sistema y también de las herramientas que se han utilizado para ello.
- **Pruebas y evaluación**. Abarca la definición de las pruebas realizadas para validar el funcionamiento apropiado del sistema, y también la evaluación del rendimiento del programa mostrando el impacto que tiene sobre los recursos de los equipos donde se ejecuta.
- **Planificación y presupuesto**. En él se refleja la planificación de las tareas en las que se ha dividido el proyecto incluyendo un diagrama de Gantt, así como el presupuesto necesario para llevar a cabo el conjunto de dichas tareas.
- **Marco regulador y entorno socio-económico**. Expone la legislación que se aplica al proyecto y además, la repercusión social y económica del mismo.
- **Conclusiones y trabajos futuros**. Contiene las conclusiones que se han obtenido con la realización del TFG, conjuntamente con las mejoras que se podrían incluir en el futuro.

El documento también incorpora varios apéndices, los cuales contienen información de distinto tipo:

- **Resumen en inglés**. Corresponde al Apéndice *Summary*.
- **Vocabulario técnico del proyecto**. Reflejado en el Apéndice *Glosario*.
- **Instrucciones de instalación y uso del programa**. Mostradas en el Apéndice *Manual de usuario*.
- **Detalles de las funciones de las clases expuestas en el diseño**. Indicados en el Apéndice *Detalles de las clases del diseño*.
- **Vista general de conceptos necesarios para la lectura de esta memoria**. Explicados en los Apéndices de *Vista general API Win32*, *Redes* y *Seguridad Informática*.

Capítulo 2

Estado del arte

El presente capítulo, primeramente, consta de la historia del surgimiento y la evolución de la tecnología de control remoto, expuesta en la Sección 2.1. Paralelamente, en las Secciones 2.2 y 2.3, se realiza un estudio de las soluciones de control remoto más populares. Para concluir, en la Sección 2.4, se lleva a cabo una comparativa de las soluciones anteriormente estudiadas con la propuesta para este proyecto.

2.1. Surgimiento y evolución de la tecnología de control remoto

Al principio de los tiempos, las interfaces que usaban los ordenadores eran de texto, todas las órdenes que se podían ejecutar en ellos había que escribirlas a través de comandos de texto. Posteriormente, a partir del año 1973, con la aparición del primer ordenador que utilizó una interfaz gráfica, llamado Xerox Alto [1] (véase la Figura 2.1), las interfaces de texto fueron desapareciendo en favor de las interfaces gráficas que se manejaban por medio de ventanas.

Figura 2.1: Xerox Alto [1]

Como es de esperar, en sus comienzos, el control remoto se realizó a través de comandos de texto por shell remota y cuando aparecieron las interfaces gráficas surgió la tecnología de escritorio remoto, consistiendo en el control remoto de la interfaz gráfica (típicamente un entorno

de escritorio). En relación con la tecnología de control remoto, la historia de la informática se puede dividir en tres hitos:

- **Desde el inicio de la informática hasta 1969**. Durante esta época todos los ordenadores usaban interfaces de texto y no existía Internet. Tampoco había ninguna tecnología similar con la que se pudiese acceder remotamente a ningún dispositivo, por lo que el control remoto no existía.

- **Desde 1969 hasta 1984**. En el año 1969 surge la red norteamericana Arpanet [2], la cual se utilizó como medio de comunicación entre distintas instituciones. De dicha red surge el Internet que existe actualmente. Gracias a este acontecimiento, en este periodo ya fue posible acceder y controlar remotamente otro equipo a través de comandos de texto. A pesar de que en el año 1973 surgiese el primer ordenador que usase una interfaz gráfica (Xerox Alto), no fue hasta el año 1984 cuando fue posible el control remoto gráfico. Véase la Figura 2.2 para visualizar una foto del surgimiento de Arpanet.

Figura 2.2: Nacimiento de Arpanet [3]

- **Desde 1984 hasta la actualidad**. En el año 1984 se creó el sistema X Window [4], el cual se convirtió en el entorno gráfico de los ordenadores que ejecutaban el sistema operativo Unix, estando todavía presente en los sistemas operativos heredados de este. Dicho sistema es peculiar debido a que está construido con una arquitectura cliente-servidor, por lo que brindó la posibilidad de ejecutar una aplicación determinada en un ordenador (actuando este como cliente) y visualizar todos los resultados gráficos en otro, adquiriendo este último el rol de servidor.
  Este fue el primer contacto con lo que se conoce hoy en día como control por escritorio remoto y se diferencia en que en este sistema los roles cliente y servidor están invertidos.

2.2. Soluciones existentes de control por Shell remota

Telnet

Telnet (**Tel**ecomunication **Net**work) [5] es un protocolo de red TCP/IP y un programa que sirve para establecer conexiones remotas con otros ordenadores utilizando por defecto el puerto 23. Este programa no tiene interfaz gráfica, por lo que se debe utilizar mediante una terminal. Para poder empezar a usar Telnet hay que tener el programa cliente en la máquina desde la cual se empieza la conexión y el programa servidor en el dispositivo al que se quiere acceder. A parte de esto se debe conocer el nombre de la cuenta y la contraseña del equipo al que se desea conectar.

Si se tiene el propósito de acceder remotamente a un ordenador se debe ejecutar el comando "telnet <nombre dominio>" o "telnet <dirección IP>". Típicamente, también es utilizado por administradores de sistemas como simple medio para verificar si un puerto TCP está abierto con "telnet <dirección IP> <puerto>".

Actualmente, el uso de Telnet no está recomendado debido a que toda la información (incluido usuario y contraseña) transmitida de una máquina a otra no se transmite cifrada; esto significa que cualquiera que esté escuchando el tráfico de la red puede ver dicha información en claro. También carece de un esquema de autentificación que asegure que las comunicaciones no sean interceptadas [6], por lo que solo se recomienda su uso en redes internas aisladas del exterior, fuera de este entorno se aconseja el uso de SSH.

Como observaciones se pueden destacar las siguientes:

- **Ventajas**
  - Posibilidad de desarrollar una aplicación que implemente el protocolo al ser público.
  - Herramienta muy apta para aprender conceptos de administración de redes en docencia.
- **Desventajas**
  - Los datos que se transmiten no van cifrados, por lo que no está recomendado su uso en redes que no sean de área local.
  - Actualmente, al ser las interfaces de los equipos gráficas, tienen más demanda las aplicaciones gráficas con capacidad de control por escritorio remoto.

Rlogin

Rlogin (**R**emote **Login**) [7] es una herramienta para acceder remotamente a otro equipo. Nació en los años 80 para los BSD Unix, y al igual que Telnet, ha sido remplazado por SSH por razones de seguridad. Es importante remarcar que es más sencillo que Telnet al no haber tanta negociación de opciones debido a su compatibilidad solo con Unix [6]. La información transmitida no va cifrada y tiene soporte de autenticación de ficheros .rhosts, esto implica que se permite el inicio de sesión sin contraseña y que estos archivos al depender de las direcciones IP, hacen que la suplantación de las mismas sea más sencilla.

Como observaciones se pueden destacar las siguientes:

- **Ventajas**
  - Más sencillo que Telnet.
- **Desventajas**
  - Los datos que se transmiten no van cifrados, por lo que no está recomendado su uso en redes que no sean de área local.
  - Actualmente, al ser las interfaces de los equipos gráficas, tienen más demanda las aplicaciones gráficas con capacidad de control por escritorio remoto.

SSH

SSH (**S**ecure **Shell**) [8], al igual que Telnet, es un protocolo y un programa que sirve para establecer conexiones remotas con otros equipos utilizando por defecto el puerto 22. Al contrario que Telnet, en este protocolo, la información viaja cifrada, pudiendo ser el método de cifrado de tres tipos (simétrico, asimétrico o hashing). También existe un mecanismo de autenticación que consiste en que las contraseñas viajen por un túnel cifrado simétricamente.

En sistemas operativos de la familia Unix (GNU/Linux y MacOS) se puede usar esta herramienta por medio de una terminal; en Windows hay que instalar un cliente SSH como, por ejemplo, Putty.
Para empezar una conexión remota hay que teclear "ssh <user>@<host>" en la terminal, siendo user el nombre de la cuenta y host una dirección IP o dominio; posteriormente, será requerida una contraseña.

Como observaciones se pueden destacar las siguientes:

- **Ventajas**
  - Posibilidad de desarrollar una aplicación que implemente el protocolo al ser público.
  - Los datos que se transmiten van cifrados, por lo que se puede utilizar estableciendo una conexión segura con el destino en redes que no sean de área local.
- **Desventajas**
  - Actualmente, al ser las interfaces de los equipos gráficas, tienen más demanda las aplicaciones gráficas con capacidad de control por escritorio remoto.

2.3. Soluciones existentes de control por escritorio remoto

Sistema X Window

El sistema X Window [4] es el entorno gráfico de los sistemas Unix. Fue creado en el Instituto Tecnológico de Massachusetts (MIT) con el nombre de proyecto Athena en 1984 siendo

el objetivo principal conseguir la compatibilidad de los distintos terminales gráficos. Athena [9] nació como un proyecto educativo que fue financiado por IBM y DEC en el MIT, y se centró en proporcionar a todos los estudiantes del instituto un entorno de computación distribuida para darles accesso sistemático a los ordenadores. A lo largo de los años se han encargado de su desarrollo corporaciones como X Consortium, Open Group y X.org Foundation, siendo esta última la que actualmente lo mantiene.

X Window está construido con una arquitectura cliente-servidor y es una capa totalmente independiente del sistema operativo, posee tres componentes:

- **Servidor X**. Se ejecuta en la máquina que posee teclado/ratón/monitor y es el encargado de ejecutar las operaciones de dibujado que provienen de los clientes X.
- **Cliente X**. Es cualquier aplicación que muestra su GUI remotamente sobre la pantalla manejada por el servidor X. De la misma forma, recibe mensajes del servidor X asociados con pulsaciones de teclas, movimiento del ratón, ocultación/descubrimiento de una ventana, etc.
- **Gestor de pantalla**. Es un cliente X "especial" que se encarga, entre otras cosas, de añadir la decoración de ventanas (barra de título, etc.).

La comunicación en el sistema se realiza usando el protocolo X11 [10], donde existen cuatro tipos de mensajes [11]:

- **Petición**. Enviado por el cliente al servidor solicitando la ejecución de una orden.
- **Respuesta**. Enviado por el servidor al cliente como respuesta a una petición. Son mensajes con información técnica como posiciones de las ventanas, tamaño de ellas, etc. No todas las peticiones generan una respuesta.
- **Evento**. Es un mensaje que le envía el servidor al cliente cuando ocurre un evento, como el movimiento de una ventana, pulsaciones de teclado, etc.
- **Error**. Se genera en el servidor y es enviado al cliente en caso de que una petición sea incorrecta.

X11 ha evolucionado mucho durante las últimas décadas, por lo que es enormemente complejo, soportando las primitivas de dibujo de las versiones iniciales, pero tambíen extensiones, por ejemplo:

- **DRI y GLX**. Usadas para rendering directo y OpenGL.
- **Xinerama**. Empleada para la compatibilidad de múltiples monitores.
- **Composite**. Para soporte de gestores de composición.
- **Damage y Render**. Utilizas por kits de herramientas gráficas modernas como GTK+ o Qt.

Debido a que X Window tiene una arquitectura cliente-servidor, se puede utilizar dicho software como escritorio remoto permitiendo visualizar la ejecución de un programa en una máquina cuando este se está ejecutando en otra [12]. Este fue el primer entorno operativo de escritorio remoto. Es importante mencionar que al ser independiente del hardware, posibilita la interoperabilidad entre distintos sistemas operativos; permitiendo por ejemplo, visualizar aplicaciones que se estén ejecutando remotamente en una máquina con GNU/Linux en un servidor X instalado en Windows como Xming.

Como observaciones se señalan las siguientes:

- **Ventajas**
  - Código abierto con posibilidad de examinarlo y saber si es software espía.
  - Posibilidad de desarrollar una aplicación que se comunique con el sistema al ser su protocolo público.
- **Desventajas**
  - Código complejo y difícil de entender su funcionamiento.
  - Protocolo extremadamente complejo, teniendo en cuenta las múltiples extensiones soportadas por un Servidor X moderno. Estas extensiones han sido necesarias para adaptarse a las necesidades de la actualidad, aumentando la complejidad del protocolo y haciéndolo más pesado.
  - Técnica utilizada para funcionar compleja y anticuada debido a que las primitivas gráficas del "core" del protocolo rara vez se usan por herramientas gráficas modernas siendo el resultado pobre (sin antialiasing, sin composición, etc.). Solo apta para situaciones excepcionales.

VNC

VNC (Virtual Network Computing) [13] es un programa de escritorio remoto que fue desarrollado por Olivetti & Oracle Research Labs en Inglaterra en la década de los 80. Posteriormente, dicha institución fue adquirida por AT&T Corporation.
VNC está construido con una arquitectura cliente-servidor y funciona con independencia del sistema operativo que tenga instalado tanto la máquina cliente como la servidora, posee dos componentes:

- **Servidor**. Ofrece el acceso de la máquina en la que se ejecuta, dejando al cliente tomar el control de la misma y enviándole a este los resultados de todas sus peticiones.
- **Visor**. Se le llama de esta manera a la parte cliente y gracias a él, el usuario final puede enviarle órdenes al servidor controlando de esta manera la máquina en la que se ejecute.

Su código fuente se publicó como software libre bajo la licencia GPL y actualmente la versión original solo se usa como referencia y compatibilidad. Existen diferentes variaciones de la misma [13, 14], algunos ejemplos son:

- **TightVNC** [15]. Con su lanzamiento se mejoraron los algoritmos de compresión y se añadió la funcionalidad de transferencia de archivos. No cifra las conexiones.
- **TigerVNC** [16]. Fue creado por Red Hat como una bifurcación de TightVNC con el objetivo de mejorarlo incorporando características como el cifrado de conexiones.
- **UltraVNC** [17]. Esta versión incorpora transferencia de archivos, chat de texto y diversos métodos de autentificación. También ofrece la posibilidad de controlar una única ventana de un programa en lugar de todo el escritorio.
- **RealVNC** [18]. Es la única de las mencionadas que a parte de tener una versión gratuita para un uso no comercial también tiene una versión empresarial de pago. Es la variante que más funciones incorpora.

El protocolo que utiliza VNC se llama Remote Framebuffer (RFB) [19] y sus características a destacar son las siguientes:

- **Opera en el nivel de framebuffer**. Debido a esto, consigue ser independiente del sistema operativo y se puede usar para todos los sistemas de ventanas. Todas las variantes de VNC son compatibles entre sí por esta razón.
  El framebuffer [20] es un dispositivo virtual del sistema operativo en el que se almacenan todos los píxeles de una imagen antes de que se muestren en la pantalla. Este dispositivo funciona como un mecanismo de abstracción, en el que los procesos pueden escribir sin necesidad de saber los detalles de implementación e interacción de cada sistema operativo con los dispositivos externos, y posteriormente mostrar información en la pantalla.
- **No mantiene estado**. A diferencia del protocolo X11, este protocolo no mantiene estado. Esto significa que si un cliente se desconecta y después se reconecta al mismo servidor, el estado del escritorio de este último será el mismo que el mostrado en el cliente antes de desconectarse.

Como observaciones se pueden destacar las siguientes:

- **Ventajas**
  - Código abierto con posibilidad de examinarlo y saber si es software espía.
  - En el caso de RealVNC, apto para empresas y usuarios exigentes.
  - Posibilidad de desarrollar una aplicación que se comunique con ella sustituyendo una de sus partes (cliente o servidor) al ser su protocolo público.
- **Desventajas**
  - Salvo en el caso de RealVNC, compleja configuración solo destinada a usuarios expertos.
  - Código complejo y difícil de entender su funcionamiento.
  - En el caso de RealVNC, su versión más completa es de pago.

Terminal Server (Microsoft Windows)

El término de Terminal Server (Servidor de Terminales) [21] se empezó a utilizar para designar a un ordenador potente al que se conectaban otros equipos con mínimas prestaciones denominados terminales. Dicho servidor de terminales era el que gestionaba y distribuía la potencia que requerían los ordenadores que se conectaban a él.

Actualmente, el término hace referencia a un servicio incorporado en los sistemas operativos Windows que ofrece la posibilidad de conectarse a otra máquina y controlar su escritorio remotamente [22]. Con el tiempo se le ha ido cambiando el nombre a este servicio, ahora llamándose Remote Desktop Service, o Servicio de escritorio remoto en español.

Para utilizar este servicio hay que habilitarlo en la configuración tanto del cliente como del servidor. Solo se puede habilitar en versiones Pro de Windows.

Este servicio utiliza un protocolo propietario de Microsoft llamado Remote Desktop Protocol (RDP) [23], destacando de él que si se realiza una conexión con una versión de Windows no destinada a servidores, solo permitirá una sesión activa en el mismo. Es decir, el usuario cuya máquina actúa como servidor no podrá ver lo que está sucediendo, se le cerrará la sesión si la tenía abierta, algo que no ocurre en los demás programas de escritorio remoto. Si se realiza una conexión con una versión servidor de Windows no sucede esto, permitiendo trabajar a más de un usuario simultáneamente.

Se pueden destacar las siguientes observaciones:

- **Ventajas**
  - Está integrado por defecto en el sistema operativo.
  - Posibilidad de desarrollar una aplicación que se comunique con ella sustituyendo una de sus partes (cliente o servidor) al ser su protocolo público.
- **Desventajas**
  - Código privativo, imposibilidad de entender su funcionamiento y de saber si posee partes que espían.
  - La parte servidora solo se puede ejecutar en las versiones de Windows Pro.
  - Limitaciones en el uso para ediciones de Windows no destinadas a servidores.

TeamViewer

TeamViewer [24] es una de las soluciones más completas de escritorio remoto, pudiéndose considerar un todo en uno debido a que no solo tiene la función de ofrecer el control del escritorio de una máquina, sino que también proporciona funcionalidades extras como llamadas telefónicas y VoIP, VPN, Wake-on-LAN, asistencia de dispositivos IoT y de realidad aumentada, monitorización de sitios web propietarios, etc.

Este software se caracteriza por ser de código privativo y por utilizar un protocolo propietario de sus creadores que no ha sido publicado. Tiene versiones compatibles con cada sistema operativo y posee una licencia gratuita para uso doméstico y otras de pago para uso comercial.

Como observaciones predominan las siguientes:

- **Ventajas**
  - Software todo en uno.
  - Ideal para empresas y usuarios avanzados que demandan funcionalidades complejas y avanzadas.
- **Desventajas**
  - Código privativo y candidato a ser extremadamente complejo, imposibilidad de entender su funcionamiento y de saber si posee partes que espían.
  - Incapacidad de desarrollar una aplicación que se comunique con ella debido a que el protocolo utilizado no es público.
  - Para ejecutar muchas de sus funcionalidades avanzadas hay que realizar una configuración no intuitiva.
  - Su versión más completa es de pago.

Escritorio remoto de Google Chrome

Este software es una de las soluciones de escritorio remoto más sencilla de utilizar y gestionar [25]. Ni siquiera se instala en el ordenador en cuestión, sino que es una extensión del navegador de Google Chrome.

Como observaciones se pueden recalcar las siguientes:

- **Ventajas**
  - Interfaz minimalista con escasos elementos e inclusión exclusivamente de funcionalidades básicas de conexión remota.
  - No hace falta instalarlo, por lo que no hay que conocer la versión del sistema operativo que hay que descargarse teniendo en cuenta el instalado en la máquina en la que se va a ejecutar.
  - Funciona con la cuenta de Google.
  - Configuraciones no intuitivas que normalmente las tiene que especificar el usuario (conectarse sin tener el programa en concreto abierto, Google Chrome en este caso, o reconocer los equipos propietarios del usuario y tener una conexión prácticamente instantánea) realizadas automáticamente.
- **Desventajas**
  - Código privativo, imposibilidad de entender su funcionamiento y de saber si posee partes que espían.
  - Incapacidad de desarrollar una aplicación que se comunique con ella debido a que el protocolo utilizado no es público.

Windows 365

Windows 365 [26] es una herramienta que ha anunciado recientemente (Julio 2021) Microsoft, la cual se trata de un sistema operativo en la nube que se podrá acceder remotamente desde cualquier dispositivo sin importar del tipo que sea a través de un navegador web. Esta herramienta como tal no es de control por escritorio remoto, pero incorpora dicha tecnología debido a que las operaciones que realiza para poder visualizar el estado de la máquina en nube directamente en el navegador, son las mismas que las que se ejecutarían si se comenzase una sesión remota con otro programa de los mencionados anteriormente.
Por el momento estará destinada únicamente para el sector empresarial.

Como observaciones se pueden recalcar las siguientes:

- **Ventajas**
  - Acceso muy sencillo y cómodo, convirtiendo cualquier dispositivo que tenga un navegador web instalado en un Windows. Independencia total del hardware.
  - Al tener todos los datos en la nube se puede acceder desde cualquier parte sin tener que preocuparse antes de subirlos o de realizar una configuración previa para que se suban automáticamente.
- **Desventajas**
  - Código privativo, imposibilidad de entender su funcionamiento y de saber si posee partes que espían.
  - Su filosofía es el almacenamiento de todos los datos en la nube. No se sabe si se procesan dichos datos para sacar información de los usuarios. Hay mucho menos control de esto que en un sistema operativo instalado en un disco duro local propio.

2.4. Comparativa de soluciones de control remoto

Después de analizar las distintas soluciones existentes en los apartados anteriores se extraen las siguientes conclusiones:

- Actualmente, las aplicaciones de control por shell remota tienen un uso muy restringido, siendo adecuadas solamente en situaciones excepcionales como el uso de sistemas operativos que puedan ser usados casi al completo con la terminal (los cuales son escasos y no demasiado populares entre la población) o la administración de determinados ajustes de configuración de un sistema operativo (redes, gestionar capacidades de un disco duro, etc).
- En una conexión de control remoto dentro de una red de área local no es excesivamente importante que la información transmitida vaya cifrada debido a que es un entorno seguro. Sin embargo, si se tiene una conexión a través de Internet, es vital que los datos vayan cifrados para que dicha conexión sea segura. Esto deriva que las soluciones como Telnet, Rlogin y algunas variantes de VNC (por ejemplo TightVNC), las cuales no cifran las conexiones, no sean aptas para este propósito.

- Los programas más completos incluyendo más funcionalidades son TeamViewer y RealVNC, estando en el primer puesto TeamViewer, por lo que son los más adecuados para entornos empresariales y para usuarios exigentes que usen funciones avanzadas en sus hogares.
- La aplicación más apta para usuarios con pocos conocimientos informáticos es la brindada por Google debido a que todas sus configuraciones son casi instantáneas.
- Para los usuarios que se preocupen por su privacidad y quieran usar software libre que pueda ser examinado por expertos, con el objeto de evitar el espionaje y el procesamiento de datos personales, las mejores opciones son las variantes de VNC o el sistema X Window.

Pero a pesar de haber soluciones muy buenas y completas, no existe ninguna que permita a estudiantes en instituciones educativas o a personas curiosas entender cómo funciona la tecnología de control por escritorio remoto. Las soluciones existentes publicadas como software libre tienen un código demasiado complejo para entenderlo y usarlo como medio docente y las que son de código privativo tampoco se puede ni ver ni entender su código fuente.

En la Tabla 2.1 se muestra la comparativa de las soluciones analizadas en el apartado anterior con la propuesta de este TFG. Se puede observar que es la única que cumple con todas las características mencionadas en dicha tabla, siendo la más importante la del código apto para la docencia.

Solución	**Escritorio remoto**	**Gratuita**	**Código abierto**	**Protocolo público**	**Conexiones cifradas**	**Código apto para docencia**
Telnet	✗	✓	✓	✓	✗*	✗
Rlogin	✗	✓	✓	✓	✗*	✗
SSH	✗	✓	✓	✓	✓	✗
Sistema X Window	✓	✓	✓	✓	✗	✗
VNC	✓	✓*	✓*	✓	✓*	✗
TeamViewer	✓	✓*	✗	✗	✓	✗
Terminal Server (Microsoft Windows)	✓	✗	✗	✗	✓	✗
Escritorio remoto Google Chrome	✓	✓	✗	✗	✓	✗
Windows 365	✓	✗	✗	✗	✓	✗
Propuesta TFG	✓	✓	✓	✓	✓	✓

Tabla 2.1: Comparativa soluciones[1]

La propuesta de este proyecto es apta para docencia debido a que su código no es extenso y es sencillo de entender, así como el protocolo implementado; por lo que desarrollar una aplicación que se comunique con esta, no solo es posible sino también asequible para prácticas docentes.

[1]Las versiones más avanzadas de la variante de RealVNC y TeamViewer son de pago, solo sus versiones de uso doméstico son gratuitas. Es importante marcar que algunas variantes como TightVNC no cifran las conexiones; Telnet y Rlogin tampoco las cifran por sí mismas, se puede utilizar un túnel SSH en conjunto con ellas para cifrarlas pero esto no saben hacerlo todos los usuarios y los que tienen dicho conocimiento también pueden olvidarse. Se concluye que es más adecuado que esta característica venga integrada en el sistema.

Capítulo 3

Análisis y diseño

En este capítulo se reflejan los procesos de análisis y diseño del sistema. En primer lugar, en la Sección 3.1 se realiza un estudio inicial dentro del cual se deciden aspectos esenciales para las siguientes fases del proyecto. En la Sección 3.2 se obtienen los requisitos de usuario, los casos de uso y los requisitos de software asociados a los de usuario; incluyendo además matrices de trazabilidad. Por último, en la Sección 3.3 se realizan los diagramas de clases, de componentes y de secuencia, la matriz de trazabilidad entre componentes y requisitos de software, el protocolo implementado en la aplicación y los distintos roles dentro del esquema PKI usado para conexiones cifradas.

3.1. Estudio inicial

En esta sección se estudiarán las opciones que existen para desarrollar la aplicación propuesta respecto al sistema operativo con el que será compatible, la técnica de detección de cambios y captura de imagenn que se utilizará, y el lenguaje de programación y la plataforma de desarrollo en la que se realizará la implementación.

3.1.1. Sistema operativo

El programa se ha desarrollado para el sistema operativo de Microsoft Windows, habiendo escogido esta opción debido a que es por excelencia el más utilizado con una cuota de mercado de casi 90 % como se observa en la Figura 3.1. Si se hubiese escogido otro sistema operativo como GNU/Linux, hubiese sido mucho más complicado conseguir el objetivo principal del proyecto, que es el que estudiantes o personas curiosas puedan leer con detenimiento el código fuente, ejecutarlo y saber cómo funciona la tecnología de control por escritorio remoto. Esto es así, debido a que dicho sistema operativo no solamente es menos popular y tiene menos usuarios, sino que típicamente no viene pre-instalado en ninguna máquina al comprarla. En cambio, si se realiza el desarrollo para un entorno en el que muchas personas están familiarizadas, como lo es Windows, se podrá cumplir dicho objetivo de una manera más sencilla.

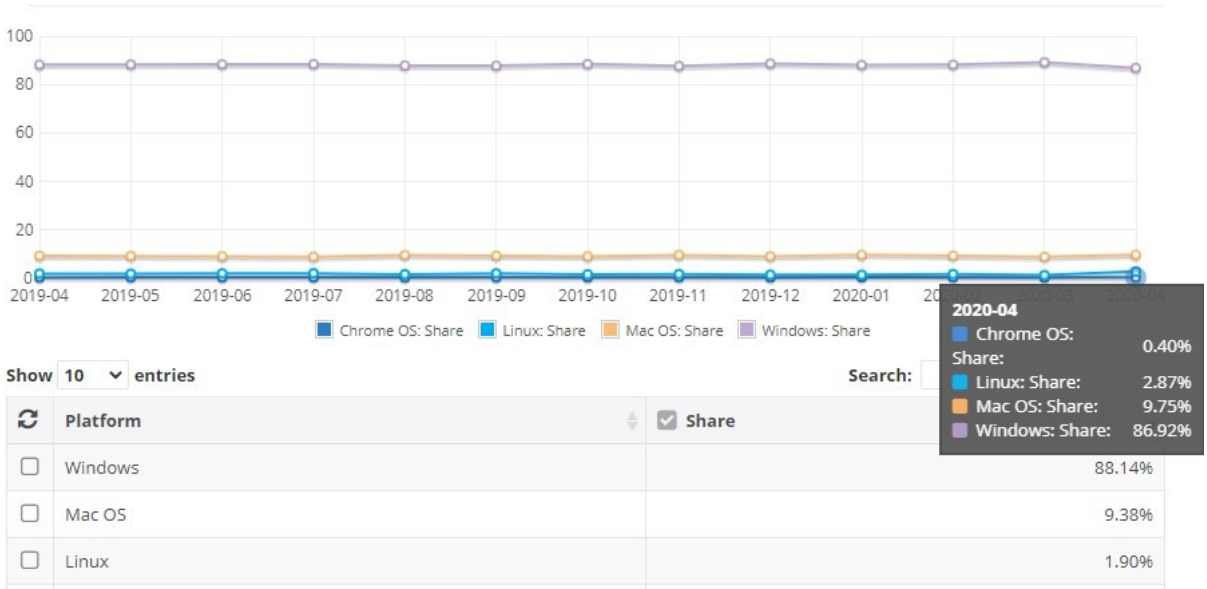


Figura 3.1: Cuota de mercado de Windows [27]

Por lo tanto, se concluye que escoger Windows como sistema operativo para el proyecto es la mejor opción. Se puede descargar y ejecutar el código fuente del programa tan solo teniendo que instalar el IDE de Visual Studio seleccionando los componentes necesarios para desarrollar aplicaciones de .NET en los correspondientes pasos de la instalación.

3.1.2. Técnica de detección de cambios y captura de imagen

Para poder transmitir las imágenes del servidor al cliente en una sesión de escritorio remoto se pueden usar las siguientes técnicas:

- **Polling**. Consiste en configurar un temporizador para tomar instantáneas del escritorio del servidor periódicamente y poder enviarlas a la máquina cliente.
- **Hook de filtro de mensajes**. En Microsoft Windows, la capa de interfaz de usuario mantiene una cola de mensajes que son despachados a las colas locales de cada ventana para su procesamiento. Estos mensajes incluyen eventos producidos por el usuario y el sistema, como por ejemplo los movimientos de ratón sobre una ventana, o que una región de una ventana deba ser repintada porque su contenido ha cambiado, etc.
  Instalar el hook permitiría interceptar dichos mensajes entregados a las colas de cada ventana con el fin de detectar exactamente qué partes del escritorio están cambiando, capturar la imagen visible en ese área, comprimirla y enviarla. Este método ofrece unos resultados mucho más precisos con una menor carga de CPU que el anterior, pero es bastante más complicado.
- **"Mirror driver" de gráficos**. Esta técnica require la implementación de un driver que replicaría las operaciones realizadas por el driver primario y se podría usar para interceptar a muy bajo nivel qué partes de la imagen del escritorio están cambiando y capturarlas eficientemente. Esta opción es con diferencia la más eficiente, pero también es la que más complejidad tiene y queda fuera del alcance de un TFG.

La técnica utilizada para el proyecto ha sido la de polling debido a que las demás aumentaban mucho la complejidad del mismo dejando de ser asequible y porque también se trata de un software de un proyecto académico, no de un software comercial.

3.1.3. Lenguaje de programación y plataforma de desarrollo

Los lenguajes de programación más adecuados para realizar este proyecto son C o C++ ya que la API Win32 de Windows ofrece sus funciones en dichos lenguajes, pudiendo de esta manera interactuar con el sistema operativo a bajo nivel para lo que se necesite.

Sin embargo, el uso de la técnica de polling únicamente requiere algunas llamadas a la API Win32, pudiéndose usar lenguajes de más alto nivel que permitan hacer este tipo de llamadas. Por ello se ha escogido C# como lenguaje haciendo uso de la plataforma de .NET, pudiendo realizar operaciones que no requieran interacción directa con el sistema operativo con funciones de alto nivel, y las que requieran interacción directa con funciones de bajo nivel. Las funciones de la API Win32 se pueden invocar, cuando proceda, desde C# gracias a los servicios de interoperabilidad.

Solamente hubiese valido la pena desarrollar el software con C o C++ si se hubiese elegido la técnica del hook de filtro de mensajes o la del "mirror driver" de gráficos en vista de que requieren una interacción a muy bajo nivel con el sistema operativo.

3.2. Análisis

Primeramente, en esta sección se explica qué metodología de desarrollo se ha aplicado sobre el proyecto. Más adelante, se definen los requisitos de usuario, los casos de uso y los requisitos de software asociados a los de usuario. Para finalizar, se incluyen las matrices de trazabilidad entre los distintos tipos de requisitos de usuario y de software.

3.2.1. Metodología de desarrollo

En este proyecto, se ha decidido seguir la metodología de desarrollo en cascada a causa de que las fases del desarrollo del mismo han sido secuenciales, es decir, no se ha comenzado una fase hasta que no finalizase por completo la anterior. Esta metodología también es adecuada porque siempre estuvieron claras las funcionalidades a implementar y el sistema no está dividido en módulos independientes. Su esquema se muestra en la Figura 3.2 .

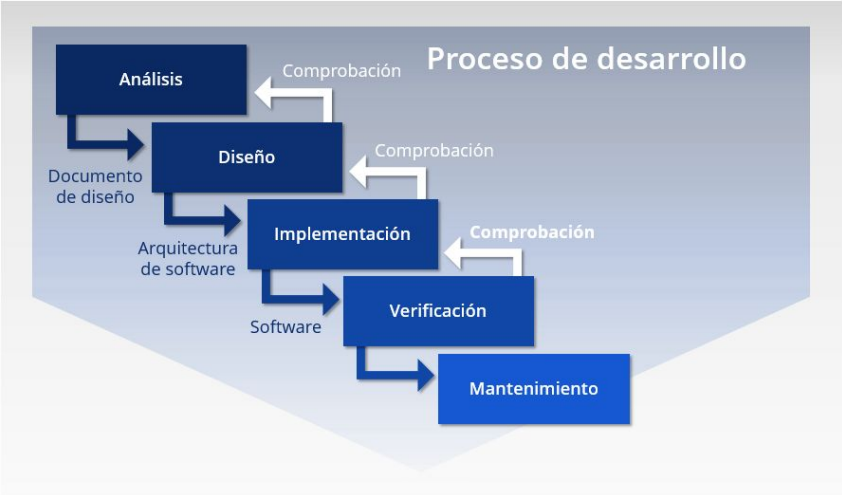


Figura 3.2: Metodología de desarrollo en cascada [28]

Dentro de cada fase, se realizan distintas tareas [28]:

- **Análisis**. En ella se realiza una definición detallada de los requisitos del sistema.
- **Diseño**. Se diseña la arquitectura del software en base a los requisitos definidos en la fase anterior.
- **Implementación**. Se implementa la arquitectura del software y se desarrollan los distintos componentes independientemente, verificando su funcionamiento con pruebas unitarias.
- **Verificación**. Se realiza la integración de las distintas partes (integración de sistemas) y también se hacen las pruebas del sistema todo integrado (pruebas de sistema y de integración).
- **Mantenimiento**. Se entrega el software, manteniéndolo y mejorándolo con el paso del tiempo.

3.2.2. Requisitos de usuario

Los requisitos de usuario especifican funcionalidades y restricciones del sistema final. Pueden ser de dos tipos:

- **Requisitos de capacidad**. Exponen las funcionalidades que el software debe tener.
- **Requisitos de restricción**. Indican restricciones sobre el cómo realiza el software sus funcionalidades.

Se establecen siguiendo la plantilla de la Figura 3.3. Sus campos se detallan a continuación:

- **Identificador**. Identifica unívocamente al requisito. Tendrá el formato "XX-UR-YY" donde *UR* especifica que es un requisito de usuario, *XX* podrá ser CA o RE para indicar que es un requisito de capacidad o restricción e *YY* corresponderá al número de secuencia empezando por 01.

- **Descripción**. Consiste en una explicación detallada del requisito.
- **Necesidad**. Se refiere a la prioridad del requisito desde la postura del cliente (esencial, conveniente u opcional).
- **Prioridad**. Hace referencia a la prioridad desde el punto de vista del desarrollador (alta, media o baja).
- **Estabilidad**. Especifica si el requisito puede variar durante las fases de desarrollo (no cambia, cambiante o muy inestable).
- **Verificabilidad**. Se refiere a la capacidad de comprobar que el requisito se pueda validar (alta, media o baja).

XX-UR-YY
Descripción:
Necesidad:
Prioridad:
Estabilidad:
Verificabilidad:

Figura 3.3: Plantilla de requisitos de usuario

Requisitos de capacidad

CA-UR-01	
Descripción:	Se podrá establecer una conexión con una máquina remota teniendo el control del escritorio de la misma.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-02	
Descripción:	Para comenzar el servicio ofrecido por el servidor es necesario especificar la dirección de la interfaz a la que estará asociado y si se aceptan conexiones seguras o no seguras.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-03

Descripción:	Para conectar el cliente con el servidor se debe detallar la dirección donde se está ejecutando este, si se desea una conexión segura o no segura y la contraseña generada previamente por el servidor.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-04

Descripción:	Se podrá elegir tener la transmisión de información en modo no seguro.
Necesidad:	Opcional
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Media

CA-UR-05

Descripción:	Se podrá elegir tener la transmisión de información en modo seguro.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

CA-UR-06

Descripción:	Se podrá elegir que la imagen del servidor se transmita de manera comprimida.
Necesidad:	Conveniente
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

CA-UR-07

Descripción:	Se podrá elegir que la imagen del servidor no se transmita de manera comprimida eligiendo la calidad de la misma.
Necesidad:	Opcional
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-08

Descripción:	Se podrá elegir la cantidad de fotogramas por segundo enviados por el servidor.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Baja

CA-UR-09

Descripción:	Se permitirá la conexión de múltiples clientes a la máquina servidora.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-10

Descripción:	El servidor mostrará el nombre de todos los clientes conectados.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	Cambiante
Verificabilidad:	Alta

CA-UR-11

Descripción:	La máquina servidora tendrá la posibilidad de deshabilitar el control de su ratón y teclado por parte de los clientes, permitiéndoles solamente visualizar el contenido de su pantalla.
Necesidad:	Opcional
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-12

Descripción:	La imagen recibida del escritorio del servidor siempre será adaptada al tamaño de la pantalla del cliente, pudiéndose hacer esta más pequeña hasta llegar a un tamaño límite en el que por debajo de este, el contenido de la imagen dejaría de ser apreciable.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-13

Descripción:	En conexiones seguras tanto la parte cliente como la servidora tendrán que autenticarse.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-14

Descripción:	En conexiones seguras, si hay errores en la autenticación de la parte cliente o la servidora, se avisará al usuario pudiendo este decidir si continuar o no.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-15

Descripción:	Las fallos de la sesión de escritorio remoto se manejarán de forma segura.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Media

Requisitos de restricción

RE-UR-01

Descripción:	La cantidad de fotogramas por segundo recibidos del servidor será igual a 5, 10, 15, 20 o 25.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	Cambiante
Verificabilidad:	Baja

RE-UR-02

Descripción:	El programa se ejecutará en ordenadores con el sistema operativo de Microsoft Windows y el software de .NET instalados.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	Cambiante
Verificabilidad:	Alta

RE-UR-03

Descripción:	La contraseña generada por el servidor será una cadena de diez caracteres alfanuméricos.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	Cambiante
Verificabilidad:	Alta

3.2.3. Casos de uso

Un caso de uso expone el proceso que ha de seguir un agente externo para hacer uso de una funcionalidad del sistema. Los casos de uso se detallan siguiendo la plantilla de la Figura 3.4 y su representación gráfica se observa en el diagrama UML de la Figura 3.5. Los campos se explican a continuación:

- **Identificador**. Identifica unívocamente al caso de uso. Tendrá el formato "UR-XX" donde *XX* corresponderá al número de secuencia empezando por 01.
- **Nombre**. Título que describe brevemente el caso de uso.
- **Actores**. Entidad que ejecuta el caso de uso.
- **Objetivo**. Finalidad del caso de uso.
- **Descripción**. Procedimientos a seguir para ejecutar el caso de uso expresados en lenguaje natural.
- **Pre-condición**. Condiciones que han de cumplirse previamente a la ejecución del caso de uso.
- **Post-condición**. Condiciones que se satisfarán posteriormente a la ejecución del caso de uso.

UC-XX

Nombre:

Actores:

Objetivo:

Descripción:

Pre-condición:

Post-condición:

Figura 3.4: Plantilla de requisitos de casos de uso

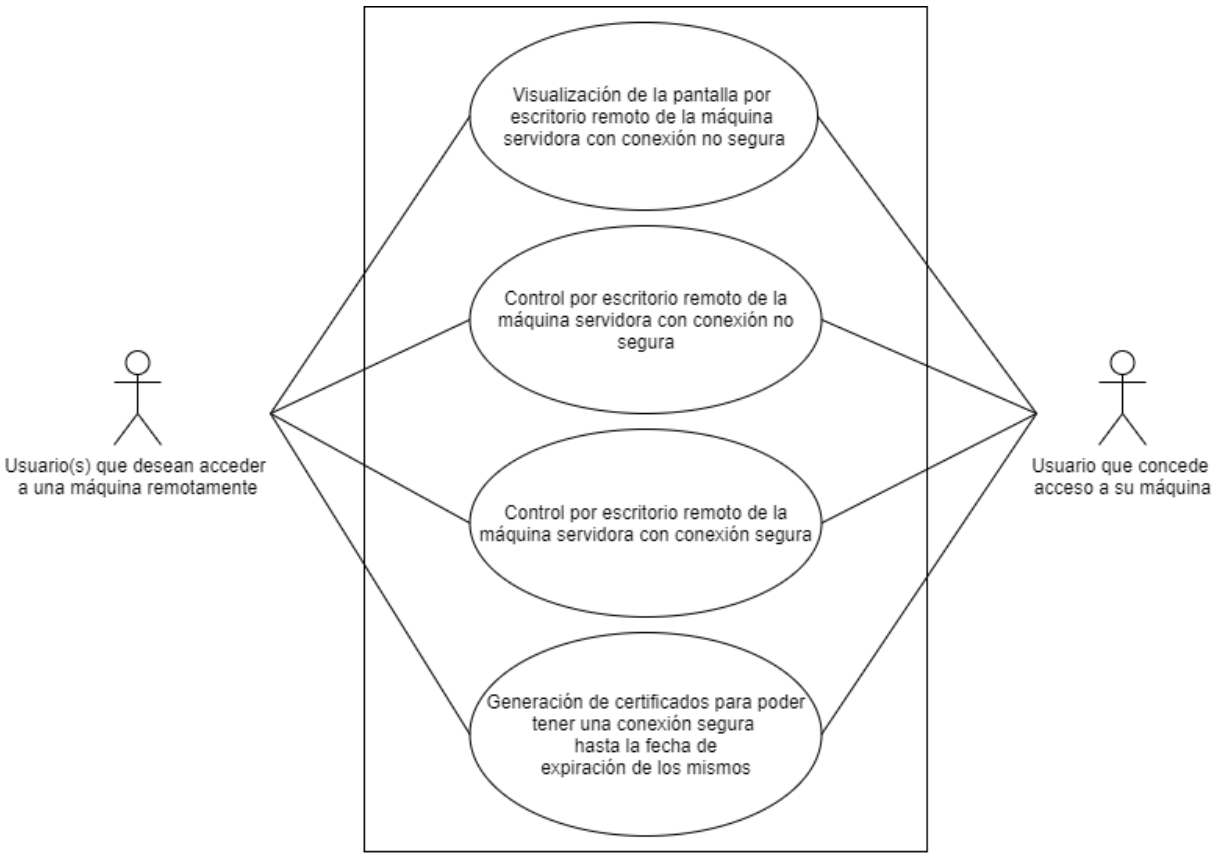


Figura 3.5: Diagrama de casos de uso

UC-01

Nombre:	Visualización de la pantalla por escritorio remoto de la máquina servidora con conexión no segura.
Actores:	Usuario o usuarios que desean acceder a una máquina remotamente y usuario que concede el acceso a su máquina.
Objetivo:	Visualización del escritorio de la máquina remota.
Descripción:	El usuario que quiere acceder a un equipo remotamente solicita conectarse a la máquina servidora para visualizar el escritorio y el que ofrece su equipo le concede el acceso.
Pre-condición:	El usuario que concede acceso a su equipo introduce una dirección IP y puerto al que conectarse, sin especificar que la conexión esté cifrada. El que solicita el acceso introduce la dirección IP y el puerto que ha especificado el usuario anterior, junto con la contraseña de autenticación; este tampoco detalla que la conexión esté cifrada.
Post-condición:	El usuario que quiere acceder a un máquina remota ve el escritorio del equipo del que concede el acceso.

UC-02

Nombre:	Control por escritorio remoto de la máquina servidora con conexión no segura.
Actores:	Usuario o usuarios que desean acceder a una máquina remotamente y usuario que concede el acceso a su máquina.
Objetivo:	Control del escritorio de la máquina remota con conexión en claro.
Descripción:	El usuario que quiere acceder a un equipo remotamente solicita conectarse a la máquina servidora para controlarla por escritorio remoto y el que ofrece su equipo le concede el acceso.
Pre-condición:	El usuario que concede acceso a su equipo introduce una dirección IP y puerto al que conectarse, sin especificar que la conexión esté cifrada. El que solicita el acceso introduce la dirección IP y el puerto que ha especificado el usuario anterior, junto con la contraseña de autenticación; este tampoco detalla que la conexión esté cifrada.
Post-condición:	El usuario que quiere acceder a un máquina remota controla el escritorio del equipo del que concede el acceso.

UC-03

Nombre:	Control por escritorio remoto de la máquina servidora con conexión segura.
Actores:	Usuario o usuarios que desean acceder a una máquina remotamente y usuario que concede el acceso a su máquina.
Objetivo:	Control del escritorio de la máquina remota con conexión cifrada.
Descripción:	El usuario que quiere acceder a un equipo remotamente solicita conectarse a la máquina servidora para controlarla por escritorio remoto y el que ofrece su equipo le concede el acceso.
Pre-condición:	El usuario que concede acceso a su equipo introduce una dirección IP y puerto al que conectarse, especificando que la conexión esté cifrada e introduciendo un certificado junto con su contraseña. El que solicita el acceso introduce la dirección IP y el puerto que ha especificado el usuario anterior, junto con la contraseña de autenticación y también especificando que la conexión esté cifrada. Este último también tiene que detallar un certificado, su contraseña y el nombre del servidor que espera que tenga este en su certificado.
Post-condición:	El usuario que quiere acceder a un máquina remota controla el escritorio del equipo del que concede el acceso.

UC-04

Nombre:	Generación de certificados para poder tener una conexión segura hasta la fecha de expiración de los mismos.
Actores:	Usuario o usuarios que desean acceder a una máquina remotamente y usuario que concede el acceso a su máquina.
Objetivo:	Generación de certificados.
Descripción:	Todos los usuarios que participen en una sesión de escritorio remoto generarán una solicitud de certificado que será verificada por una entidad de confianza (CA) que tengan instalada en su sistema. Después de verificarse se generará el correspondiente certificado.
Pre-condición:	Tener una entidad emisora de confianza instalada en el sistema.
Post-condición:	Los certificados están listos para utilizarse en una conexión de escritorio remoto cifrada.

3.2.4. Requisitos de software

Los requisitos de software se han obtenido a partir de los requisitos de usuario expuestos en la Subsección 3.2.2 y detallan de forma precisa lo que el analista ha interpretado de ellos. Se diferencian dos tipos:

- **Requisitos funcionales**. Aportan las funcionalidades del software.
- **Requisitos no funcionales**. Brindan características adicionales que no añaden nuevas funcionalidades.

Se describen siguiendo la plantilla de la Figura 3.6, sus campos son los siguientes:

- **Identificador**. Identifica unívocamente al requisito. Tendrá el formato "XX-SR-YY" donde *SR* especifica que es un requisito de software, *XX* podrá ser F o NF para indicar que es un requisito funcional o no funcional e *YY* corresponderá al número de secuencia empezando por 01.
- **Descripción**. Consiste en una explicación detallada del requisito.
- **Necesidad**. Se refiere a la prioridad del requisito desde la postura del cliente (esencial, conveniente u opcional).
- **Prioridad**. Hace referencia a la prioridad desde el punto de vista del desarrollador (alta, media o baja).

- **Estabilidad**. Especifica si el requisito puede variar durante las fases de desarrollo (no cambia, cambiante o muy inestable).
- **Verificabilidad**. Se refiere a la capacidad de comprobar que el requisito se pueda validar (alta, media o baja).
- **Origen**. Señala qué requisitos de usuario ocasionaron la aparición de este.

XX-SR-YY
Descripción:
Necesidad:
Prioridad:
Estabilidad:
Verificabilidad:
Origen:

Figura 3.6: Plantilla de requisitos de software

Requisitos funcionales

F-SR-01	
Descripción:	Se podrá establecer una conexión de control por escritorio remoto con una máquina, que previamente esté ejecutando la aplicación servidora.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-01

F-SR-02	
Descripción:	Para comenzar el servicio ofrecido por el servidor se tiene que introducir como mínimo una dirección IP con un puerto y especificar si se aceptan conexiones seguras (precisando un certificado y su contraseña) o no seguras.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-02

F-SR-03

Descripción:	Para conectar el cliente con el servidor es obligatorio introducir como mínimo una dirección IP con un puerto, especificar si se desea una conexión segura (precisando un certificado, su contraseña y el nombre esperado del servidor en su certificado) o no segura y también la contraseña para autenticarse generada previamente por el servidor.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-03

F-SR-04

Descripción:	Las conexiones podrán ser no seguras transmitiéndose la información en claro.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	CA-UR-04

F-SR-05

Descripción:	Las conexiones podrán ser seguras transmitiéndose la información cifrada usando SSL/TLS con certificados X.509.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Media
Origen:	CA-UR-05

F-SR-06

Descripción:	Se podrá elegir que la imagen recibida del servidor se transmita comprimida con el algoritmo JPEG.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	CA-UR-06

F-SR-07

Descripción:	Se podrá elegir que la imagen recibida del servidor no se transmita de manera comprimida eligiendo una profundidad de color concreta en bpp (bits por píxel).
Necesidad:	Conveniente
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-07

F-SR-08

Descripción:	Se capturan imágenes completas del escritorio, sin detección de cambios (polling), donde el usuario podrá elegir la cantidad de fotogramas por segundo recibidos del servidor, modificando la frecuencia del temporizador que captura el contenido de la pantalla.
Necesidad:	Conveniente
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Baja
Origen:	CA-UR-08

F-SR-09

Descripción:	El programa servidor permitirá a múltiples clientes conectarse y controlar la máquina en la que se está ejecutando.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	Cambiante
Verificabilidad:	Alta
Origen:	CA-UR-09

F-SR-10

Descripción:	El programa servidor mostrará el nombre de todos los clientes conectados en una lista en su ventana principal.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	Cambiante
Verificabilidad:	Alta
Origen:	CA-UR-10

F-SR-11

Descripción:	El programa servidor podrá ignorar las órdenes recibidas de los clientes a través de un botón en la interfaz habilitado para ello, permitiéndoles solamente recibir el estado de su escritorio.
Necesidad:	Opcional
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-11

F-SR-12

Descripción:	La imagen recibida por el servidor podrá ser escalada en el lado del cliente, nunca sobrepasando las dimensiones físicas de la pantalla y también manteniendo la relación de aspecto. El tamaño mínimo será de 960 píxeles de ancho y 540 de alto.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-12

F-SR-13

Descripción:	En conexiones cifradas tanto la parte cliente como la servidora tendrán que autenticarse mediante un certificado X.509.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-13

F-SR-14

Descripción:	En conexiones cifradas, si hay errores en el proceso de autenticación de la parte cliente o la servidora, se avisará al usuario mediante un cuadro de diálogo pudiendo este decidir si continuar o no.
Necesidad:	Esencial
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-14

F-SR-15

Descripción:	Las fallos de la comunicación de la parte cliente y la servidora se manejarán de forma segura avisando al usuario del error mediante un cuadro de diálogo.
Necesidad:	Conveniente
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	CA-UR-15

Requisitos no funcionales

NF-SR-01

Descripción:	La frecuencia del temporizador que captura el contenido del escritorio del servidor será igual a un valor de 200, 100, 66, 50 o 40 ms, correspondiendo estos valores a una tasa de 5, 10, 15, 20 o 25 FPS respectivamente.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	Cambiante
Verificabilidad:	Baja
Origen:	RE-UR-01

NF-SR-02

Descripción:	El programa se ejecutará en ordenadores con el sistema operativo de Microsoft Windows y la plataforma de .NET 5.0 o compatibles instalados.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	Cambiante
Verificabilidad:	Alta
Origen:	RE-UR-02

NF-SR-03

Descripción:	La contraseña generada por el servidor será una cadena de diez caracteres alfanuméricos, construida con un algoritmo que asegure tener un número entre dos y cuatro caracteres de distinto tipo (numéricos, letras minúsculas y letras mayúsculas) distribuidos aleatoriamente.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	Cambiante
Verificabilidad:	Alta
Origen:	RE-UR-03

3.2.5. Matrices de trazabilidad

En esta subsección se pueden contemplar las distintas matrices de trazabilidad. La Figura 3.7 muestra la matriz de trazabilidad entre requisitos de capacidad con requisitos funcionales, mientras que la Figura 3.8 refleja la de requisitos de restricción con requisitos no funcionales.

	CA-UR-01	CA-UR-02	CA-UR-03	CA-UR-04	CA-UR-05	CA-UR-06	CA-UR-07	CA-UR-08	CA-UR-09	CA-UR-10	CA-UR-11	CA-UR-12	CA-UR-13	CA-UR-14	CA-UR-15
F-SR-01	•														
F-SR-02		•													
F-SR-03			•												
F-SR-04				•											
F-SR-05					•										
F-SR-06						•									
F-SR-07							•								
F-SR-08								•							
F-SR-09									•						
F-SR-10										•					
F-SR-11											•				
F-SR-12												•			
F-SR-13													•		
F-SR-14														•	
F-SR-15															•

Figura 3.7: Matriz de trazabilidad requisitos de capacidad/requisitos funcionales

	RE-UR-01	RE-UR-02	RE-UR-03
NF-SR-01	•		
NF-SR-02		•	
NF-SR-03			•

Figura 3.8: Matriz de trazabilidad requisitos de restricción/requisitos no funcionales

3.3. Diseño

En la presente sección se detallan los distintos diagramas del software desarrollado, estos son los de clases, los de componentes y los de secuencia. Seguidamente, se muestra la matriz de trazabilidad entre los componentes y los requisitos de software funcionales. Adicionalmente, se explica en profundidad el protocolo que se ha implementado para la comunicación de los dos componentes principales, y finalmente se describe los distintos roles en el marco de PKI para la aplicación desarrollada.

3.3.1. Diagrama de clases

El diagrama de clases definido se muestra en la Figura 3.9. Para facilitar la lectura, se han omitido los parámetros de las funciones, pudiendo de esta manera mostrarse el diagrama entero en una imagen con un tamaño adecuado para su visualización. Para ver cada clase en detalle, con los parámetros de sus métodos, dirigirse al Apéndice D.

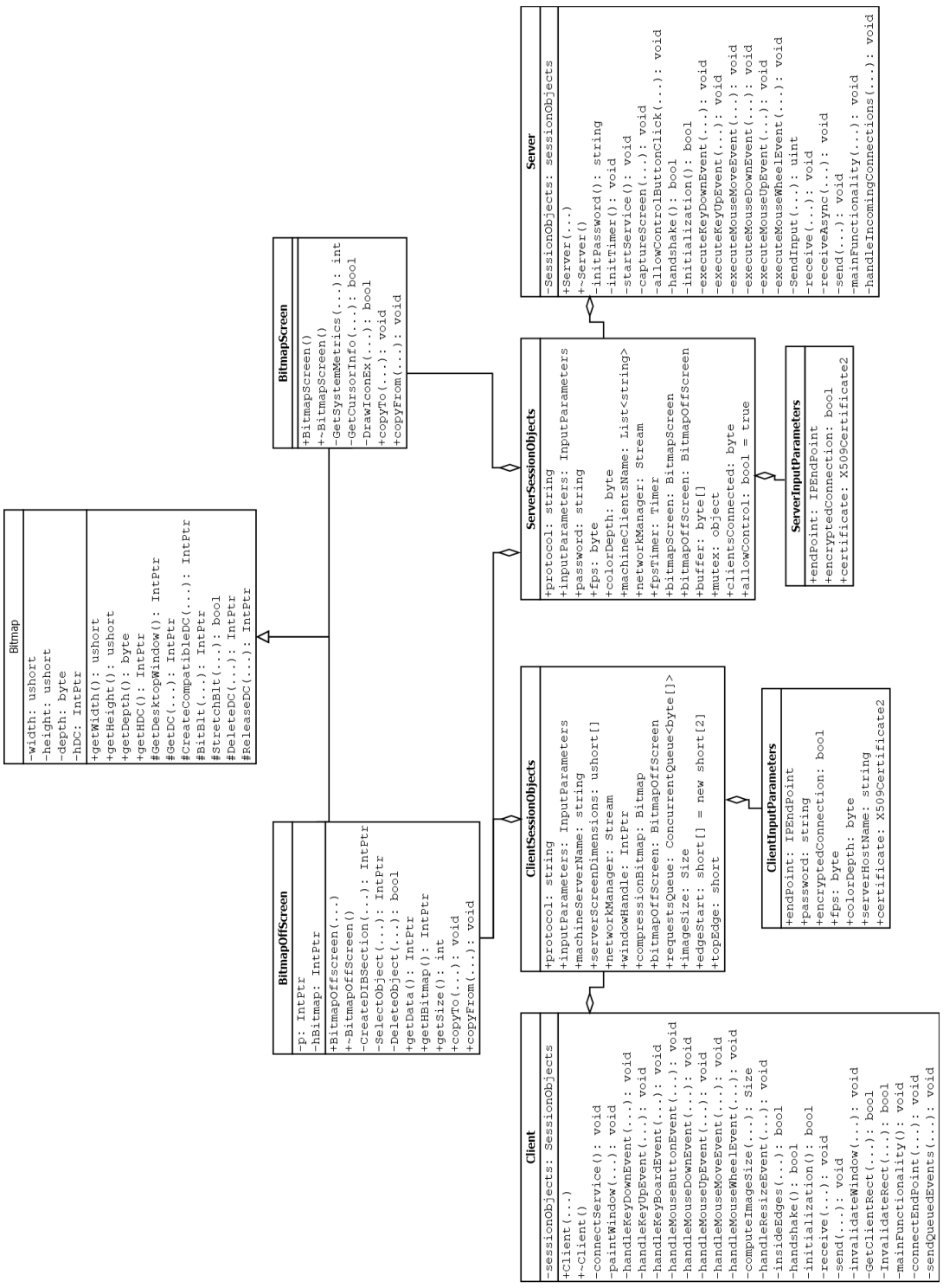


Figura 3.9: Diagrama de clases

Se ha decidido optar por elaborar un diseño lo más simple y eficiente posible, y también sencillo de entender. Todas las funciones principales del programa recaen sobre las clases Client y Server. La gestión de mapas de bits para el cliente y el servidor, y las operaciones asociadas queda encapsulada en las clases BitmapScreen o BitmapOffScreen, según el caso.
Las clases SessionObjects e InputParameters agrupan y encapsulan todos los objetos requeridos para la operación de las clases Client y Server. De esta manera, la estructura InputParameters

contiene parámetros especificados por el usuario, mientras que SessionObjects agrupa aquellos objetos que forman parte del estado de la aplicación.

A continuación, se explican las distintas funcionalidades de las cinco clases principales:

- **Bitmap**. Esta clase encapsulará todas las operaciones de bajo nivel que se puedan realizar sobre el contenido de un mapa de bits sin importar del tipo que sea. Todo mapa de bits tiene un ancho, alto, profundidad de color (en bits por píxel) y un manejador de contexto de dispositivo de GDI (véase Apéndice E) para operar con él.

- **BitmapScreen**. Hereda de la clase Bitmap y representa el mapa de bits asociado con la ventana raíz del sistema de ventanas (el escritorio). La implementación permite copiar el contenido de este mapa de bits a un mapa de bits off-screen.

- **BitmapOffScreen**. También hereda de la clase Bitmap, y representa un mapa de bits en memoria (no asociado a un área visible), cuya representación es independiente de dispositivo. La copia desde cualquier otro mapa de bits, implica la tradución de la representación hacia un forma independiente de dispositivo que puede ser transportada más allá de los límites de la máquina. De igual manera, la copia desde un BitmapOffScreen a un BitmapScreen podrá implicar una traducción a una representación dependiente de dispositivo. En general, estas operaciones se realizan internamente por GDI y son transparentes a la implementación de esta clase.

- **Client**. Su funcionalidad consiste en conectarse al servicio que va a ofrecer la clase Server. Después de conectarse, se encargará de recibir imágenes de lo que está ocurriendo en la máquina servidora, descomprimiendo (opcionalmente), y copiando éstos datos a una instancia de BitmapOffScreen. Finalmente, la representación intermedia provista por BitmapOffScreen es copiada a la ventana cliente. También suministrará la posibilidad al usuario de enviar órdenes con el fin de ejecutarlas en el equipo remoto.

- **Server**. Se encargará de ofrecer un servicio al que se podrá conectar cualquier máquina. Una vez comience una conexión con un cliente, enviará el estado del escritorio periódicamente, id est, la imagen del escritorio será copiada a una instancia de BitmapOffScreen, cuyo contenido podrá ser comprimido (opcionalmente), y enviado al extremo cliente. También iniciará una espera asíncrona de posibles órdenes de la(s) máquina(s) que se le conecten con el objeto de procesarlas y ejecutarlas.

Este diseño es adecuado para construir la aplicación debido a que es modular, separándose cada funcionalidad en una clase distinta. Un diseño con menos clases implicaría no distinguir tipos de mapas de bits, observándose que tiene sentido tener dos tipos distintos con funcionalidades diferentes pero complementarias. También se podría observar una sobrecarga de atributos en las clases principales Client y Server, no teniendo claro si los especifica el usuario o si se usan exclusivamente para un procesamiento interno.
Un mayor número de clases sería también erróneo debido a que el diseño sufriría particiones de una misma funcionalidad en varias y no tendría sentido. Lo complicaría más conceptualmente y sería ineficiente.

3.3.2. Diagrama de componentes

En la Figura 3.10 se puede observar el diagrama general de los componentes de la aplicación. En las Figuras 3.11 y 3.12 se muestra una vista detallada de los componentes Cliente y Servidor revelando su estructura interna. Esto se ha realizado de esta manera porque conceptualmente tanto el Cliente como el Servidor tienen dos partes diferenciadas, una es la que envía/recibe peticiones y otra es la que envía/recibe el estado del escritorio.

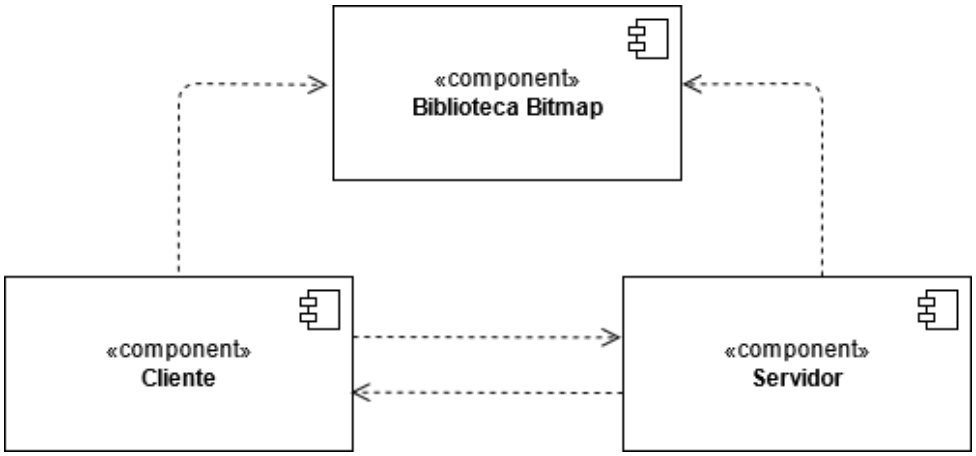


Figura 3.10: Diagrama de componentes general

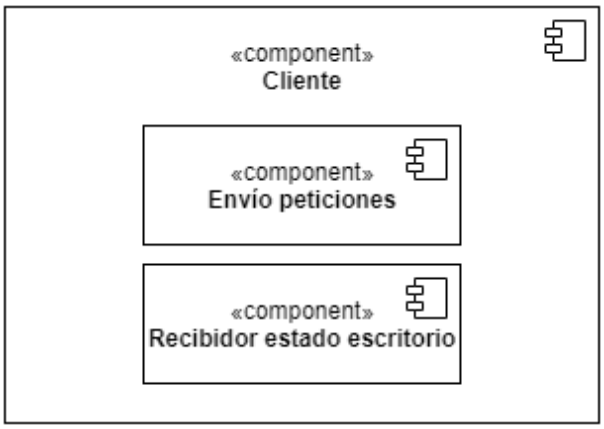


Figura 3.11: Diagrama de subcomponentes Cliente

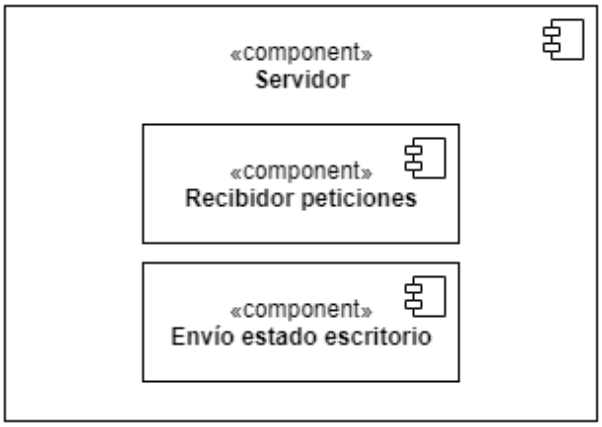


Figura 3.12: Diagrama de subcomponentes Servidor

Los componentes se detallan siguiendo la plantilla de la Figura 3.13. Sus campos se explican a continuación:

- **Identificador**. Corresponde al nombre del componente.
- **Rol**. Define la labor a realizar del componente.
- **Dependencias**. Refleja los componentes que dependan de este.
- **Descripción**. Explicación detallada de todas las funciones que realiza el componente.
- **Datos**. Valores que se deben especificar como entrada o que serán considerados como salida del componente.
- **Recursos**. Expone los recursos externos que necesita el componente para funcionar.
- **Origen**. Señala qué requisitos de software ocasionaron la aparición de este componente.

Identificador

Rol:

Dependencias:

Descripción:

Datos: n/a

Recursos: n/a

Origen:

Figura 3.13: Plantilla de requisitos de software

Biblioteca bitmap

Rol: Se encarga de ofrecer funcionalidades comunes de operaciones sobre mapas de bits a los otros componentes.

Dependencias: Cliente, Servidor

Descripción: Ofrece funcionalidades que usarán el Servidor y el Cliente, por ejemplo, obtener propiedades de un mapa de bits como su tamaño, y también copiar información de un mapa de bits a otro.

Datos:

- [Entrada]: Ninguno.
- [Salida]: Implementación de servicios para operar con mapas de bits en pantalla (BitmapScreen). Mapas de bits en memoria (BitmapOffScreen). Copia de información entre ellos.

Recursos:

- Win32
- .NET 5.0

Origen: F-SR-01, F-SR-06, F-SR-07

Cliente

Rol: Enviar órdenes y recibir estado del escritorio.

Dependencias: Servidor

Descripción: Se ocupa de ofrecer la función de enviar órdenes a través del teclado o del ratón y recibir el estado del escritorio del Servidor.

Datos:

- [Entrada]: Array de bytes conteniendo la imagen del escritorio del Servidor.
- [Salida]: Array de bytes conteniendo las órdenes para ejecutar en el Servidor.

Recursos:

- Win32
- .NET 5.0

Origen: F-SR-01, F-SR-03, F-SR-04, F-SR-05, F-SR-06, F-SR-07, F-SR-08, F-SR-12, F-SR-13, F-SR-14, F-SR-15

Servidor

Rol: Recibir órdenes y enviar estado del escritorio.

Dependencias: Cliente

Descripción: Su responsabilidad es recibir las posibles órdenes del Cliente y enviar el estado de su escritorio habiendo o no recibido órdenes.

Datos:

- [Entrada]: Array de bytes conteniendo las órdenes para ejecutar del Cliente.
- [Salida]: Array de bytes conteniendo la imagen del escritorio.

Recursos:

- Win32
- .NET 5.0

Origen: F-SR-01, F-SR-02, F-SR-04, F-SR-05, F-SR-09, F-SR-10, F-SR-11, F-SR-13, F-SR-14, F-SR-15

3.3.3. Diagrama de secuencia

Los componentes principales, Cliente y Servidor, tienen varios hilos que se encargan de ejecutar sus distintas funciones. A continuación, se muestra en la Figura 3.14 el diagrama de secuencia del componente Cliente y en la Figura 3.15 el del componente Servidor. Para ver más detalles de la interacción de todos los hilos, ver Capítulo 4.

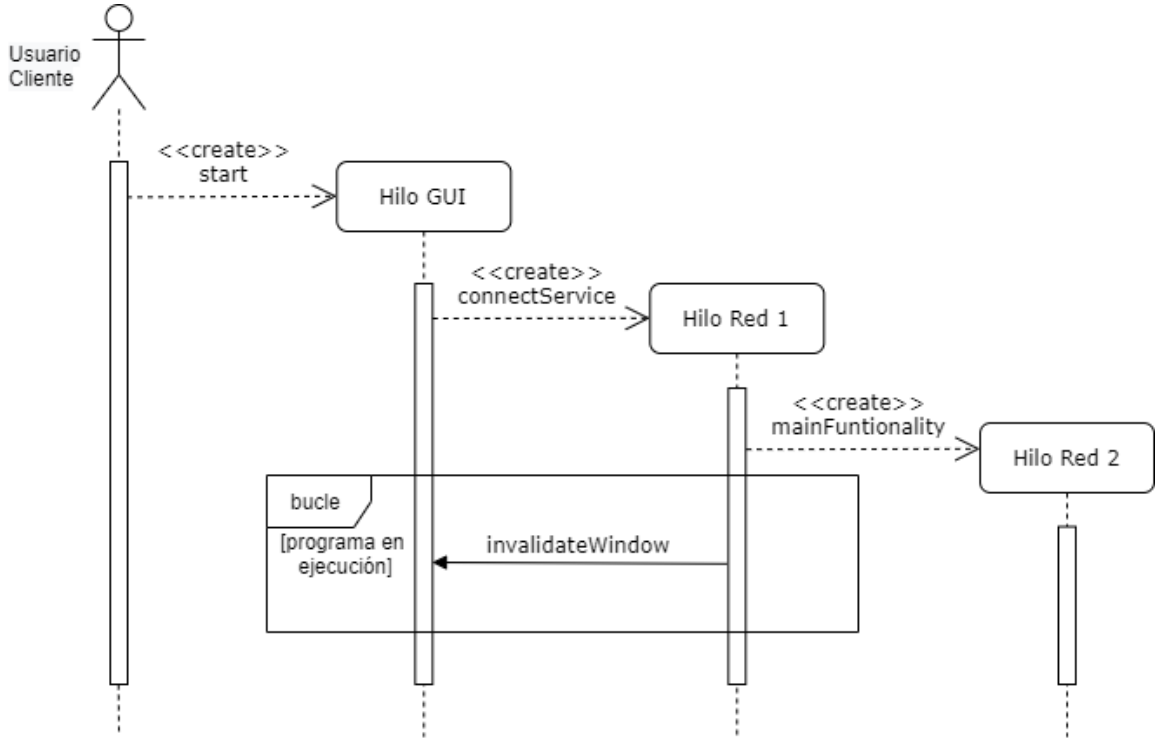


Figura 3.14: Diagrama de secuencia Cliente

El usuario de la máquina cliente crea el hilo GUI al ejecutar la aplicación. A su vez, este último para conectarse al servicio ofrecido por el servidor crea el hilo de red 1, que se encarga de recibir el estado del escritorio de la máquina remota. Dicho hilo de red, al iniciar su ejecución y entrar en la fase de funcionamiento normal del protocolo, crea el hilo de red 2, el cual se encargá de comunicar las peticiones al servidor. Después de crearlo, cada vez que reciba datos del servidor se comunicará con el hilo GUI con el objetivo de invalidar los datos que muestra y que este los remplace por los recibidos. Dicha invalidación es capaz de realizarla con su método *invalidateWindow*.

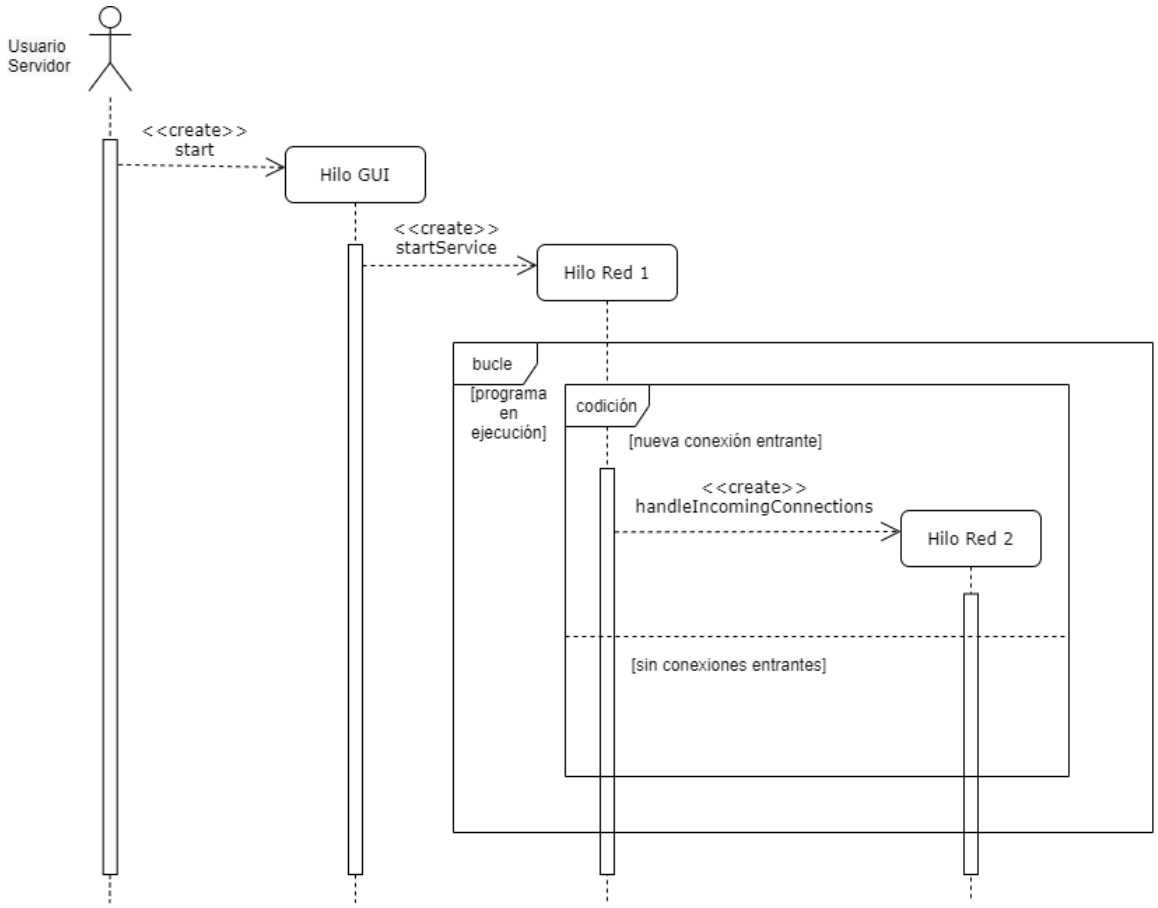


Figura 3.15: Diagrama de secuencia Servidor

El hilo GUI es creado por el usuario de la máquina servidora al ejecutar la aplicación. Para empezar a ofrecer el servicio, este hilo crea el hilo de red 1, que se encargará de comenzar a escuchar en la dirección correspondiente y manejará toda conexión entrante detectada mientras el programa esté en ejecución a través de su función *handleIncomingConnections*. Cada vez que se detecte una nueva conexión entrante se creará un hilo de red que se encargará de prestarle servicio al cliente recién conectado enviándole el estado del escritorio.

3.3.4. Matriz de trazabilidad

En esta subsección se contempla la matriz de trazabilidad entre componentes y requisitos de software funcionales. Dicha matriz es la de la Figura 3.16.

	F-SR-01	F-SR-02	F-SR-03	F-SR-04	F-SR-05	F-SR-06	F-SR-07	F-SR-08	F-SR-09	F-SR-10	F-SR-11	F-SR-12	F-SR-13	F-SR-14	F-SR-15
Biblioteca bitmap	•					•	•								
Cliente	•		•	•	•	•	•	•				•	•	•	•
Servidor	•	•		•	•				•	•	•		•	•	•

Figura 3.16: Matriz de trazabilidad componentes/requisitos funcionales

3.3.5. Protocolo de la aplicación

Con respecto al protocolo que utilizará la aplicación, hay dos opciones:

- **Diseño de un protocolo propio**. De esta manera, la aplicación puede utilizar un protocolo sencillo que solo realice las transmisiones necesarias de datos para las tareas que se quieran implementar.
- **Utilizar un protocolo existente**. Esta alternativa consiste en implementar un protocolo ya existente que emplee una aplicación de escritorio remoto como RFB para VNC y sus variantes. Brindaría la posibilidad de sustituir una de las partes del programa (cliente o servidor) por otra de otro programa completamente distinto y que su uso en conjunto fuese posible.

Se ha escogido crear un protocolo propio por simplicidad a la hora de la implementación y porque el objetivo principal de la aplicación no es que se comunique con otras ya existentes, sino que sea sencilla de entender. A continuación, se exponen los detalles del protocolo diseñado:

Identificación y características del cliente y del servidor

Los actores en el protocolo podrán tener dos roles:

- **Cliente**. El equipo en el que se encuentra el usuario que realiza todas las acciones tomará el rol de cliente. Su función será enviar peticiones de acciones de ratón o de teclado al servidor y recibir los cambios de su escritorio, envíe o no este peticiones.
- **Servidor**. Dicho rol pertenecerá al equipo que comparte su escritorio, y por tanto en el que se ejecutan las acciones demandadas por el cliente. Se ocupa de procesar las peticiones recibidas del cliente, ejecutándolas, y también de enviar el estado actual de su escritorio, envíe o no este último dichas peticiones.

Es importante mencionar que el cliente será secuencial debido a que siempre se conectará a un único servidor para enviarle órdenes. El servidor será concurrente permitiendo de esta manera la conexión de múltiples clientes pudiendo manejar o ver su escritorio.

El tipo de conexión que se ha seleccionado es TCP en vista de que se busca la fiabilidad de la conexión, y que las tramas perdidas sean automáticamente retransmitidas por la pila de red. UDP no garantiza la entrega de paquetes, por eso se ha evitado el uso de este protocolo. Para saber más de estos protocolos ver Apéndice F.

Las dirección IP y el puerto de escucha del servidor los elegirá el usuario que se encuentre físicamente en frente de esa máquina. Desde el punto de vista del cliente, conectarse al servicio ofrecido por el servidor, consistirá en la especificacón de dicha dirección y puerto.

Intercambio de mensajes

Dependiendo de la fase en la que se encuentre el protocolo, se intercambiarán distintos tipos de mensajes. Las fases son las siguientes:

- **Negociación**

  Es la primera fase del protocolo. En ella el objetivo es acordar qué versión del protocolo se usará. Comienza con un mensaje enviado por el servidor en el que declara cuál es la versión más alta del protocolo que soporta. El cliente responderá con la versión del protocolo que se utilizará, siendo esta igual o menor que la recibida pero nunca mayor, y también le enviará una contraseña como medio de autenticación. Tanto si la contraseña es correcta, como si no, el servidor mandará un mensaje al cliente comunicándoselo; en caso de que sea incorrecta no se continuará con las siguientes fases. El diagrama de dichos mensajes se puede observar en la Figura 3.17.

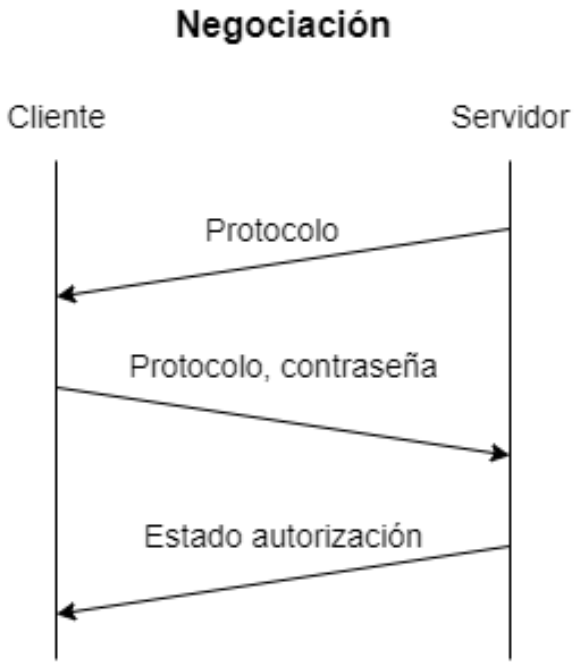


Figura 3.17: Protocolo fase negociación

- **Inicialización**

  Después de que el cliente y el servidor acuerden la versión del protocolo y en caso de que la contraseña enviada por el cliente sea la que haya generado el servidor, comenzará la fase de inicialización. En ella, el servidor enviará un mensaje con la longitud del nombre de su máquina, el propio nombre y las dimensiones de su escritorio. El cliente responderá con la longitud del nombre de su dispositivo, el nombre, y también los FPS y la profundidad de color que haya seleccionado el usuario. El diagrama de la Figura 3.18 muestra una representación gráfica de lo explicado.

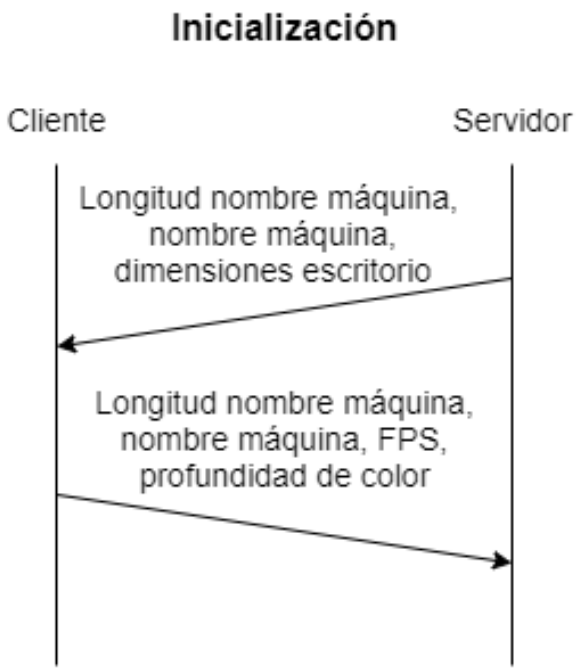


Figura 3.18: Protocolo fase inicialización

- **Funcionamiento normal**

  Una vez ya intercambiados los datos de inicialización, comenzará la tercera y última fase del protocolo. Consistirá en el posible envío de un mensaje de petición del cliente al servidor y en el envío estricto del estado del escritorio (tamaño de la imagen + imagen) del servidor al cliente hasta que finalice la ejecución del programa. En la Figura 3.19 se describe lo expuesto en este párrafo.

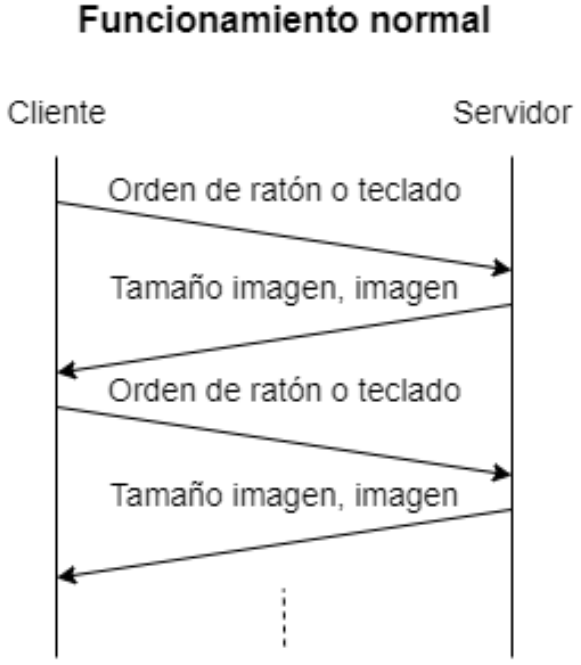


Figura 3.19: Protocolo fase funcionamiento normal

Formato de los mensajes

- **Contenido de los mensajes de la etapa de negociación**

  - **Versión del protocolo**

    Es un mensaje que se envían mutuamente el cliente y el servidor. Contendrá nueve bytes y será una cadena de texto con el formato "RDE xx.yy" donde xx e yy corresponden a los números de versión mayor y menor del protocolo (con ceros a la izquierda si su versión no tiene dos cifras).

No. de bytes	Tipo [Valor]	Descripción
9	U8 array	versión-protolo

  - **Contraseña**

    Enviada por el cliente al servidor para autenticarse. Será una cadena de texto alfanumérica de diez caracteres y su estructura es la siguiente:

No. de bytes	Tipo [Valor]	Descripción
10	U8 array	contraseña

  - **Resultado de validación de contraseña**

    Enviado por el servidor al cliente para comunicarle si la contraseña de autenticación recibida es correcta o no. Su estructura es la siguiente:

No. de bytes	Tipo [Valor]	Descripción
1	U8	validación-contraseña

    La codificación de dicho byte se muestra a continuación:

Valor	Descripción
0	contraseña-incorrecta
1	contraseña-correcta

- **Contenido de los mensajes de la etapa de inicialización**

  - **Longitud nombre máquina**

    Lo envían el cliente y el servidor. Se usará para saber cuántos bytes se recibirán codificando el nombre de máquina. Tiene la siguiente estructura:

No. de bytes	Tipo [Valor]	Descripción
1	U8	longitud-nombre-máquina

  - **Nombre máquina**

    Al igual que el anterior, es enviado por el cliente y el servidor. El objetivo es que ambos extremos conozcan sus nombres de máquina. Esto es especialmente relevante para el servidor, que deberá mostrar una lista de los nombres de los clientes conectados. Consta de la siguiente estructura:

No. de bytes	Tipo [Valor]	Descripción
16	U8 array	nombre-máquina

  - **Dimensiones escritorio**

    Enviado por el servidor al cliente para que este pueda, entre otros, reservar la memoria necesaria con la finalidad de poder recibir correctamente todos los datos que le va a enviar el servidor. Posee la siguiente estructura:

No. de bytes	Tipo [Valor]	Descripción
2	U16	píxeles-ancho-pantalla
2	U16	píxeles-alto-pantalla

  - **FPS**

    Lo remite el cliente definiendo de esta manera la frecuencia de actualización de imágenes que va a enviarle el servidor.

No. de bytes	Tipo [Valor]	Descripción
1	U8	fps

    Es importante comentar que en una conexión multicliente, los FPS los fijará el primer cliente que realice la conexión, ignorando el servidor los valores que le envíen sus próximos consumidores. Este valor podrá ser 5, 10, 15, 20 o 25.

- **Profundidad de color**

  Enviado por el cliente especificando el número de bits por píxel usado para codificar las imágenes recibidas del servidor (en el caso de no usar compresión).

No. de bytes	Tipo [Valor]	Descripción
1	U8	profundidad-color

  Los valores que puede especificar el usuario explícitamente son 1, 4, 8, 16, 24 o 32. En caso de que no se precise ningún valor, este parámetro tomará el valor de 0, interpretando el servidor que el usuario quiere recibir las imágenes con compresión JPEG. Al igual que los FPS en una conexión multicliente, este valor también lo fijará el primer cliente.

- **Contenido de los mensajes de la etapa de funcionamiento normal**

  - **Petición**

    La envía el cliente para comunicarle al servidor que ejecute una acción determinada. Está formada por cinco bytes fijos con el siguiente significado:

No. de bytes	Tipo [Valor]	Descripción
1	U8	tipo-evento
4	U8 array	información-evento

    El primer byte codifica el tipo de evento, tendrá un valor distinto según el evento que se produzca:

Valor	Descripción
0	pulsación-tecla-teclado
1	liberación-tecla-teclado
2	movimiento-ratón
3	pulsación-botón-ratón
4	liberación-botón-ratón
5	rueda-ratón

    En caso de que el valor del primer byte sea 0 o 1, los siguientes dos bytes especificarán la tecla que se ha pulsado o liberado basándose en la codificación de teclas que usa Microsoft Windows [29]. Los últimos dos bytes no se utilizarán.

    En el supuesto de que el primer byte sea 2, los siguientes cuatro bytes determinan las coordenadas del puntero del ratón (x e y), mapeadas a las dimensiones de pantalla del servidor.

Si el valor del primer byte es 3 o 4, el siguiente byte codifica el botón del ratón que se ha pulsado o soltado. Los últimos 3 bytes no se usan. La codificación de los botones es la siguiente:

Valor	Descripción
0	botón-izquierdo
1	botón-derecho
2	botón-mitad
3	botón-X1
4	botón-X2

Por último, si el valor del primer byte es 5, los siguientes dos bytes especificarán el desplazamiento y la dirección de la rueda del ratón usando la codificación que usa Windows [30].

- **Imagen del escritorio**

  Enviada por el servidor al cliente. Contiene los datos de la imagen de su escritorio y su tamaño; tiene el siguiente formato:

No. de bytes	Tipo [Valor]	Descripción
4	U32	tamaño-imagen
tamaño-imagen	U8 array	datos-imagen

  La imagen recibida podrá estar comprimida en JPEG, o sin comprimir, dependiendo de si el usuario elige una profundidad de color específica o no. Este mensaje se enviará periódicamente, aun sin intervención del lado cliente, debido a que la técnica de detección de cambios y captura de imagen implementada es la de polling (ver Subsección 3.1.2 para más detalles).

3.3.6. Estructura PKI

La jerarquía de las entidades que existirán en la aplicación cuando la conexión sea segura, se muestra en la Figura 3.20. Para entender conceptos básicos de criptografía y la necesidad de una PKI, visitar el Apéndice G.

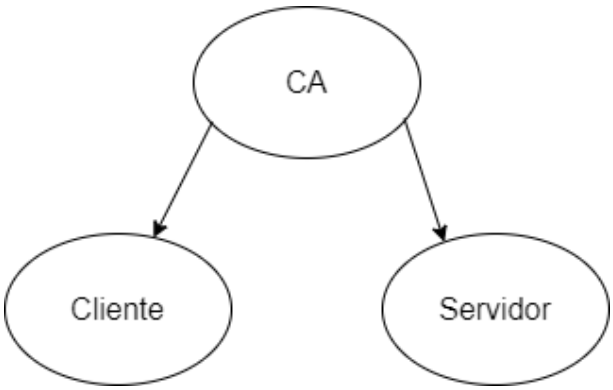


Figura 3.20: Estructura PKI

Es importante señalar que, en general, el servidor asumirá dos roles: el propio rol de Servidor que ofrece el servicio de escritorio remoto y el de CA que firma los certificados. El cliente solamente posee un rol, que es el mostrado como Cliente en la figura.

Capítulo 4

Implementación

En este capítulo se exponen los aspectos de implementación más importantes de los componentes del sistema en la Sección 4.1, así como los detalles de las herramientas que se han usado para llevar a cabo la misma en la Sección 4.2. Si no se está familiarizado con la API Win32, se recomienda la previa lectura del Apéndice E. El proceso de instalación y uso del sistema queda descrito en el manual de usuario, correspondiente al Apéndice C.

4.1. Detalles de implementación de los componentes

En esta sección se exponen los detalles de implementación más importantes acerca de los tres componentes de la aplicación.

4.1.1. Biblioteca bitmap

Este componente se ha implementado como una biblioteca dinámica debido a que posee definiciones de clases y funciones en común que usarán tanto el componente Cliente como el Servidor. Está compuesto por las siguientes clases:

Bitmap

Se ha implementado como una clase abstracta que, por definición, no puede ser instanciada, representa un mapa de bits genérico y tiene atributos y operaciones comunes a todo tipo de mapa de bits que existen en la aplicación (mapa de bits en memoria y mapa de bits de pantalla).

BitmapScreen

Al instanciarse, un objeto de esta clase mantendrá una referencia al contexto de dispositivo GDI (hDC) asociado con la pantalla primaria. El ancho, alto, y profundidad de color en bits-por-píxel corresponderán, por tanto, con aquéllos de la pantalla primaria. De este modo, cualquier operación de copia que tenga como origen este mapa de bits, realizará implícitamente una captura del área visible.

Posee la capacidad de copiar el contenido actual de la pantalla a otro mapa de bits que se le especifique, incluyendo el icono del ratón, así como de recibir el contenido de otro mapa de bits y mostrarlo en pantalla.

BitmapOffScreen

Se instancia especificándole un ancho y un alto en píxeles, conjuntamente con una profundidad de color determinada. En este proceso de instanciación se realiza lo siguiente:

- Se crea un mapa de bits independiente del dispositivo precisando que sus datos estarán dispuestos en memoria de arriba a abajo (bitmap top-down). Específicamente, el mapa de bits es de tipo DIB-section; este tipo de mapa de bits permite obtener un puntero a la región de datos del mismo.
- Creación de un contexto de dispositivo GDI (hDC) en memoria, que llevará asociado el mapa de bits creado en el paso anterior. De esta manera, una operación de copia llevará asociada una conversión implícita (por GDI) del mapa de bits a una representación independiente de dispositivo.

  Dado que el mapa de bits es de tipo DIB-section, se puede acceder directamente a los bytes del bitmap dado el puntero a la región de datos. Por tanto, su contenido es fácilmente accesible referenciando dicho puntero.

Al igual que la clase BitmapScreen, esta también es capaz de copiar su contenido a otro mapa de bits, además de recibir el contenido de otro mapa de bits y alojarlo en memoria previamente reservada. Estos métodos se diferencian de los de la clase BitmapScreen en que no tienen en cuenta el cursor del ratón y que el copiado de imagen (el contenido del mapa de bits) se realiza estirándola o comprimiéndola con el fin de que se respete la relación de aspecto original.

Por último, es necesario señalar que el tamaño en bytes de la imagen del mapa de bits se calculará usando la fórmula de la Ecuación 4.1

$$Tamaño = Ancho\,en\,píxeles \times Altura\,en\,píxeles \times \frac{Profundidad\,de\,color}{8} \tag{4.1}$$

4.1.2. Cliente

En la Figura 4.1 se muestra un esquema general de la implementación del tratado de la imagen del escritorio que realiza el cliente. Sus detalles se excplicarán a lo largo de esta subsección.

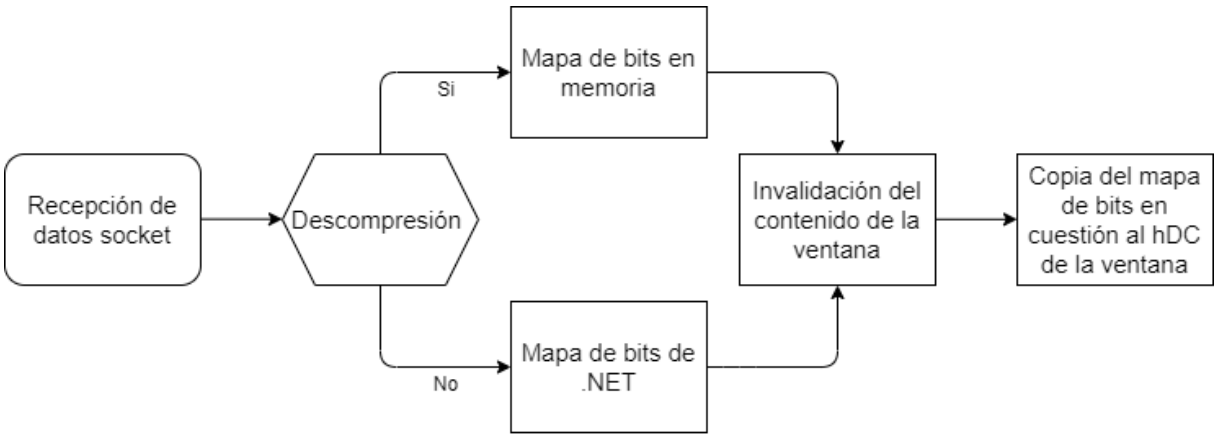


Figura 4.1: Esquema manejo recepción imagen - cliente

La clase en la que se ha implementado el componente Cliente tendrá un único atributo que encapsulará todos los objetos de sesión que serán necesarios en la ejecución del programa. Estos objetos se especifican a continuación:

- **Protocolo de la aplicación**. Cadena de caracteres codificando el protocolo y su versión.
- **Parámetros iniciales especificados por el usuario**.
  - **Dirección IP y puerto a conectarse**. Extremo final en el que el servidor ofrece su servicio.
  - **Contraseña de autenticación**. Necesaria para prevenir el acceso no autorizado al servicio del servidor.
  - **Indicador de si la conexión estará cifrada o no**. Codifica si los datos se van a transmitir cifrados o en claro.
  - ***FPS**. Frecuencia de actualización de las imágenes recibidas.
  - ***Profundidad de color**. Bits por píxeles que tendrán las imágenes recibidas.
  - **Nombre esperado en el certificado del servidor**. Nombre común (o nombre de máquina) que se espera que el servidor tenga en su certificado.
  - **Certificado del cliente**. Fichero PKCS #12 envolviendo un certificado X509 junto con su clave privada.
- **Nombre de máquina del servidor**. Usado para identificar al servidor de una manera sencilla cuando empieze la sesión.
- **Dimensiones de la pantalla de la máquina del servidor**. Información necesaria para que el cliente sepa el tamaño de las imágenes que va a recibir.

- **Manejador de red de la aplicación**. Flujo de datos perteneciente al socket que mantiene una conexión entre el cliente y el servidor. Permite las operaciones de envio/recepción de información.
- **Manejador de ventana que mostrará el contenido**. Necesario para operar con el contenido mostrado en la ventana de la aplicación.
- **Mapa de bits de .NET**. Necesario para alojar los datos recibidos del servidor si viajan comprimidos.
- **Mapa de bits en memoria**. Instancia de la clase BitmapOffScreen que se usará para alojar los datos recibidos del servidor en caso de que no viajen comprimidos.
- **Cola de peticiones**. Cola concurrente en la que se guardará cada evento producido por el usuario para mandarlo al servidor.
- **Tamaño de la imagen adaptada al tamaño de ventana**. Codifica el tamaño que debe tener la imagen recibida del servidor en la ventana que se va a mostrar, teniendo en cuenta las dimensiones actuales de dicha ventana.
- **Coordenadas de comienzo de area a dibujar**. Requeridas para centrar la imagen en la ventana donde se reciben los datos del servidor.
- **Tamaño del borde superior de la ventana con mensajes o acciones disponibles**. En el borde superior siempre habrá un botón que permitirá finalizar la sesión, y en caso de que el usuario no marque que la conexión sea segura, se mostrará una barra advirtiendo que no lo es. Este atributo codifica el tamaño reservado de los elementos que se van a mostrar en el borde superior.

La ejecución del cliente involucra la creación de algunos hilos adicionales. En particular, la implementación hace uso de tres hilos, cuyos puntos de entrada están implementados como métodos de la clase Client. Dichos hilos se detallan a continuación:

Hilo GUI

Este subproceso se encarga de las siguientes tareas:

- **Instanciar la clase**. En el constructor de la clase se asocian los parámetros iniciales que el usuario haya introducido a los objetos de sesión correspondientes. En caso de que no se hayan especificado algunos de carácter opcional (marcados con "*" anteriormente) se igualarán dichos objetos a un valor por defecto. En el supuesto de que la conexión no sea segura, se colocará una barra roja en la parte superior advirtiéndolo con un mensaje.
- **Conectarse al servicio ofrecido por el servidor**. Se inicia la ejecución del hilo de red responsable de recibir imágenes del escritorio del extremo servidor.
- **Repintar el contenido de la ventana cuando sea necesario**. Este método será llamado cuando el contenido actual de la ventana sea invalidado. El hilo de red responsable de

recibir imágenes del escritorio se encargará de invalidarlo cuando se reciban nuevos datos. La invalidación de la superficie de la ventana implica que el contenido sea redibujado.
En el caso de que se trabaje con compresión, el hilo que se acaba de mencionar, alojará los datos en el mapa de bits de .NET. Si no se trabaja con compresión los alojará en el mapa de bits en memoria. Por lo que para actualizar el contenido de la pantalla, se copiarán los datos de un mapa de bits u otro dependiendo de si se trabaja con compresión o no.

- **Manejar los eventos de teclado o de ratón producidos por el usuario**. Cada vez que se capture un evento se creará una estructura en la que se alojará la información necesaria para luego encolarla en la cola de peticiones. Se distinguen los siguientes eventos:

  - **Tecla de teclado**. A su vez, existen dos tipos, los cuales son la pulsación de una tecla y su liberación.
  - **Botón de ratón**. En este tipo de evento se verificará que el cursor del ratón no se encuentre en una posición donde no esté siendo pintada la imagen del servidor. Análogamente al anterior caso, habrá dos tipos diferenciando de esta manera, el pulsado y la liberación de los botones.
  - **Movimiento de ratón**. Al igual que el anterior, se comprobará también la posición del cursor del ratón. Posteriormente, se hará un mapeo de las coordenadas cliente del cursor dentro de la ventana a las coordenadas equivalentes en el servidor, teniendo en cuenta su resolución de pantalla.
  - **Rueda de ratón**. Su función es captar la dirección con la que se ha girado la rueda del ratón.

  Para revisar en detalle qué estructura se crea y qué datos de los eventos se introducen en ella, véase la Subsección 3.3.5.

- **Manejar el evento de reescalado de la ventana**. Cuando se reescale la ventana, se adaptará la imagen recibida del servidor al nuevo tamaño de ventana respetando siempre su relación de aspecto original.

Hilo de red responsable de recibir imágenes del escritorio

Este subproceso se encarga de las siguientes tareas:

- **Envío y recepción de datos**. Los métodos creados para estas funcionalidades son bloqueantes, es decir, no cesan su ejecución hasta enviar o recibir toda la información garantizando una correcta transmisión.

- **Invalidación de contenido**. El contenido de la ventana actual de la aplicación se debe de invalidar cuando se reciben nuevos datos.

- **Conectarse al extremo final dónde se ejecuta el servicio del servidor**. Se crea un socket con la dirección IP y puerto introducidos por el usuario, y se realiza una conexión a dicho extremo final. Seguidamente, se iguala el manejador de red al flujo de datos del socket. En el caso de que se haya seleccionado cifrar la conexión, se le añadirá una capa de

protección SSL al flujo de datos y se validará el certificado remoto del servidor, notificando al usuario con un cuadro de diálogo en caso de que haya problemas durante el proceso de validación. Si no se ha marcado que la conexión esté cifrada no se le añadirá ninguna capa de protección al flujo de datos.

- **Fases de la aplicación**. Como se ha explicado en la Subsección 3.3.5, la aplicación tiene tres fases, a continuación se explica los detalles más relevantes de la implementación de cada una de ellas:

  - **Negociación**. Se comprueba que los protocolos que hablan la aplicación cliente y la servidora sean los mismos y se envía la contraseña de autenticación al servidor, esperando luego una respuesta de confirmación por parte del mismo.
    En caso de que el protocolo no sea compatible o la contraseña sea incorrecta, no se continuará con las siguientes fases.
  - **Inicialización**. Se procederá al intercambio de información que permitirá inicializar los objetos de sesión necesarios para comenzar la sesión de escritorio remoto. Es importante destacar que si la negociación no se termina correctamente, el método que se encarga de la inicialización no realizará ninguna de sus operaciones.
  - **Funcionamiento normal**. Primeramente, si se ha elegido que la conexión esté cifrada se realiza la autenticación del cliente con el certificado que especificó el usuario. En segundo lugar, si las etapas de negociación e inicialización terminan con éxito, se procederá a crear los objetos necesarios para el funcionamiento normal del programa. Estos son la cola de peticiones concurrente y el bitmap en memoria para alojar los datos que se van a recibir; también se iniciará la ejecución del hilo de red encargado de enviar las peticiones al servidor.
    Después de realizar lo que se ha mencionado, el hilo entra en un bucle infinito en el que se recibirá el tamaño de la imagen del servidor y la propia imagen. En el caso de que se trabaje con compresión, se copiará dicha imagen en el mapa de bits de .NET. Si no se trabaja con ella, se copiará en el mapa de bits en memoria. Por último, después de realizar la copia de la imagen al mapa de bits en cuestión, se invalida el contenido actual de la ventana.

Hilo de red comunicador de peticiones

Este subproceso únicamente tendrá una funcionalidad, la cual consiste en un bucle infinito en el que se comprueba si en la cola concurrente de peticiones hay peticiones pendientes de enviar al servidor. En caso de que haya pendientes, se enviarán.

4.1.3. Servidor

El esquema general de la implementación del tratado de la imagen del escritorio que realiza el servidor se expone en la Figura 4.2. Sus detalles se describirán en esta subsección.

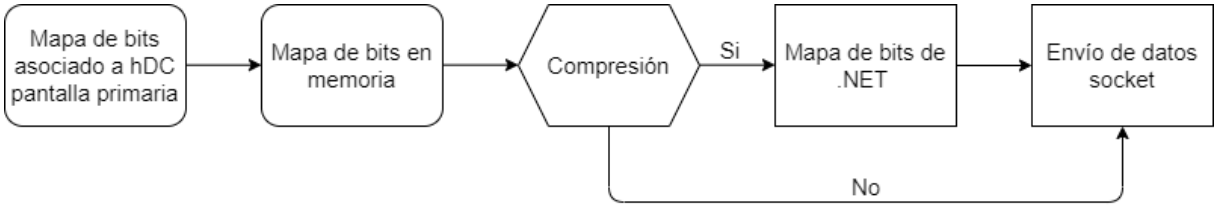


Figura 4.2: Esquema manejo envío imagen - servidor

Al igual que el componente Cliente, la clase del componente Servidor también tendrá un único atributo que encapsulará todos los objetos de sesión necesarios. Se detallan a continuación:

- **Protocolo de la aplicación**. Cadena de caracteres codificando el protocolo y su versión.
- **Parámetros iniciales especificados por el usuario**.
  - **Dirección IP y puerto a conectarse**. Extremo final en el que se va a ofrecer el servicio.
  - **Indicador de si la conexión está cifrada o no**. Codifica si los datos se van a transmitir cifrados o en claro.
  - **Certificado del servidor**. Fichero PKCS #12 envolviendo un certificado X509 junto con su clave privada.
- **Contraseña de autenticación**. Necesaria para prevenir el acceso no autorizado al servicio ofrecido.
- **FPS**. Frecuencia de envío de imágenes.
- **Profundidad de color**. Bits por píxeles que tendrán las imágenes a enviar.
- **Lista de nombres de las máquinas de los clientes conectados**. Usada para controlar qué clientes hay conectados.
- **Manejador de red de la aplicación**. Flujo de datos perteneciente al socket que mantiene una conexión entre el cliente y el servidor. Permite las operaciones de envío/recepción de información.
- **Temporizador para capturar la pantalla**. Mecanismo capaz de ejecutar una función determinada, en este caso capturar la pantalla, cada cierto tiempo.
- **Mapa de bits de la pantalla**. Instancia de la clase BitmapScreen necesaria para adquirir propiedades de la pantalla y que sea posibe la copia de su contenido posteriormente.
- **Mapa de bits en memoria**. Instancia de la clase BitmapOffScreen que se usará para alojar en memoria el contenido de una instancia de BitmapScreen y posteriormente copiarse a un buffer accesible desde todos los hilos existentes.

- **Buffer en el que se alojarán los bytes de la imagen**. Área de memoria en la que se alojarán las imágenes capturadas del escritorio para posteriormente ser enviadas.
- **Mutex sincronizador de hilos**. Mecanismo necesario para sincronizar los hilos que leen y escriben en el buffer en el que se encuentran los bytes de la imagen.
- **Número de clientes conectados**. Codifica cuántos clientes hay en la sesión de escritorio remoto.
- **Indicador de si se permite el control por parte de los clientes o no**. Codifica si los clientes conectados pueden controlar el escritorio o solo visualizarlo.

La aplicación servidora es un programa multi-hilo, los puntos de entrada de todos los hilos se implementan como métodos de la clase Server. Cada cliente conectado al servidor será manejado en un hilo independiente. Todos los hilos se explican a continuación:

Hilo GUI

- **Instanciar la clase**. En este método se asocian los parámetros que haya introducido el usuario a los objetos de sesión correspondientes, se inicializa la contraseña, el temporizador que enviará la imagen del escritorio periódicamente y en caso de que la conexión no sea segura se coloca una barra roja en la parte superior advirtiéndolo.
- **Empezar servicio**. Para comenzar el servicio se inicia la ejecución del hilo de red que se encarga de manejar las conexiones entrantes de los clientes.
- **Capturar pantalla**. Primeramente, lo que se hace es copiar el contenido del mapa de bits asociado a la pantalla primaria al mapa de bits en memoria. Después, en caso de que se trabaje con compresión, se creará un mapa de bits propio de .NET con los datos del mapa de bits en memoria para luego poder comprimirlos en formato JPEG.
  Una vez los datos estén comprimidos, se escribirán en el buffer donde se guardan los bytes de la imagen en exclusión mutua. Después de esta copia, se avisa a todos los hilos de red comunicadores del estado del escritorio que estén en ejecución de que el buffer puede ser leído; y finaliza la zona de exclusión mutua. Si no se trabaja con compresión, lo único que se hace es copiar los datos del mapa de bits en memoria al buffer directamente (en exclusión mutua también); y luego, dar la señal a los hilos de redes correspondientes, como se acaba de mencionar.

Hilo de red manejador de conexiones entrantes

Este proceso ligero se encargará de ejecutar una única función, la cual consiste en crear un socket con la dirección IP y puerto introducidos por el usuario, y comenzar a escuchar en ese extremo final, para posteriormente entrar en un bucle infinito en el que la ejecución se bloquea hasta recibir una solicitud de conexión entrante. Una vez se acepta una conexión, se iguala el manejador de red al flujo de datos del socket que se crea al aceptar la conexión. Si se ha seleccionado cifrar la conexión, se le añadirá una capa de protección SSL al flujo de datos y se validará el certificado remoto del cliente, avisando al usuario con un cuadro de diálogo en

caso de que haya incidencias durante el procedimiento de validación. En caso de que no se haya especificado que la conexión esté cifrada, no se le añadirá ninguna capa de protección al flujo de datos. Por último, se inicia la ejecución del hilo de red comunicador del estado del escritorio correspondiente al cliente que se acaba de conectar y se vuelve a esperar por otra conexión entrante.

Hilo de red comunicador del estado del escritorio

Se crea un hilo adicional de este tipo por cada cliente conectado, encargándose este de las siguientes tareas:

- **Envío y recepción de datos**. Para estas funcionalidades se han desarrollado tres funciones. Dos de ellas son las mismas de envío y recepción de información que tiene el cliente, la tercera consiste en una recepción asíncrona de datos encadenada que se utilizará para recibir las órdenes del cliente conectado y luego procesarlas.
- **Simulación de ejecución de eventos**. Se diferenciarán los mismos tipos de eventos que en el cliente; para más información, dirigirse a la Subsección 4.1.2.

  Es importante recalcar que, en la clase del componente Servidor, al procesar el evento de movimiento del ratón, se realiza un mapeo de las coordenadas de pantalla recibidas a coordenadas absolutas normalizadas [30].
- **Fases de la aplicación**. Como es de esperar, el servidor también tiene tres fases:
  - **Negociación**. Se realiza el intercambio de mensajes correspondiente a la fase análoga del cliente (revisar Subsección 4.1.2).
  - **Inicialización**. Al igual que el anterior caso, hace las mismas operaciones que realiza el cliente en la misma fase pero de manera inversa, con la diferencia de que en la conexión con el primer cliente se siguen una serie de pasos de inicialización de objetos que con los siguientes no.
    En primer lugar, se instancia el mapa de bits de pantalla y se procesan los valores de FPS y profundidad de color recibidos[1]. Después, se instancia el mapa de bits en memoria con la información proporcionada por el mapa de bits en pantalla y la profundidad de color, y si no se trabaja con compresión se reserva memoria para el buffer donde se alojarán los bytes de la imagen. Se puede reservar la memoria en este punto debido a que si los datos no están comprimidos la cantidad de bytes de la imagen enviada va a ser siempre la misma. Para terminar, se comenzará el temporizador para capturar periódicamente el contenido del escritorio.
  - **Funcionamiento normal**. Lo primero que se hace es realizar la autenticación del servidor con el certificado que se le ha especificado en caso de que la conexión esté cifrada. Posteriormente, si las etapas de negociación e inicialización terminan con éxito, comienza la ejecución normal. Esta consiste en primeramente crear un buffer en el que se van a recibir asíncronamente las peticiones del cliente conectado, para

[1] Es importante remarcar que los FPS y profundidades de color de los próximos clientes que se conecten, se recibirán pero no se procesarán.

posteriormente entrar en un bucle infinito. El bucle infinito se fundamenta en la ejecución de instrucciones en exclusión mutua. El hilo que lo está ejecutando, se quedará esperando a que se le avise de que haya datos listos para ser leídos en el buffer donde se aloja la imagen. Una vez se le avise, este enviará al cliente el tamaño de la imagen y una copia del buffer que representa la imagen del escritorio.

4.2. Detalles de herramientas utilizadas

En la presente sección se toman decisiones acerca de las tecnologías de desarrollo que se pueden utilizar, así como de la forma de implementar la compresión de la imágenes transmitidas durante la sesión de escritorio remoto.

4.2.1. Tecnologías de desarrollo

Se ha hecho el desarrollo de la aplicación utilizando el marco de trabajo de interfaz gráfica de Windows Forms para .NET. Como IDE se ha usado Visual Studio 2019, el cual incluye un diseñador visual que permite construir la interfaz gráfica de una aplicación de una manera muy productiva usando Windows Forms u otros marcos de trabajo como Windows Presentation Foundation.

.NET tiene dos versiones disponibles [31], las cuales son:

- **.NET Framework**. Se caracteriza por ser de código privativo y solo compatible con Windows.
- **.NET Core**. Se le considera una versión de .NET Framework mejorada, escrita desde cero. Es de código abierto, multiplataforma, modular y ofrece un rendimiento muy superior al de su antecesor. Es importante señalar que su última versión estable tiene el nombre de .NET 5.0, habiendo suprimido “Core” de su nombre.

Debido a que lo que se busca es trabajar sobre un entorno lo más moderno y con el mayor rendimiento posible se ha compilado la aplicación para la última versión de la variante .NET Core, siendo esta .NET 5.0.

4.2.2. Algoritmo de compresión de la información

Se ha escogido la opción de comprimir la información transmitida del servidor al cliente, o sea, las imágenes de su escritorio, para de esta manera enviar la menor carga de datos posible por la red. Para poder implementar dicha función se pueden usar algoritmos como JPEG o ZLIB, diferenciándose en que JPEG es muy eficiente con el tratado de imágenes pero complicado de implementar, y ZLIB es un algoritmo genérico de compresión y más sencillo de implementar pero menos eficiente.

Finalmente, se eligió utilizar JPEG aplicándolo desde GDI+. Dicha técnica sigue siendo eficiente y más sencilla de implementar que el algoritmo de JPEG nativo. Para saber más sobre GDI+ visitar Apéndice E.

Capítulo 5

Pruebas y evaluación

En este capítulo se detallan las pruebas que se han realizado para verificar el funcionamiento adecuado del programa en la Sección 5.1. Asimismo, en la Sección 5.2, se efectúa una evaluación de rendimiento del mismo.

5.1. Pruebas

Esta sección dispone de las pruebas que se han llevado a cabo para validar los requisitos de software enumerados en la Subsección 3.2.4.

Entorno de prueba

La pruebas de la aplicación se han ejecutado sobre el entorno hardware/software mostrado en la Tabla 5.1.

Equipo	**CPU**	**Memoria RAM**	**Disco duro**	**Sistema operativo**	**Versión de .NET**
Ordenador cliente	Intel Core i7-7700HQ 2.80GHz	16 GB DDR4	HDD 1 TB	Windows 10 21H1	.NET 5.0.302
Ordenador servidor	Intel Core i5-4590 3.30GHz	16 GB DDR3	HDD 2 TB	Windows 10 21H1	.NET 5.0.302

Tabla 5.1: Entorno hardware/software

Casos de prueba

Los casos de prueba definidos se rigen utilizando la plantilla de la Figura 5.1. Sus campos son los siguientes:

- **Identificador**. Identifica unívocamente al caso de prueba. Tendrá el formato "CP-XX" donde *XX* refleja el número de secuencia comenzando en 01.
- **Descripción**. Explicación detallada del caso de prueba.

- **Entrada**. Refleja los datos a suministrar para la ejecución de la prueba.
- **Salida**. Es el resultado que se espera al ejecutar la prueba.
- **Origen**. Señala qué requisitos de software ocasionaron la aparición de la prueba.
- **Entorno**. Enuncia los requisitos del entorno en el que se realiza la ejecución de la prueba.

Identificador
Descripción:
Entrada:
Salida:
Origen:
Entorno: n/a

Figura 5.1: Plantilla de casos de prueba

CP-01	
Descripción:	Se tiene una conexión con un equipo remoto, no especificando profundidad de color y tampoco que la conexión esté cifrada.
Entrada:	Dirección IP y puerto al que conectarse junto con la contraseña de autenticación.
Salida:	Se ha transmitido la información en claro y comprimida. Se observa también que la imagen recibida por el servidor no sobrepasa las dimensiones físicas de la pantalla y se puede redimensionar manteniendo la relación de aspecto.
Origen:	F-SR-01, F-SR-04, F-SR-06, F-SR-12
Entorno:	n/a

CP-02	
Descripción:	El equipo remoto deshabilita el control de su escritorio durante una sesión de escritorio remoto.
Entrada:	Click en el botón para deshabilitar el control remoto.
Salida:	El cliente conectado solamente puede visualizar el escritorio del servidor y no controlarlo.
Origen:	F-SR-01, F-SR-04, F-SR-06, F-SR-11, F-SR-12
Entorno:	n/a

CP-03

Descripción:	Se tiene una conexión con un equipo remoto determinando que la conexión esté cifrada.
Entrada:	Dirección IP y puerto al que conectarse junto con la contraseña de autenticación habiendo especificado que la conexión esté cifrada. Hay que detallar también un certificado con su contraseña, y si se está en el cliente el nombre que se espera que tenga el servidor en su certificado.
Salida:	Se ha transmitido la información cifrada.
Origen:	F-SR-01, F-SR-05, F-SR-06, F-SR-12, F-SR-13
Entorno:	n/a

CP-04

Descripción:	Se tiene una conexión con un equipo remoto detallando una profundidad de color concreta.
Entrada:	Dirección IP y puerto al que conectarse junto con la contraseña de autenticación y una profundidad de color.
Salida:	Se ha transmitido la información no comprimida.
Origen:	F-SR-01, F-SR-04, F-SR-07, F-SR-12
Entorno:	n/a

CP-05

Descripción:	Se tiene una conexión con un equipo remoto concretando unos FPS determinados.
Entrada:	Dirección IP y puerto al que conectarse junto con la contraseña de autenticación y una cantidad de FPS.
Salida:	Se ha transmitido la información con la frecuencia de actualización que se ha marcado.
Origen:	F-SR-01, F-SR-04, F-SR-06, F-SR-08, F-SR-12
Entorno:	n/a

CP-06

Descripción:	Varios clientes tienen una conexión con un equipo remoto y lo controlan al mismo tiempo.
Entrada:	Dirección IP y puerto al que conectarse junto con la contraseña de autenticación por parte de varios clientes.
Salida:	Se muestran todos los nombres de los equipos cliente en el servidor y no hay errores al ejecutar órdenes simultáneamente.
Origen:	F-SR-01, F-SR-04, F-SR-06, F-SR-09, F-SR-10, F-SR-12
Entorno:	n/a

CP-07

Descripción:	Se introducen uno o varios parámetros iniciales incorrectos.
Entrada:	Parámetros iniciales de la interfaz del programa mostrados en las Figuras C.7 y C.6.
Salida:	Se notifica al usuario que ha introducido un valor inválido.
Origen:	F-SR-02, F-SR-03, F-SR-13
Entorno:	n/a

CP-08

Descripción:	Se especifica un certificado no válido.
Entrada:	Dirección IP y puerto al que conectarse junto con la contraseña de autenticación. Hay que especificar que la conexión esté cifrada, detallando un certificado junto con su contraseña usada para cifrar la clave privada (de haberla) y el nombre esperado en el certificado del servidor si se está en el cliente.
Salida:	Se notifica al usuario que ha introducido un certificado que no es correcto.
Origen:	F-SR-13, F-SR-14
Entorno:	n/a

CP-09

Descripción:	Valores de las fases del protocolo inválidos (tamaño de imagen a recibir incoherente, dimensiones de pantalla ilógicas, etc.).
Entrada:	Cualquier dato intercambiado en el protocolo, mostrados en la Subsección 3.3.5.
Salida:	Se lanza un mensaje avisando del error del protocolo y se cesa la ejecución.
Origen:	F-SR-15
Entorno:	n/a

Matriz de trazabilidad

La matriz de trazabilidad entre los casos de prueba y los requisitos de software funcionales es la mostrada en la Figura 5.2.

	F-SR-01	F-SR-02	F-SR-03	F-SR-04	F-SR-05	F-SR-06	F-SR-07	F-SR-08	F-SR-09	F-SR-10	F-SR-11	F-SR-12	F-SR-13	F-SR-14	F-SR-15
CP-01	•			•		•					•				
CP-02	•			•		•				•	•				
CP-03	•				•	•					•	•			
CP-04	•			•			•				•				
CP-05	•			•		•		•			•				
CP-06	•			•		•			•	•	•				
CP-07		•	•									•			
CP-08												•	•		
CP-09															•

Figura 5.2: Matriz de trazabilidad casos de prueba/requisitos funcionales

5.2. Evaluación

En esta sección se evalúa el rendimiento del programa diseñado. Para ello, se mostrará el consumo de recursos tanto del ordenador en el que se ejecutará el programa cliente como el del que ejecutará el servidor. Se detallará el uso de los recursos antes de iniciar el programa, y también durante su ejecución en modos mínimos y óptimos de calidad (5 FPS y 25 FPS).

El consumo de recursos sin haber ejecutado el programa se muestra en las Figuras 5.3 y 5.4. Durante la ejecución del programa, el rendimiento en modo mínimo se expone en las Figuras 5.5 y 5.6. Su rendimiento en modo óptimo se muestra en las Figuras 5.7 y 5.8. Es importante marcar que los valores de todas las gráficas son los medidos durante cinco minutos.

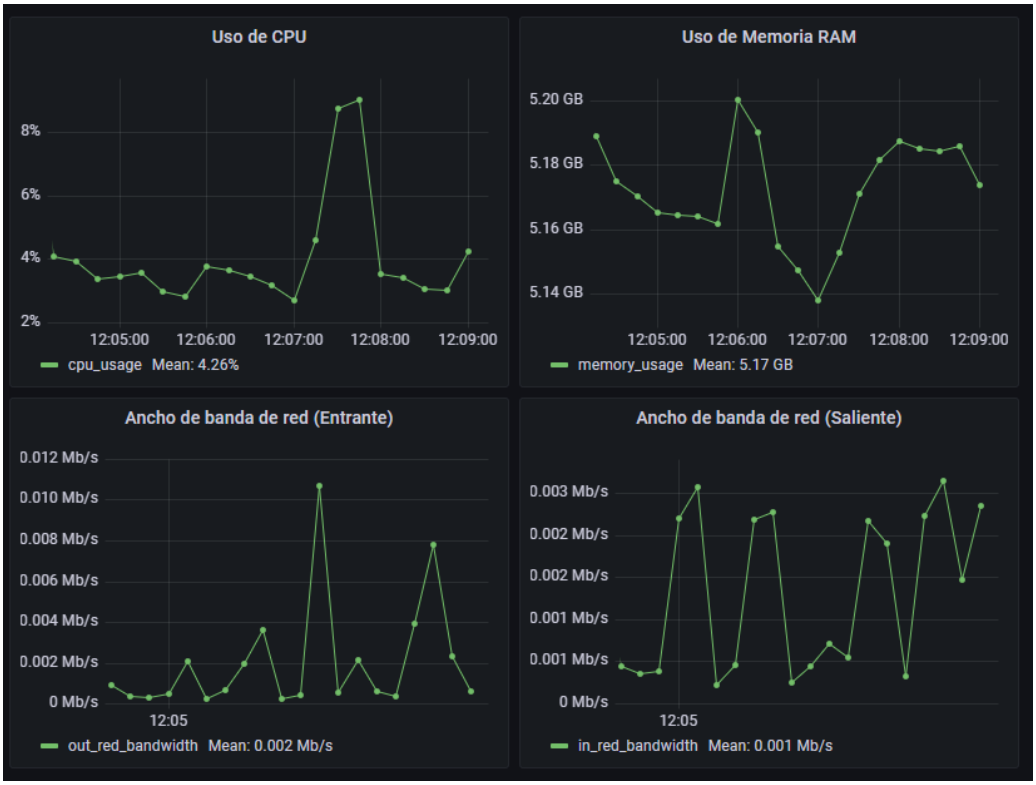


Figura 5.3: Consumo ordenador cliente - preejecución

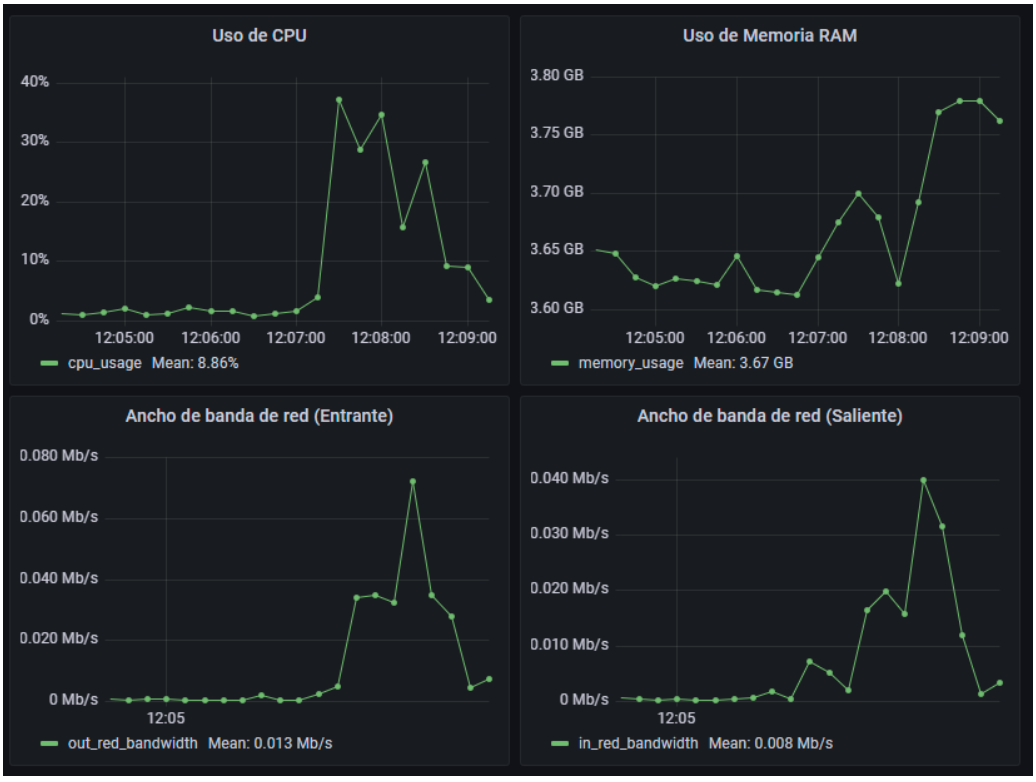


Figura 5.4: Consumo ordenador servidor - preejecución

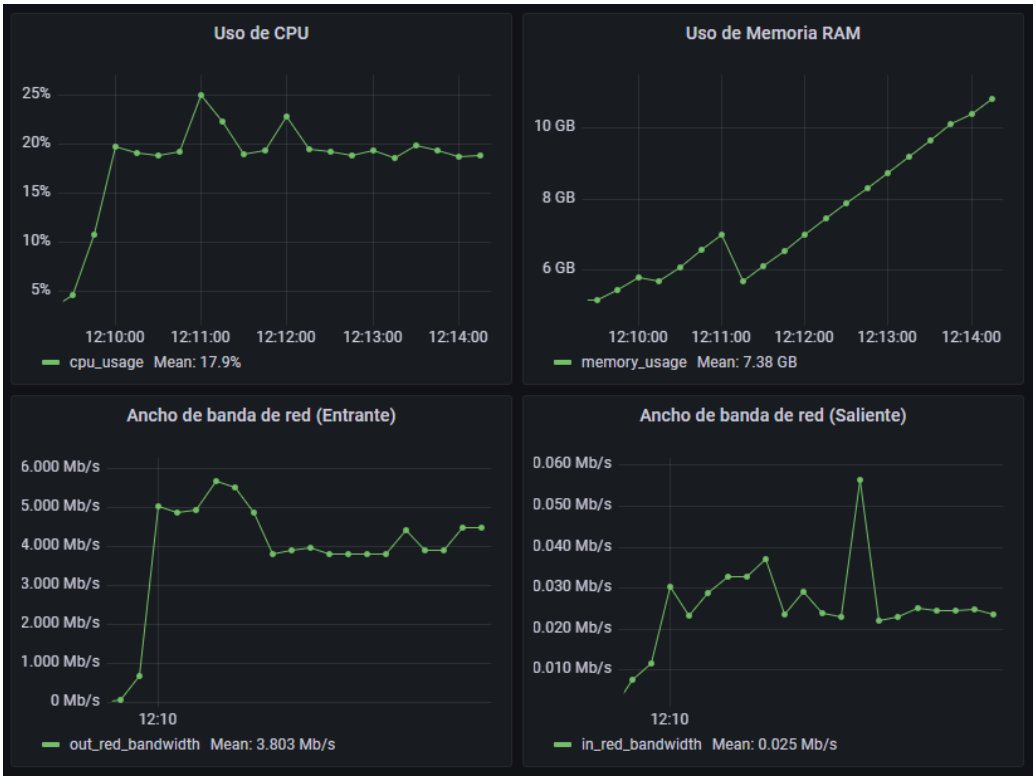


Figura 5.5: Consumo ordenador cliente - ejecución modo mínimo

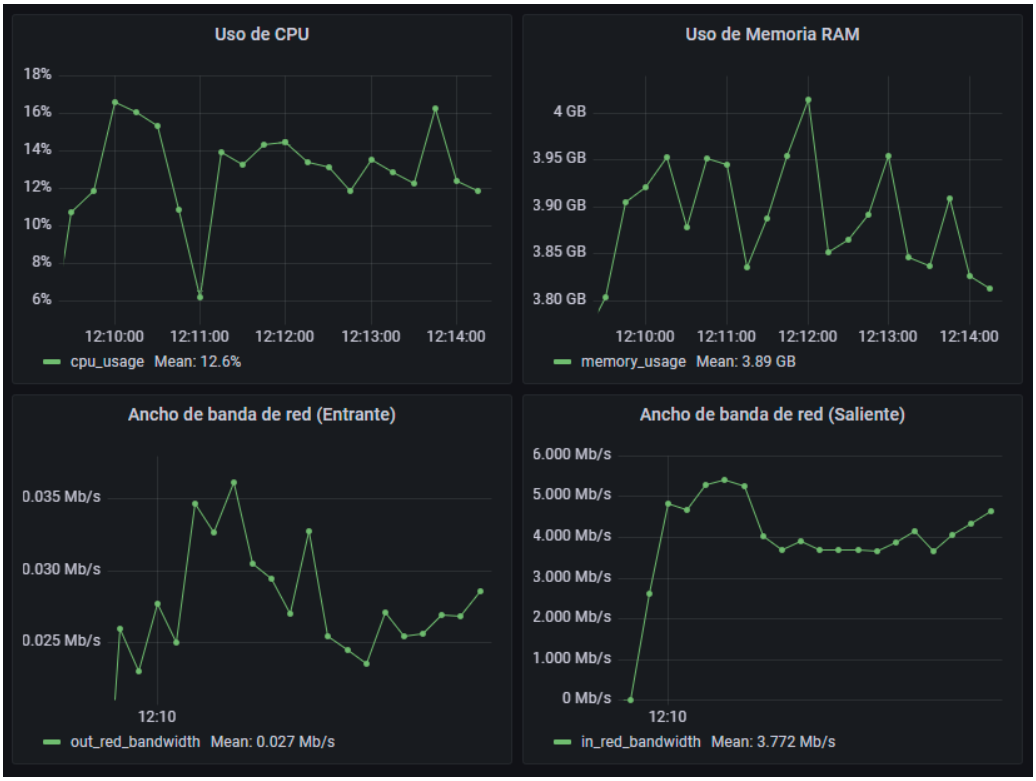


Figura 5.6: Consumo ordenador servidor - ejecución modo mínimo

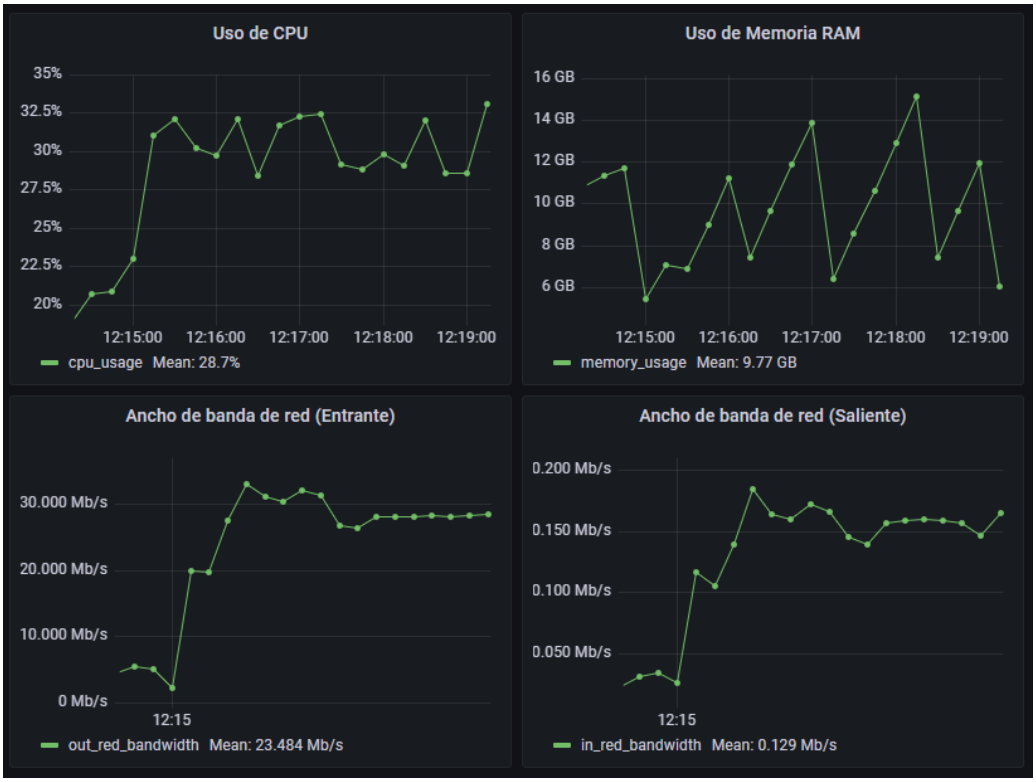


Figura 5.7: Consumo ordenador cliente - ejecución modo óptimo

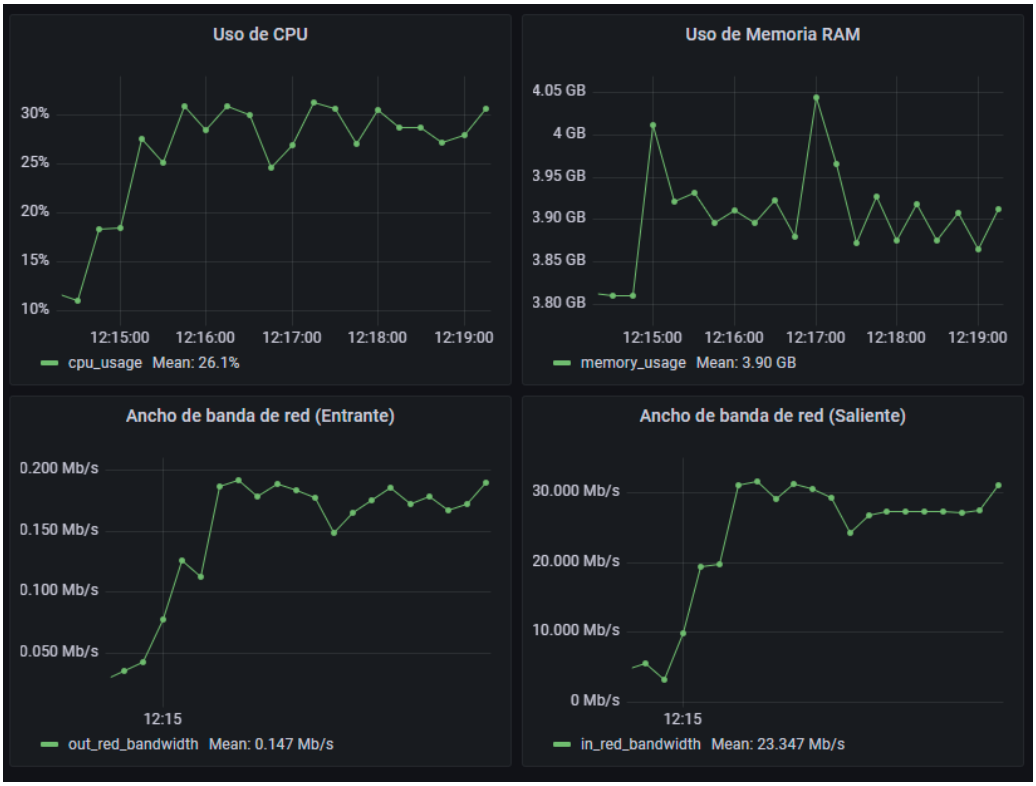


Figura 5.8: Consumo ordenador servidor - ejecución modo óptimo

Restando los valores medios del uso de cada recurso en la preejecución del programa al valor obtenido en el modo mínimo, se tiene que el impacto de la ejecución del programa en dicho modo es el mostrado en la Tabla 5.2.

Equipo	**Uso de CPU**	**Uso de memoria RAM**	**Ancho de banda de red (entrante)**	**Ancho de banda de red (saliente)**
Ordenador cliente	17,9 % - 4,26 % = 13,64 %	7,38 GB - 5,17 GB = 2,21 GB	3,803 Mb/s	0,025 Mb/s
Ordenador servidor	12,6 % - 8,86 % = 3,74 %	3.89 GB - 3,67 GB = 0,22 GB	0,027 Mb/s	3,772 Mb/s

Tabla 5.2: Impacto ejecución modo mínimo[1]

Haciendo lo mismo para el modo óptimo, se tiene que el impacto es el expuesto en la Tabla 5.3.

Equipo	**Uso de CPU**	**Uso de memoria RAM**	**Ancho de banda de red (entrante)**	**Ancho de banda de red (saliente)**
Ordenador cliente	28,7 % - 4,26 % = 24,44 %	9,77 GB - 5,17 GB = 4,6 GB	23,484 Mb/s	0,129 Mb/s
Ordenador servidor	26,1 % - 8,86 % = 17,24 %	3,90 GB - 3,67 GB = 0,23 GB	0,147 Mb/s	23,347 Mb/s

Tabla 5.3: Impacto ejecución modo óptimo[1]

Se observa que el impacto seleccionando el modo de ejecución mínimo (5 FPS) es aceptable teniendo en cuenta que la técnica escogida es polling y no se realiza detección de cambios, es decir, se envía cada fotograma independientemente de si hubo cambios con respecto al anterior (ver Subsección 3.1.2). El impacto en el modo de ejecución óptimo (25 FPS) es bastante mayor, pero aún así es un consumo admisible para un uso en una red de área local, o incluso sobre Internet asumiendo que se dispone de conexiones de banda ancha (DSL o FTTH). Se debe destacar que el mayor consumo de recursos se da en el programa cliente, que el consumo de RAM del servidor es mínimo y que la mínima diferencia de los Mb/s que salen del servidor y entran al cliente o viceversa se puede deber a la multitud de procesos nativos que Windows realiza internamente.

Examinando solamente las gráficas de las dos ejecuciones, se puede extraer lo siguiente:

- **Cliente**. El único valor que varía notablemente a lo largo de la ejecución es el de la memoria RAM usada. El programa reserva cada vez más memoria hasta llegar a un valor determinado, para después liberarla y reservarla otra vez. Esto es debido a la reserva de memoria que se hace para cada fotograma recibido. Se da este comportamiento en vista de que que .NET no libera memoria inmediatamente, sino que espera a la ejecución del recolector de basura. Dicho patrón se puede visualizar mejor en la Figura 5.7, donde se contempla también la liberación de memoria al ejecutarse el recolector de basura. Esto se podría solucionar reservando memoria una única vez para un buffer de recepción lo

[1]El ancho de banda de red consumido tanto entrante como saliente cuando no se ejecuta ninguna actividad en línea es muy cercano a 0 Mb/s. Por este motivo, no se ha hecho una resta de este valor no significativo al indicado por las gráficas en el momento de la ejecución.

suficientemente grande como para alojar cualquier fotograma recibido.
Es importante destacar que el consumo de RAM real de la parte cliente no es el marcado por la gráfica. Si el programa se ejecutase en un ordenador con menor cantidad de memoria (por ejemplo 1 GB), funcionaría también correctamente debido a que .NET se adaptaría a la memoria disponible en el equipo y ejecutaría su recolector de basura con más frecuencia siempre que se diesen condiciones de baja memoria.

- **Servidor**. Ninguno de los recursos varía demasiado a lo largo del tiempo de ejecución.

Se concluye que el rendimiento alcanzado no es perfecto, pero es adecuado, teniendo en cuenta que el software desarrollado es de un proyecto académico y no está destinado a comercializarse. Es importante mencionar que sólo se ha podido probar en las máquinas que disponen del hardware detallado en la Tabla 5.1, pero debido a la naturaleza de la implementación, se espera que también funcione correctamente en equipos con bajas prestaciones.

Capítulo 6

Planificación y presupuesto

En el presente capítulo se refleja la planificación temporal que se ha seguido para el proyecto, incluyendo su representación con un diagrama de Gantt, así como el presupuesto necesario para llevar a cabo el mismo. Dichos datos aparecen en las Secciones 6.1 y 6.2 respectivamente.

6.1. Planificación

El proyecto se ha dividido en las siguientes tareas:

- **Estudio tecnologías**. Basada en la lectura de documentación de las tecnologías necesarias para el desarrollo del proyecto. Estas son la API Win32 y la plataforma de desarrollo de .NET.
- **Análisis y diseño**. Formada por las siguientes subtareas:
  - **Definición de requisitos**. Descripción de requisitos de usuario, casos de uso y requisitos de software.
  - **Diseño del sistema**. Definición de la arquitectura de la aplicación y su protocolo en base a los requisitos especificados en la subtarea anterior.
- **Implementación**. En ella se han desarrollado los distintos componentes de la aplicación. Sus hitos han sido los siguientes:
  - **Visualización remota**. Fue la primera característica que se buscó hacer funcional para la conexión remota. Consistió en una transmisión de datos unidireccional del servidor al cliente realizando un procesamiento adecuado de los mismos para poder mostrarlos.
  - **Control remoto**. Se transformó el tipo de transmisión de la fase anterior a bidireccional pudiendo el cliente enviar también datos con órdenes encapsuladas y que el servidor las ejecutase.
  - **Conexiones seguras**. Se añadió una capa de seguridad a las conexiones con el fin de que estas pudiesen estar cifradas y autenticadas.

- **Pruebas y evaluación**. Comprobación de que el sistema funcionase correctamente, tal y como se especificó en los requisitos, y que tuviese un buen rendimiento.
- **Documentación**. Desarrollo de todos los documentos de este proyecto.
  - **Memoria**. Corresponde a este documento.
  - **Presentación**. Diapositivas explicativas del trabajo realizado presentadas el día de la defensa.

A continuación, en la Figura 6.1, se muestra la planificación temporal que se ha llevado a cabo de las tareas recién mencionadas. Para ello se ha usado un diagrama de Gantt, representándose en el eje vertical las tareas y en el horizontal la duración de las mismas. El proyecto ha tenido una duración total de diez meses, siendo 750 el número total de horas dedicadas.

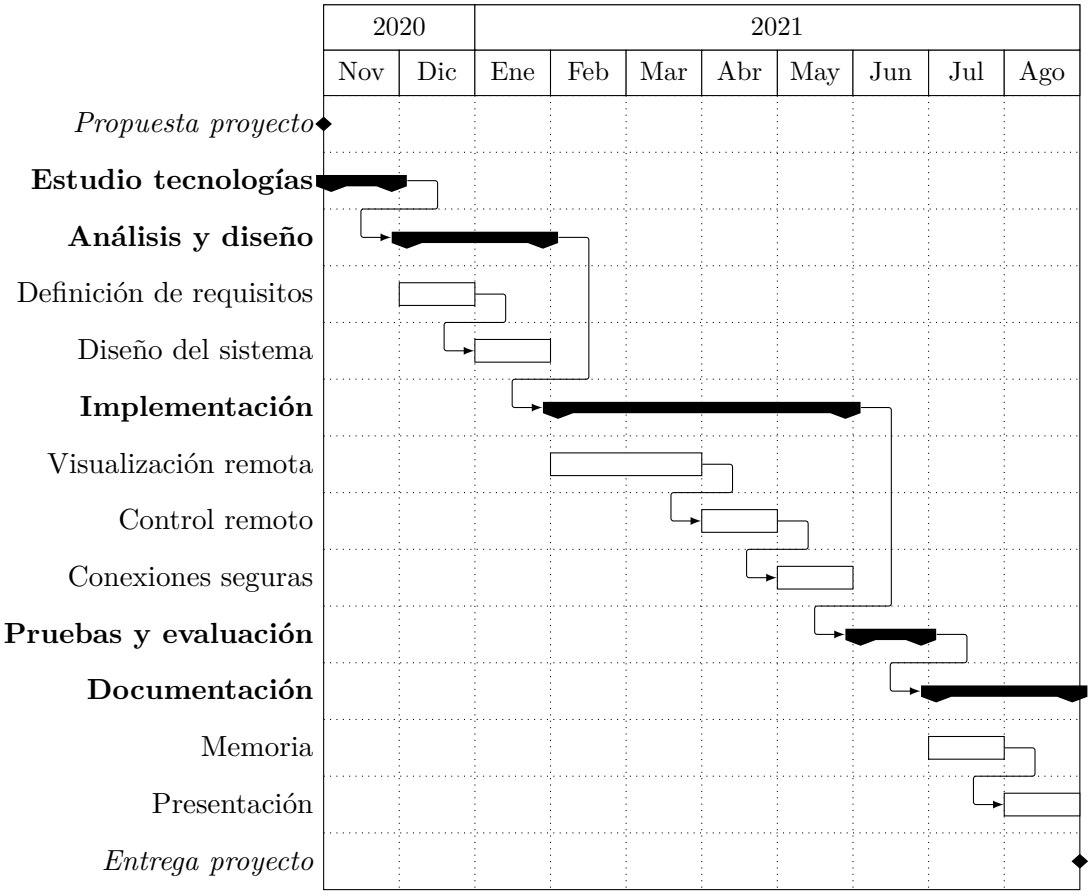


Figura 6.1: Planificación tareas

6.2. Presupuesto

En esta sección se especifica el presupuesto del proyecto, fundamentándose en la planificación expuesta en la Sección 6.1. Los costes se dividen en dos tipos, los directos, que son los relacionados con el gasto realizado en el personal y en el material hardware y software; y los indirectos, los cuales repercuten indirectamente al proyecto. Todos los costes que se calculan en los siguientes apartados no integran el I.V.A (21 %), este se incluirá en el cálculo de los costes totales.

Costes directos

Los costes directos se asocian a dos tipos de costes:

- **Costes del personal**

  Para calcular los costes derivados del personal contratado es necesario saber la cantidad de horas que ha llevado cada tarea. Las diferentes tareas del proyecto han sido definidas en la Sección 6.1 y sus horas están concretadas en la Tabla 6.1.

Tarea	**Horas**
Estudio tecnologías	50 h
Análisis y diseño	80 h
Implementación	400 h
Pruebas y evaluación	20 h
Documentación	200 h
Total	**750 h**

Tabla 6.1: Tiempo empleado a cada tarea

Una vez se tengan las horas dedicadas a cada tarea, se calculan los costes del personal empleando la Ecuación 6.1.

$$Coste = Horas\,totales \times Coste/Hora \tag{6.1}$$

Para calcular el coste por hora del personal, se ha consultado el sueldo bruto anual de los tres perfiles de trabajo, correspondiendo a 34900 € para el analista [32], 29600 € para el analista/programador [33] y 29800 € para el programador [34]; así como el número de horas trabajadas al año estimadas [35], siendo 1820. Una vez obtenidas estas cifras, se ha dividido cada sueldo entre las horas anuales trabajadas.

Dichos costes están reflejados en la Tabla 6.2.

Cargo	**Horas**	**Coste/Hora**	**Coste**
Analista	130 h	19,17 €	2492,10 €
Analista/Programador	280 h	16,26 €	4552,80 €
Programador	340 h	16,37 €	5565,80 €
Total			**12610,70 €**

Tabla 6.2: Costes del personal

- **Costes del material**

  Los costes del material hardware adquirido para realizar el proyecto se detallan en la Tabla 6.3.

Material	**Unidades**	**Coste**
Ordenador portátil MSI GT72VR Dominator Pro	1	2000,00 €
Ordenador sobremesa ASUS	1	500,00 €
Ratón Pictek	2	13,00 €
Teclado Logitech	1	10,90 €
Monitor LG	1	150,00 €
Cable HDMI	1	7,90 €
Total	**2694,80 €**	

Tabla 6.3: Costes material hardware

Los costes del material software obtenido para elaborar el proyecto se reflejan en la Tabla 6.4.

Material	**Unidades**	**Coste**
Microsoft Windows 10	2	145,00 €
Visual Studio 2019	2	0,00 €
Visual Studio Code	2	0,00 €
LaTeX	2	0,00 €
Git	2	0,00 €
Prometheus	2	0,00 €
Windows Exporter	2	0,00 €
Grafana	2	0,00 €
Total	**290,00 €**	

Tabla 6.4: Costes material software

La amortización es el deterioro que sufre anualmente tanto el material hardware como el software, perdiendo de esta manera su valor. Se supondrá que este deterioro será constante, por lo que se utilizará la amortización anual. Se puede calcular con la Ecuación 6.2.

$$Amortización = \frac{Tiempo\ de\ utilización}{Vida\ útil} \times Coste \times Coeficiente \tag{6.2}$$

Es necesario mencionar que el tiempo de utilización de los materiales corresponderá a la duración del proyecto. Se considera que la vida útil de los materiales es aproximadamente de ocho años para los ordenadores, diez para los equipos electrónicos que no procesen información, seis para el software y treinta para los cables [36]. El coeficiente será del 100 % debido a que es el porcentaje de uso que se le dedica al proyecto.

La amortización de los materiales del proyecto se recoge en la Tabla 6.5.

Material	**Coste**	**Coeficiente**	**Tiempo de utilización**	**Vida útil**	**Amortización**
Ordenador portátil	2000,00 €	100 %	10 meses	96 meses	208,33 €
Ordenador sobremesa	500,00 €	100 %	10 meses	96 meses	52,08 €
Ratón x2	26 €	100 %	10 meses	120 meses	2,16 €
Teclado	10,90 €	100 %	10 meses	120 meses	0,90 €
Monitor	150,00 €	100 %	10 meses	120 meses	12,5 €
Cable HDMI	7,90 €	100 %	10 meses	360 meses	0,21 €
Microsoft Windows 10 x2	290,00 €	100 %	10 meses	72 meses	40,27 €
Total					**316,45 €**

Tabla 6.5: Amortización de materiales

Costes indirectos

El gasto de electricidad se ha calculado teniendo en cuenta que el coste promedio mensual de la electricidad en España es de 56,30 € al mes [37]. El importe de Internet corresponde a la tarifa de *fusión Inicia* de Movistar con una fibra simétrica de 300 Mbps [38], el cual se sitúa en los 74,00 € mensuales. Por último, no se han tenido gastos de transporte debido a que las reuniones han sido telemáticas. Tras haber obtenido esas cifras, se han multiplicado por los meses que ha llevado la realización del proyecto.

Los costes están expuestos en la Tabla 6.6.

Material	**Coste**
Electricidad	563,00 €
Internet	740,00 €
Transporte	0,00 €
Total	**1303,00 €**

Tabla 6.6: Costes indirectos

Costes totales

Teniendo en cuenta los costes calculados en las secciones anteriores se calcula el precio de venta de la solución desarrollada. Con el fin de remediar posibles imprevistos, como bajas médicas del personal o averías del material, se añade un margen del 10 % sobre el coste. Por otro lado, se aplicará un margen de beneficio del 25 % sobre el coste incluyendo la cantidad del margen de imprevistos.

El presupuesto del proyecto se resume en la Tabla 6.7.

Descripción	**Coste**
Costes del personal	12610,70 €
Amortización del material hardware y software	316,45 €
Costes indirectos	1303,00 €
Margen de imprevistos (10 %)	1423,02 €
Margen de beneficio (25 %)	3913,30 €
Total sin I.V.A (21 %)	19566,47 €
Total	**23675,43 €**

Tabla 6.7: Presupuesto

Se concluye que el precio final del proyecto corresponde a la cantidad de **23675,43 € (VEINTITRÉS MIL SEISCIENTOS SETENTA Y CINCO EUROS Y CUARENTA Y TRES CÉNTIMOS)**.

Capítulo 7

Marco regulador y entorno socio-económico

En el presente capítulo se detallan en la Sección 7.1 las leyes que se aplican al proyecto y el cómo lo hacen, así como los estándares software que se han seguido. En la Sección 7.2 se explica el impacto social y económico que tiene el proyecto realizado.

7.1. Marco regulador

7.1.1. Legislación y normativa

A continuación, se nombran las leyes que están vigentes tanto en territorio español como europeo:

- **España**

  En España existe una ley de protección de datos, concretamente está en vigor la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales [39]. Dicha ley expone que todos los datos personales de los usuarios deben ser protegidos por los legisladores europeos [40]. Los datos personales son todos aquellos que pueden identificar a una persona física, o sea, cualquier información que por sí misma o que en conjunto con otra pueda llevar a la identificación de alguien.

- **Europa**

  A nivel europeo existe un reglamento que protege la información personal de los ciudadanos y les otorga el derecho de decidir cómo desean que sus datos sean tratados y cómo quieren recibir información de las empresas [41]. Dicho reglamento recibe el nombre de RGPD (Reglamento General de Protección de Datos) [42], entró en vigor en mayo de 2016 y se aplica obligatoriamente desde mayo de 2018.

Es importante mencionar que el software de este proyecto cumple tanto la ley de protección de datos de España como el reglamento europeo debido a que no captura ningún dato del usuario para su posterior procesamiento.

Al transmitir información por la red, hay riesgo de que esta sea interceptada. Esto sobre todo es peligroso si la información es confidencial, como lo son datos de cuentas bancarias, contraseñas, información médica de pacientes, etc. Para evitar la interceptación de estos datos y su posterior procesado se debe elegir tener conexiones cifradas. El programa tiene un sistema de prevención que avisa al usuario de si su conexión no está cifrada mediante una barra roja con el mensaje de "Tu conexión no es segura".

7.1.2. Derechos de autor

La aplicación desarrollada se publicará como software libre en Github [43], bajo la licencia GPL [44].

7.1.3. Estándares software

Para el desarrollo de este TFG se han seguido los siguientes estándares software:

- **Win32**. Win32 es la API usada por Windows y define la forma que ofrece dicho sistema operativo para interactuar con él. Se ha usado esta API para manejar la interfaz de usuario y los gráficos del programa. Sus derechos de autor pertenecen a Microsoft, siendo no pública su implementación y pudiéndose consultar su documentación en su página oficial [45].
- **ECMA-CLI**. El estándar de ECMA-CLI [46] define las bases de la plataforma .NET de Microsoft. Con el paso de los años .NET ha ido divergiendo, pero sigue respetando dicho estándar. Hasta el año 2015, solo era compatible con Windows y era de código privativo, ello derivó la creación de un proyecto llamado Mono Project [47], el cual es una implementación de código abierto multiplataforma del Framework de Microsoft. Dicha implementación también sigue el estándar de ECMA-CLI.

  Se ha usado .NET para todas las funciones que no tuviesen que interactuar directamente con el sistema operativo (las de Win32 en este caso). Es importante entender que el software de este proyecto no sigue el estándar de .NET como tal, debido a que también utiliza funciones de interoperación directa con Windows; por lo que a pesar de que .NET sea multiplataforma actualmente, este programa no se podría ejecutar en un sistema operativo distinto al que es propiedad de Microsoft.
- **JPEG**. Sus siglas vienen de Joint Photographic Experts Group, y es un estándar de compresión y codificación de imágenes digitales. Se ha usado este método de compresión para transmitir las imágenes del servidor al cliente en caso de que el usuario no elija una profundidad de color específica. Los derechos de autor de JPEG pertenecen a la ITU [48].
- **BMP**. BMP es un estándar de facto, pero según la librería del congreso de EE.UU. [49] es un formato propietario de Microsoft debido a que nunca se ha publicado ningún documento de su especificación. Solamente se dispone de fuentes de información a través de Microsoft Docs [50].
- **X.509**. Es un estándar que se utiliza en certificados de clave pública. Se ha usado para generar los certificados de los roles, definidos en la Subsección 3.3.6, existentes en una sesión de escritorio remoto. Sus derechos de autor pertenecen a IETF [51].

- **PKCS #12**. Hace referencia a un formato de fichero que se utiliza como un contenedor de varios objetos criptográficos. Estos objetos son una cadena de certificados X.509 y su clave privada asociada. En el proyecto se ha utilizado para almacenar los certificados X.509 generados junto con su clave privada. Sus derechos de autor pertenecen a IETF [52].

Hay partes del proyecto que en aras de simplificar el desarrollo, no se ha aplicado un estándar, sino una alternativa igual de válida pero mucho menos compleja. Dichos estándares más complejos son:

- **RFB**. Sus siglas vienen de Remote Framebuffer Protocol y es un protocolo que permite la conexión por escritorio remoto entre equipos ejecutando cualquier sistema operativo debido a que funciona en el framebuffer. Los derechos de autor pertenecen a IETF [19] y es el protocolo que utiliza VNC.
  En lugar de implementar un protocolo existente se ha decidido diseñar uno propio (véase la Subsección 3.3.5).

- **RDP**. Sus siglas vienen de Remote Desktop Protocol y es otro protocolo que permite la conexión por escritorio remoto entre equipos ejecutando Windows. Sus derechos de autor son de Microsoft [23].
  De forma similar al anterior caso, se ha optado por implementar un protocolo propio.

- **Radius**. Sus siglas vienen de Remote Authentication Dial In User Service y es un protocolo de autenticación y autorización. Los derechos de autor pertenecen a IETF [53].
  En lugar de usar este protocolo se ha utilizado conexiones SSL añadiendo un método de autenticación con una contraseña lo más segura posible.

7.2. Entorno socio-económico

El software de control remoto tiene un impacto muy positivo en distintos factores:

- **Factores económicos y medioambientales**

  Afecta a las personas individualmente y a las empresas, ya que si necesitan acceder a un equipo remotamente no necesitan desplazarse, esto les supone un ahorro económico muy notable. Asimismo, ayuda a tener unas mejores condiciones medioambientales, debido a que si se ahorran desplazamientos se reduce la contaminación provocada por los transportes, los cuales en la actualidad, mayormente son de combustión.

- **Factores sociales**

  Facilita mucho el teletrabajo, el cual es muy importante actualmente. También hace más sencilla la asistencia y la ayuda de problemas informáticos a personas que lo necesiten sin necesidad de desplazarse. A estas comodidades se le suma también la inmediatez con la que se accede al otro equipo sin importar lo lejos que esté.

  El software de control remoto está evolucionando, pudiendo mimetizarse en distintas funcionalidades como compartir escritorio en videoconferencias, extensión de escritorio entre

varios dispositivos (Airplay de Apple, Chromecast de Google), etc. Para estudiantes que tengan curiosidad por aprender cómo funciona, o quieran trabajar en alguna propuesta de estas adaptaciones de escritorio remoto comentadas; un ejemplo con el código fuente de una versión simplificada, les sería muy útil para comenzar a dirigir su futuro profesional por ese camino.

Capítulo 8

Conclusiones y trabajos futuros

En el capítulo actual, en la Sección 8.1, se exponen las conclusiones a las que se ha llegado tras realizar este Trabajo de Fin de Grado. Esta sección se divide en las Subsecciones 8.1.1, 8.1.2 y 8.1.3; donde se verifica el cumplimiento de los objetivos establecidos en la Sección 1.2, se detallan las conclusiones con respecto al proyecto, y las conclusiones personales respectivamente. También se incluyen las mejoras que se podrían incorporar como trabajo futuro en la continuación del proyecto en la Sección 8.2.

8.1. Conclusiones

8.1.1. Objetivos

A continuación, se hace una revisión de los objetivos definidos al principio del proyecto, en la Sección 1.2, y se confirma el logro de los mismos:

- **Poder controlar un equipo remotamente a través de una conexión de escritorio remoto**
  La aplicación desarrollada no solamente tiene la posibilidad de conectarse a una máquina y controlar su escritorio, sino que también se puede utilizar para visualización remota, siendo en este caso equivalente a la característica de compartir pantalla disponible en otras aplicaciones como Google Meet o Skype. Todo esto es capaz de hacerlo permitiendo el redimensionado de la imagen recibida, respetando su relación de aspecto original y adaptando los movimientos del ratón que se realicen en la nueva ventana redimensionada.

- **Hacer un uso lo más eficiente posible de los recursos utilizados, adaptándose a las capacidades del equipo donde se ejecute**
  Dicho objetivo se ha cumplido basándose en los siguientes hechos:

  - Se ha diseñado un protocolo eficiente minimizando el número de transmisiones de mensajes y el número de bytes intercambiados necesarios para codificar dichos mensajes. Con esta acción se ahorra ancho de banda de red consumido.
  - Se ofrece la posibilidad de elegir la profundidad de color de las imágenes recibidas o

también que se transmitan comprimidas; de esta manera, se consumirá menos ancho de banda de red.

- Se adapta a las capacidades del equipo en el que se vaya a ejecutar, pudiendo tener un funcionamiento aceptable en equipos con CPU menos potentes, menor cantidad de memoria RAM o con una conexión de red lenta. Esta adaptación se realizará seleccionando una frecuencia de envío de imágenes adecuada teniendo en cuenta los recursos que se tengan en el momento. Por consiguiente, se podrá regular el uso de los recursos del equipo en el que se ejecute el programa. La demostración de dicha regulación y el consumo de recursos se muestran en la Sección 5.2.

- **Realizar la transmisión de la información con conexiones seguras**

  Se le permite al usuario tener una conexión cifrada o en claro dependiendo del tipo de entorno en el que se encuentre y la sensibilidad de los datos que desee transmitir. El cifrado no es una imposición debido a que si el usuario no lo necesita porque se encuentra en una red local o los datos a transmitir no son de valor, no es adecuado obligarlo a realizar todo el proceso de generación de certificados.

8.1.2. Proyecto

El Trabajo de Fin de Grado ha requerido mucho tiempo de dedicación, constancia y esfuerzo al ser la "asignatura" del grado en la que no solo hay que demostrar que se han adquirido correctamente todos los conocimientos enseñados en el resto de las asignaturas, sino también ponerlos en práctica. Para el desarrollo del mismo se han hecho reuniones semanales, estando estas primeramente enfocadas en la obtención de conocimientos adecuados del funcionamiento de la API Win32 y la plataforma de .NET; y también el desarrollo del análisis y el diseño. Meses más tarde, estando finalizadas estas fases, se comenzó con la implementación.

Durante la realización del proyecto se encontraron ciertas dificultades, que finalmente pudieron abarcarse. Se detallan a continuación:

- **Generación de los certificados necesarios para cifrar la conexión**. Pese a tener claro cómo era la estructura jerárquica de la PKI de la aplicación, saber qué comandos había que ejecutar exactamente para generar los certificados y firmarlos correctamente, no fue nada sencillo por falta de documentación en Internet al respecto.

- **Parte de programación asíncrona del servidor**. En el grado se ha estudiado una vista general de este tipo de programación, pero no fue obvio aplicarla a este proyecto debido a que la API Win32 ofrecía una amplia variedad de funciones asíncronas y con ejemplos complejos sin realizar llamadas encadenadas tal y como se necesitaba para este proyecto. También hubo que añadir la falta de experiencia previa y el desconocimiento de cuál era la manera óptima de hacerlo.

- **Sincronización de hilos del servidor en una sesión con múltiples cliente**. Debido a los conocimientos de ciertas asignaturas del grado se tenía interiorizada la manera de implementar mecanismos de exclusión mutua y señalización en sistemas multihilo. Sin embargo, su implementación en la plataforma de .NET ha sido distinta y no tan sencilla.

Había distintas maneras de implementarlos y a un nivel mayor de abstracción con el cual no se estaba acostumbrado a trabajar.

Se ha comprendido mejor el principio en el que se basan las aplicaciones de interfaz gráfica. El desarrollo de estas, a diferencia de las de consola se fundamenta en la programación basada en eventos. También se ha comprendido mucho mejor en qué se basa un protocolo de aplicación y con ello la manera de comunicarse entre sí de las aplicaciones cliente-servidor.

Las fases de estudio de la API Win32 y la plataforma de .NET, y la implementación, especialmente esta última, han durado mucho más de lo esperado, pudiéndose confirmar que la implementación ha sido la parte más dura con diferencia del proyecto. Esto ha sido así debido a la complejidad del sistema distribuido que se ha desarrollado y a que no solo ha habido que realizar un estudio previo muy intenso del funcionamiento de Win32 y .NET como se ha mencionado antes, sino saber ponerlo en práctica.

Un proyecto futuro que tenga relación con este se podría realizar más rápido debido a la experiencia adquirida en los ámbitos del desarrollo de aplicaciones distribuidas de interfaz gráfica (con la API Win32 o .NET sobre todo), de programación asíncrona y de criptografía para generar certificados.

8.1.3. Personales

Este proyecto ha sido mi mayor reto del grado con diferencia y también uno de los mayores a nivel personal hasta la fecha. Su desarrollo me ha llevado mucho esfuerzo y sacrificio, pero todo el tiempo dedicado se me ha hecho más llevadero, al haberme encantado la temática del mismo. Me siento afortunado de haber podido hacer un trabajo como este y también orgulloso del resultado obtenido.

Las asignaturas que han sido muy útiles para poder realizar este Trabajo de Fin de Grado son las siguientes:

- **Sistemas operativos**. Ha sido necesario tener claros los conceptos estudiados de programación multihilo y los mecanismos de sincronización y señalización de los procesos ligeros.
- **Sistemas distribuidos**. Los conocimientos adquiridos en esta asignatura han sido vitales para realizar este proyecto. Se conocieron los fundamentos de las aplicaciones cliente-servidor y la necesidad de definir un protocolo para su comunicación.
- **Rama de ingeniería del software**. Dichas asignaturas han sido esenciales para la realización de las fases de análisis, diseño y pruebas del trabajo.
- **Redes de ordenadores**. El funcionamiento de la aplicación desarrollada se basa en el envío y recepción de información por la red, por lo que las nociones aprendidas en esta asignatura han sido importantes. Se asimilaron conceptos como el mecanismo de socket, para permitir comunicación entre procesos, o la tecnología NAT, para poder realizar transmisión de datos fuera de una red local.

- **Criptografía y seguridad informática**. Se han afianzado ideas aprendidas como la necesidad de la existencia de certificados digitales, o la de cifrar la información y firmarla para garantizar las propiedades de confidencialidad e integridad del triángulo CIA.
- **Computación ubicua**. Ha sido útil debido a que se estudió programación asíncrona, la cual ha sido muy importante en la implementación del servidor.

Se ha aprendido a programar en el lenguaje C# para la plataforma .NET, siendo las dos tecnologías desconocidas hasta la fecha debido a que no se han estudiado en el grado. También es importante recalcar que se han aprendido conceptos de desarrollo en el sistema operativo de Windows usando su API Win32, lo cual también ha sido nuevo en vista de que el desarrollo de programas en la asignaturas cursadas se ha centrado en GNU/Linux.

Con este proyecto se han mejorado mucho las habilidades para leer documentación de una API desconocida y poner en práctica las funcionalidades que ofrece. En ciertas asignaturas se ha requerido el estudio de API, pero la documentación que se ha leído de ellas no ha sido tan extensa ni tan compleja como la de Win32.

Por último, se ha conocido y aprendido a usar la herramienta LaTeX, la cual me ha parecido magnífica y mucho más adecuada para documentos científicos o extensos que herramientas habituales como Microsoft Office Word o LibreOffice Writer, teniendo en cuenta su simplicidad y su capacidad de automatización de procesos. Para la generación de tablas de requisitos, casos de uso, matrices de trazabilidad, etc., se ha usado el paquete SRS de Javier López [54].

8.2. Trabajos futuros

A continuación se detallan algunas de las mejoras que se podrían incluir en el proyecto, pudiéndose implementar en las siguientes iteraciones del mismo:

- **Implementación de un mecanismo para atravesar NAT**

  Tal y como está desarrollada la aplicación, si se quisiera usar con dos máquinas que no estén en la misma red local, habría que abrir puertos en el router al que se conecta el servidor para que este fuese visible directamente desde el cliente. Sería interesante implementar un mecanismo para evitar este proceso como STUN, UPnP o desarrollar un servidor de relevos en su lugar. Para más información consultar Apéndice F.

- **Utilización de un protocolo ya existente**

  El sustituir el protocolo diseñado por uno estándar ya existente, implicaría la posibilidad de usar una de las partes de esta aplicación, sea la cliente o la servidora, con la opuesta de una aplicación ya existente. Es decir, si se implementase el protocolo RFB, por ejemplo, se podría usar la parte cliente del programa diseñado, con el servidor de una de las variantes de VNC o viceversa. Esta intercompatibilidad con soluciones populares permitiría incrementar su uso y también su popularidad.

- **Uso de una técnica de captura de imagen y detección de cambios más eficiente**

  El sustituir la técnica escogida por otra como hook de mensajes o "mirror driver" de gráficos implicaría realizar cambios bastante grandes. Habría que migrar el código a un lenguaje de programación de más bajo nivel como C o C++ para poder interactuar directamente con el sistema operativo y no con .NET tal y como se ha hecho. Efectuar esta modificación sería muy adecuada si se quisiese publicar el programa como software profesional, ya que se mejoraría drásticamente la eficiencia del mismo.

- **Implementación del control por shell remota**

  Como se comentó en el Capítulo 2, el control por shell remota no tiene mucha demanda actualmente debido a que las interfaces de los ordenadores son gráficas, pero a pesar de ello, sigue siendo muy útil para ciertas tareas de administración de servidores o usar un número escaso de sistemas operativos. Por esa razón, complementar la aplicación desarrollada con el control por shell remota brindaría a los usuarios la posibilidad de elegir el tipo de control remoto que más se adaptase a sus necesidades sin la obligación de tener instalada más de una herramienta para ello.

- **Control del nivel de calidad de compresión de JPEG**

  Las compresiones de las imágenes se realizan utilizando el valor de calidad por defecto que usa .NET internamente. Si se pudiese elegir este parámetro, se podría variar el nivel de compresión según el usuario lo decida y con ello ahorrar también ancho de banda.

Apéndice A

Summary

A.1. Introduction

Motivation

The great advances of technology and the Internet boom during recent years have shown that people's technological needs have been changing and continue to change. Forty years ago it was not necessary to have a computer at home, although, nowadays this device is indispensable to be able to work and live a normal life. This need is increasing as time passes due to more and more processes becoming computerized (e-commerce, previous online appointment, telematic learning, etc.).

The digital communication tools such as videoconference and remote control have taken center stage in recent years, especially after the pandemic COVID-19. Of the aforementioned tools, the ones that offer remote control stand out, due to the fact that they allow distance assistance when there are problems in a specific system or just if help is needed.

A remote desktop control application offers the possibility of controlling a remote machine just like being in front of it. This project was born because of the need to be able to teach how this technology works (nowadays indispensable) inside the educational environment. Therefore, the development of an application which offers remote desktop assistance in an easy way and in the most efficient way possible was proposed, guaranteeing that it is appropriate to use for teaching some subjects' practicals at degree level, for example, the practicals for Distributed Systems.

On the market there exist many solutions that offer this functionality, each of them has different advantages and disadvantages, however, one element that they all have in common is the fact that they are extremely complex and it is really complicated to understand their functioning, so these solutions are far from being apt for their use in the educational field.

For this Bachelor's Degree Final Project, an easy solution has been proposed, a solution of open source, that is easy to understand and also suitable for education proposals. The most important aspect is that it meets the minimum requirements, that it must have a remote desktop application, and that it achieves this functionality with less complexity, regarding code lines, in contrast to existing solutions.

Objetives

To continue, through doing this project the desirable goal is to reach the following objectives:

- **To be able to control a remote device through a remote desktop connection**

  The user will be able to see what is happening and manage the desktop of the computer which will be controlled remotely, as if it were physically in the same place as the user.

- **To make use of the computer's resources in the most efficient way possible, adapting them to the conditions of the conditions of the system**

  The aim is to manage, in the best way possible, the resources which the program uses, such as the CPU, the RAM, or the used broadband network. In addition to this, the goal is to achieve the adaptation to the conditions of the system, being able to have an acceptable functioning in less optimal conditions with minimal resources.

- **Carry out the transmission of information with secure connections**

  The user will have the possibility of choosing, if they want, their data to be transmitted encrypted or not.

A.2. Proposed solution

- At present, the remote shell applications have a very restricted use, being only adequate in exceptional situations like the use of operating systems that can be used almost completely with the terminal (these are scarce and not too popular in the general population), or the control of configurative settings of an operating system (networks, managing the hard disk capacity, etc.).

- In a remote control connection within a local area network, it is not extremely important that the information transmitted is encrypted due to the fact that it is a secure environment. However, if it is a connection through the Internet, it is vital that the data is encrypted so that the aforementioned connection is safe. This concludes that solutions such as Telnet, Rlogin and some variations of VNC (for example TightVNC), which do not encrypt the connections, are not apt for this purpose.

- The most complete programs which include the greatest amount of functionality are TeamViewer and RealVNC (with TeamViewer in the top position), these are the most adequate for business environments and for more demanding clients who use advanced functions at home.

- The most appropriate application for users with little IT knowledge is what is offered by Google, given that all of its configurations are almost instantaneous.

- For users who worry about their privacy and want to use free software that can be examined by experts with the objectives of avoiding being spied on and avoiding the processing of their personal data, the best options are the variants of VNC or X Window System.

However, despite there being very good and very complete solutions, there does not exist a solution which allows students in educational institutions or curious people to understand how remote control desktop technology works. The existing, published solutions such as free software have a code which, for the average student, is too complex to use and to understand. The existing private code solutions have a source code which can neither be seen nor understood.

The purpose of this project is apt for education given that the code is not extensive and is simple to understand, being also the implemented protocol. For this reason, to develop an application which communicates with the software of this project is not only possible but also achievable for educational projects.

A.3. Initial study

Operating system

The program was developed for Microsoft Windows, this options was chosen because it is, by far, the most used operating system dominating 90 % of the market. If another like GNU/Linux had been chosen, it would have been much more difficult to achieve the main objective of the project, which is that students or curious people can read the source code in depth and run it, as well as understanding how remote control desktop technology works. Therefore it is like this because the aforementioned operating system not only is less popular and has less users, but also because it is not typically pre-installed on any machine at the time of purchase. On the other hand, if the development of the application were for an operating system with which many people were familiar, as in the case with Windows, the aforementioned objective could be achieved in an easier way.

Technique for screenshot images and change detection

In order to be able to transmit the images from the server to the client in a remote desktop session the following techniques can be used:

- **Polling**. It consists of configuring a timer to take screenshots of the desktop server periodically and to be able to send then to the client's machine.
- **Message hook filter**. In Microsoft Windows, the user interface layer has a queue of messages that are sent to the local queues of each window for their processing. These messages include events produced by the user and also events of the system, like for example the mouse movements on a window or a region of a window that must be repainted because its content has changed. Installing the hook would allow the aforementioned messages sent to the queue of each window to be intercepted, with the aim of detecting exactly which parts of the desktop are changing, capturing the visible image in that area, compressing it and sending it. This method offers much more accurate results with less of a workload for the CPU than the aforementioned polling technique, but it is somewhat more complicated.
- **Graphics mirror driver**. With this technique, the video driver would replicate the operations carried out by the primary driver and it could be used to intercept, at a very low

level, the parts of the desktop of a window which are changing and capture them efficiently. This is, by far, the most efficient option, but it is also the most complicated one and it is not attainable for an academic project like this.

The technique that was used for the project was polling, due to the fact that the others would have complicated the implementation so much more - being that they were not attainable, and also because the subject matter is software for an academic project, not commercial software.

Programming language and development platform

The most adequate programming languages for this project are C and C++, given that API Win32 for Windows offers its functions in the aforementioned languages, in this way, enabling interaction with the operating system at a low level for what it needs.

However the use of the polling technique only requires some calls to the API Win32, enabling the use of a higher level language which allows these type of calls to be made.
Therefore, C# was chosen as the language, making use of the .NET Framework as the developed platform, enabling operations to be carried out which do not require direct interaction with the operating system with high level functions and enabling the operations which require direct interaction with low level functions to be carried out. The API Win32 can be invoked, when they proceed from C# thanks to the interoperablility services.

It would only have been worthwhile developing the software with C or C++ if the message hook filter had been chosen or the graphics mirror driver technique, due to the fact that these require a very low level interaction with the operating system.

A.4. Implementation

BitmapLibrary

This component was implemented as a dynamic library due to the fact that it has objects and functions in common which will use both the Client and the Server components. It is composed of the following classes:

- **Bitmap**

  It was implemented as an abstract class which, by definition, cannot be instanced, it represents a generic bitmap and it has common attributes and common operations typical of every type of bitmap that exists on the application (off-screen bitmap and screen bitmap).

- **BitmapScreen**

  When it is instanced, an object of this type will maintain a system context reference GDI (hDC) associated with the primary screen. The width, height and depth of colour in bits-per-pixel will correspond, therefore, with those of the primary screen. In this way, regardless of the type of the original copying function which the bitmap has, it will absolutely carry out a screenshot of the visible area.

It has the capacity to copy the current content of the screen to another bitmap that is specified to it, including the icon of the mouse, in addition to this, receiving the content of another bitmap shown on the screen.

- **BitmapOffScreen**

  It is instanced, specifying a particular height and width of pixels, also specifying a certain color depth. In this process of instancing the following is carried out:

  - A device-independent bitmap is created, specifying that its data be organised in the memory from top to bottom (bitmap top-down). Specifically the type of bitmap is DIB-section, this type of bitmap provides a pointer to the region of its data.
  - The creation of a device context GDI (hDC) in the memory, which will be associated with the bitmap created in the last step. In this way, a copying function will be associated implicitly (by GDI) from the bitmap to a representation independently of the device.

    Due to the fact that the bitmap type is DIB-section, there is direct access to the bytes of the bitmap thanks to the pointer to the data region. Therefore its contents are easily accessible with regard to the aforementioned pointer.

  As well as the BitmapScreen class it is also capable of copying its content to another bitmap, in addition to receiving the content from another bitmap and allocating it in a previously reserved memory.

Client

Figure A.1 shows a general scheme of the implementation of the treatment of the desktop image carried out by the client. The details of this will be explained in the subsection.

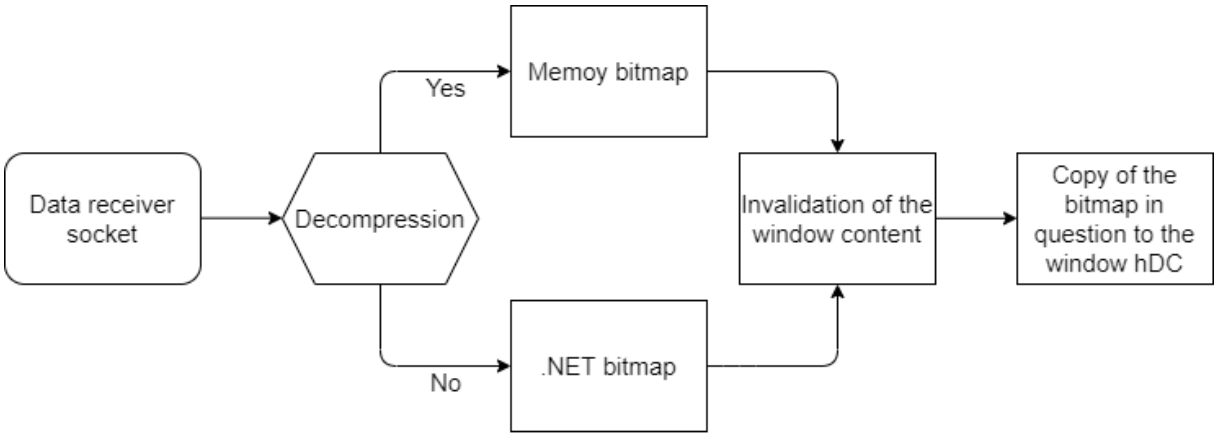


Figure A.1: Handling of receiving images scheme - client

The client functioning involves some additional threads. To be exact, the implementation makes use of three threads, whose entry points are implemented as methods of the Client class. The aforementioned threads are detailed as follows:

- **GUI Thread**
  - **Instancing the class**. In the constructor of the class the initial parameters which the user has introduced are associated with the corresponding session objects.
  - **Connecting to the service offered by the server**. The running of the network thread responsible for receiving desktop server images is initiated.
  - **Repainting the content of the window when it is necessary**. This method will be called when the current content of the window is invalidated. The network thread responsible for receiving the desktop images will be in charge of invalidating it when new data is received. The invalidation of the surface of the window implies that the content can be redrawn.
  - **Handling the events of the keyboard or the mouse produced by the user**. There are the following events:
    - **Keyboard key**
    - **Mouse button**
    - **Mouse movement**
    - **Mouse wheel**
  - **Handling the resize event of the window**. When the window is resized, the received image from the server will be adapted to the new size of the window maintaining its original aspect ratio.
- **Network thread responsible for receiving the desktop images**

  This sub process is in charge of the following tasks:
  - **Sending and receiving data**.
  - **Invalidation of the content**. The content of the current window of the application must be invalidated when new data is received.
  - **Connecting to the endpoint where the service of the server is running**
  - **Application phases**. There are three phases:
    - **Handshake**. It must be checked that the protocol that is implemented on client and server application is the same and the authentication password is sent to the server, then waiting a confirmation message from the server.
    - **Initialization**. There will be processed an information exchange allowing necessary session objects to initialize in order to begin the remote desktop session.
    - **Normal running**. If the handshake and initialization stages finish successfully, the necessary objects for the normal running of the program will be created (these are the concurrent request queue and the bitmap in memory to allocate the data that will be received). In addition to this, there will begin the running of the network thread in charge of sending requests to the server. After carrying

out the aforementioned steps, the thread will go into an infinite loop in which it will receive the size of the image from the server and the image itself. Then the current content of the window will be invalidated.

- **Network thread request communicator**

  This subprocess will only have one functionality, which consists of an infinite loop in which the concurrent request queue is checked to see if there are any pending requests to send to the server. In the case that there are pending requests, they will be sent.

Server

The general scheme of the implementation of the treatment of the desktop image which the server carries out is shown in Figure A.2. The details of this will be described in the subsection.

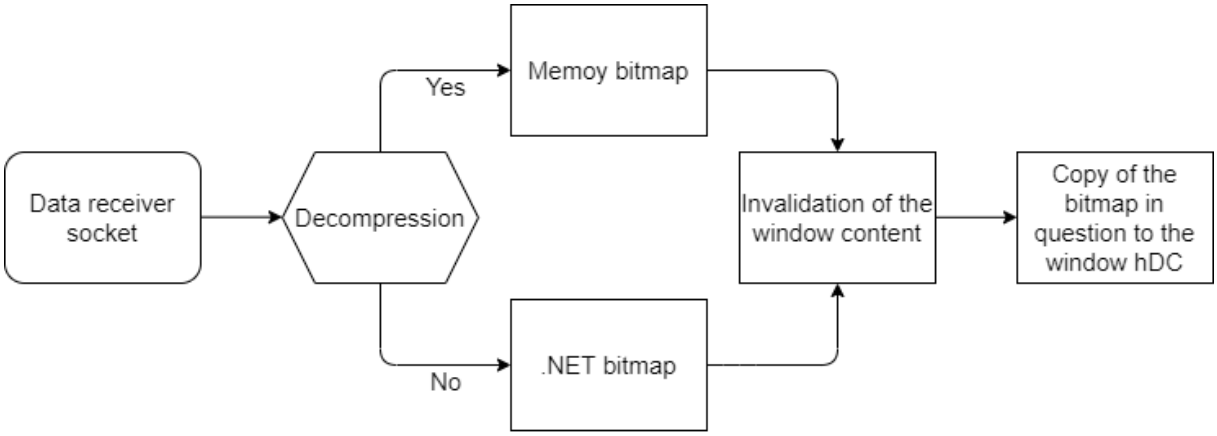


Figure A.2: Handling of sending images scheme - server

The server application is a multi-thread program, the entry points of all the threads are implemented as methods of the Server class. Each client connected to the server will be handled by an independent thread. All the threads are explained as follows:

- **GUI Thread**

  - **Instancing the class**. The initial parameters which the user has introduced are associated with the corresponding session objects. Also the password and the timer which will send the image of the window periodically will be initialized.
  - **Starting the service**. To begin the service, the running of the network thread which is in charge of handling the incoming connections of the clients is initiated.
  - **Taking a screenshot**. First of all, it is necessary to copy the content of the screen bitmap to the memory bitmap. Then, the memory bitmap data is copied to the buffer where the image bytes are saved in mutual exclusion and after that the signal is given to the corresponding network threads.

- **Network thread handler of incoming connections**

  Its function is to enter into an infinite loop in which the running is blocked until it captures an incoming connection. Once a connection is accepted, the network thread desktop status communicator corresponding to the recently connected client starts running and waits again for the next incoming connection.

- **Network thread desktop status communicator**

  An additional thread of this type for each connected client is created. It is in charge of the following tasks:

  - **Sending and receiving data**.
  - **Simulation of execution of events**. The server has the same type of events that the client has.
  - **Application phases**. There are three phases:
    - **Handshake**. The same operations as the client are carried out, but in reverse.
    - **Initialization**. Similar to the above case, the same functions carried out by the client, but they are done in reverse.
    - **Normal running**. If the handshake and initialization phases are concluded successfully, normal running can begin. In this phase, a buffer is created which is going to receive asynchronously the requests of the connected client and then later go into an infinite loop. The aforementioned loop is based on the execution of instruction in mutual exclusion. The thread which is running it, will keep waiting until it is made aware that there is data ready to be read in the buffer where the image is allocated. Once it is aware of this, it will send the size of the image and the image itself to the client.

A.5. Evaluation

The impact of the running of the program in its minimum mode (5 FPS) and in its optimum mode (25 FPS) is shown in the Tables A.1 and A.2:

Machine	**CPU use**	**RAM use**	**Broadband network (incoming)**	**Broadband network (outgoing)**
Client computer	13,64 %	2,21 GB	3,803 Mb/s	0,025 Mb/s
Server computer	3,74 %	0,22 GB	0,027 Mb/s	3,772 Mb/s

Table A.1: Impact running minimum mode

Machine	CPU use	RAM use	Broadband network (incoming)	Broadband network (outgoing)
Client computer	24,44 %	4,6 GB	23,484 Mb/s	0,129 Mb/s
Server computer	17,24 %	0,23 GB	0,147 Mb/s	23,347 Mb/s

Table A.2: Impact running optimum mode

It has been observed that the impact by selecting the minimum running mode (5 FPS) is acceptable, bearing in mind that the chosen technique is polling and change detection is not carried out, that is to say, each photogram was sent independently whether there were changes with respect to the previous one. The impact in the optimum running mode (25 FPS) is quite high, but even so, it is an admissible consumption for use in a local area network or even on the Internet, assuming that it is using broadband connections (DSL or FTTH).

It must be concluded that the level of performance achieved is not perfect, nevertheless it is adequate, bearing in mind that the software developed is for an academic project and is not destined for commercial use. It is important to point out that the software was only able to be tested in machines that contain the hardware detailed, however due to the nature of the implementation, it is expected to function correctly also in less powerful devices.

A.6. Planning and budget

Planning

The project consisted of the following tasks:

- **Technologies study**. Based on the reading of the documentation of the technologies for the development of the project. These are the Win32 API and .NET Framework.
- **Analysis and design**. It is composed of the definition of requirements and system design.
- **Implementation**. The application components have been developed in it. It is divided into three phases, remote visualization, remote control and secure connections.
- **Testing and evaluation**. These involve the verification that the system works correctly and that it has a good performance.
- **Documentation**. This correponds to this document and the guidance slides explained on defense day.

The project has had a duration of 10 months, with 750 being the total number of hours dedicated to it.

Budget

The detailed direct and indirect costs do not have V.A.T. included, it will be added on to the final calculation of the total costs.

- **Direct costs**

  Direct costs are those related with the expenditure on staff, hardware and software material. Amortization is the annual deterioration of both hardware and software material, losing its value in this way. It will be assumed that this deterioration will be constant, for this reason, the annual depreciation shall be used.

  The personnel cost and the amortization of materials are displayed in the Tables A.3 and A.4.

Job title	**Hours**	**Cost/Hour**	**Cost**
Analyst	130 h	19,17 €	2492,10 €
Analyst/Programmer	280 h	16,26 €	4552,80 €
Programmer	340 h	16,37 €	5565,80 €
Total			**12610,70 €**

Table A.3: Personnel costs

Material	**Cost**	**Coefficient**	**Utilization time**	**Average life span**	**Amortization**
Laptop computer	2000,00 €	100 %	10 months	96 months	208,33 €
Desktop computer	500,00 €	100 %	10 months	96 months	52,08 €
Mouse x2	26 €	100 %	10 months	120 months	2,16 €
Keyboard	10,90 €	100 %	10 months	120 months	0,90 €
Monitor	150,00 €	100 %	10 months	120 months	12,5 €
HDMI Cable	7,90 €	100 %	10 months	360 months	0,21 €
Microsoft Windows 10 x2	290,00 €	100 %	10 months	72 months	40,27 €
Total					**316,45 €**

Table A.4: Amortization of materials

- **Indirect costs**

  Indirect costs are those which have an indirect impact on the project. They are shown in the Table A.5.

Material	**Cost**
Electricity	563,00 €
Internet	740,00 €
Transport	0,00 €
Total	**1303,00 €**

Table A.5: Indirect costs

- **Total costs**

  In order to cover possible contingencies, such as medical casualties of personnel or damage to equipment, a margin of 10 % is added to the cost. On the other hand, a profit margin of 25 % will be applied to the cost including the amount of the unforeseen margin.

  The budget is detailed in the Table A.6.

Description	**Cost**
Personnel costs	12610,70 €
Amortization of hardware and software equipment	316,45 €
Indirect costs	1303,00 €
Contingency margin (10 %)	1423,02 €
Profit margin (25 %)	3913,30 €
Total excluding V.A.T (21 %)	19566,47 €
Total	**23675,43 €**

Table A.6: Budget

It is concluded that the final project price corresponds to the amount of **23675,43 € (TWENTY-THREE THOUSAND, SIX HUNDRED AND SEVENTY-FIVE EUROS, THREE CENTS)**

A.7. Regulatory framework and socio-economic environment

Regularoty framework

- **Spain**

  In Spain there exists a data protection law, to be specific the "Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales" which is currently in effect. The aforementioned law expresses that the personal data of the user must be protected by European legislative. Personal data is everything that can identify a physical person, or that is to say, any information related to the person or which together with other information could lead to the identification of the owner.

- **Europe**

  In Europe there exists a regulation that protects the personal information of citizens and it gives them the right to decide how they want their data to be dealt with and how they want to receive information from companies. The aforementioned regulation has the name of GDPR (General Data Protection Regulation), it came into effect in May 2016 and it has been in obligatory use since May 2018.

It is important to mention that the software for this project complies with both the European GDPR and the Spanish data protection law due to the fact that there is no data processing.

Transmitting information via the Internet carries a risk that such information might be intercepted. This is especially dangerous if the information is confidential, such as bank account details, passwords, medical information of patients, etc. To avoid the interception of this data and its prior processing, encrypted connections must be chosen. The program has a system of prevention of information leakage that warns the user if their connection is not encrypted by showing a red bar with the message "Your connection is not secure".

Socio-economic environment

- **Economic and environmental factors**

  The software helps individual people or companies, in that if they need to access a machine remotely they do not have to move to a different place, this reduces the economic expenditure substantially. In addition to this, regarding the environment, it has great benefits, given that the user is not contributing to the problem of pollution (not using transport) as they do not have to leave the office or home environment.

- **Social factors**

  Teleworking, which is very important nowadays, is much easier. IT problems can be solved more easily, in the sense that the problem solvers do not need to leave their work or home environment.

  Remote control software is evolving, it can adapt to different functionalities to share the desktop in videoconferences, desktop extension amongst various devices (Apple Airplay, Google Chromecast), etc. For students who are curious about learning how it works or for those who want to work in one of the aforementioned remote desktop adaptions, an example with a simplified version of the source code would be very useful in leading them in the right direction for their future profession.

A.8. Conclusions and future work

Objetives

- **To be able to control a remote device through a remote desktop connection**

  The developed application not only has the possibility of being connected to a device with the control of its desktop, but it can also be used for remote visualisation, being, in this

case, equivalent to the screen sharing characteristic which is available in other applications like Google Meet or Skype. All this is capable of being done, allowing the resizing of the received image, respecting its original aspect ratio and adapting the movements of the mouse which are done in the new resized window.

- **To make use of the computer's resources in the most efficient way possible, adapting them to the conditions of the system**

  - The user can choose the possibility of depth of color of the received images or choose that they are transmitted in a compressed format, therefore consuming less broadband.
  - The software is adapted to the conditions of the system, enabling an acceptable functioning on devices with less powerful CPUs, less amount of RAM or with a slower connection. This adaptation will be done by choosing an adequate frequency for sending the images bearing in mind the resources which the user has at that moment. Consequently, the user can regulate the use of the resources of the device in which the program is running.

- **Carry out the transmission of information with secure connections**

  This allows the user to have an encrypted connection or not, depending on the particular type of environment and the sensitivity of the data which the user wants to transmit. The encryption is not obligatory due to the fact that it is not necessary in a local network or in the case that the data being transmitted is not important, it is not necessary to subject the user to the creating of the certificate.

Project

The Bachelor's Degree Final Project has required a lot of time and dedication, continuous work and effort in that it is the "subject" of the degree course in which it must be shown that not only all the educational knowledge has been acquired from the rest of the subjects but also putting this acquired knowledge into practice. In order to achieve this, there were weekly meetings, the focus of the first meetings was the acquisition of the adequate knowledge of the functioning of the API Win32, the functioning of the .NET Framework, as well as the development of the analysis and the design. Many months later, having completed these phases, the implementation process started. Throughout the course of the project, some difficulties arose which were finally resolved, these are detailed as follows:

- **Creation of the certificates required to encrypt the connection**
- **Asynchronous programming part of the server**
- **Synchronization of the threads of the server in a session with multiple clients**

A better understanding was attained of the principle on which are based the graphic interface applications. The development of this, unlike those of the console is rooted in the programming

based on events. A greater knowledge was gained on what an application protocol is based on, and with this protocol, the client-server applications can communicate amongst themselves.

The study phase of the API Win32 and the .NET Framework, as well as the implementation phase, especially the latter, lasted much longer than expected, confirming that the implementation process has been, by far, the most complicated part of the project. This has been due to the complexity of the distributed system which was developed and also based on the fact that not only was there a very intensive, prior study of the functioning of the aforementioned Win32 and the .NET Framework, but there was also the time taken to put them into practice.

A future project which may be similar to this one could be carried out more quickly due to the attained experience in the areas of the development of the graphic interface distributed applications (especially with regard to the API Win32 or .NET), also in the areas of asynchronous programming and cryptography to create certificates.

Personals

This project has been, by far, the greatest challenge of my degree course and on a personal level too, it has been one of the biggest challenges of my life. Although the thesis has required a huge amount of effort and self-sacrifice, for me, every moment dedicated to it has been worthwhile because the subject matter of the dissertation is of particular interest to me as well as being something that I love. I feel fortunate for having been able to complete a project like this and I also feel proud of the final result.

The subjects which have been very useful in order to do this Bachelor's Degree Final Project are the following:

- **Operating systems**
- **Distributed systems**
- **Related to software engineering**
- **Computer networks**
- **Cryptography and computer system security**
- **Ubiquitous computing**

I have learned to program in C# for the .NET Framework being the two technologies which were previously unknown to me until I started working on the project. Also it is important to point out that my attainment of the concepts of development of the operating system of Windows (using API Win32), has also being something new, due to the fact that during the degree course the programming was centered on the GNU/Linux operating system.

This project has helped improve the skills for reading the documentation of an unknown API and for putting into practice its functionality. In addition to this, I have gained the knowledge and the understanding of the use of the tool LaTeX , for me, this tool has been magnificent and much more appropriate for scientific or extensive documents than the usual tools like Microsoft office Word or LibreOffice Writer.

Future work

- **Implementation of a mechanism for NAT traversal**

  It would be interesting to implement a mechanism like STUN, UPnP or to develop a relay server to avoid port forwarding in case the user wants to use the application with two machines that are not on the same local network.

- **Using an already existing protocol**

  To substitute the design protocol for a standard already existing one would imply the possibility of using one of the parts of this application, be it the client or the server, with the opposite one on the already existing application.

- **Use of a more efficient technique for screenshot images and change detection**

  If the program were to be published as professional software, this modification would be very appropriate to put into effect.

- **Implementation of remote shell control**

  Including the remote shell control would offer the user the possibility of choosing the type of remote control which is more suited to their needs without the obligation of having to have more than one already installed tool for this.

- **Controlling the quality level of the JPEG compression**

  Image compression is carried out by using the default quality value which .NET uses internally. If this parameter can be chosen, the compression level could be varied, and, consequently, broadband will be saved.

Apéndice B

Glosario

- **API**. Application Programming Interface, es un conjunto de funciones que se utilizan para desarrollar componentes software y permitir su comunicación por medio de un conjunto de reglas.
- **CPU**. Central Processing Unit, es un componente hardware que se encarga de interpretar instrucciones empleando operaciones aritméticas y matemáticas.
- **Framework**. Marco de trabajo, es una estructura conceptual que se usa como una plantilla base para desarrollar un proyecto en lugar de desarrollarlo desde cero.
- **GPL**. GNU General Public License, es una licencia usada en el software libre que otorga a los usuarios de un programa la capacidad de usarlo, examinarlo, compartirlo y modificarlo; protegiéndolo a su vez de intenciones de apropiación que reduzcan estos derechos.
- **GUI**. Graphical User Interface, es un programa que utiliza una colección de objetos gráficos para representar información y permitir la interacción con el usuario.
- **IDE**. Integrated Development Enviroment, es un programa que facilita el desarrollo de software al programador a través de herramientas de automatización y depuración.
- **Software libre**. Es un tipo de software que se caracteriza porque su código fuente puede ser examinado, modificado y compartido sin restricciones.
- **TFG**. Trabajo de Fin de Grado.

Apéndice C

Manual de usuario

Este apéndice[1] contiene instrucciones de instalación y uso de la aplicación desarrollada.

Manual de instalación

Existen dos maneras de instalar el programa:

- **Descargar código fuente y compilarlo**

  Primeramente, hay que dirigirse a la página del IDE de Visual Studio de Microsoft [55] y descargar la última versión Community. Una vez descargado y abierto el instalador, se especificará *Desarrollo de escritorio de .NET* como carga de trabajo a instalar como se muestra en la Figura C.1.

Figura C.1: Instalación Visual Studio [55]

[1]En este manual hay capturas de pantalla del software de Visual Studio 2019, Github y Windows. A pesar de que dichas capturas sean de propiedad privada, su contenido no lo es, por lo que al hacer uso de ellas, se han referenciado las páginas de dichas herramientas.

Cuando acabe la instalación, ya se tendrán preparados tanto el IDE como el SDK de .NET, incluyendo sus herramientas de desarrollo. Como segundo paso, se procede a abrir el programa y se clicará en *Clonar un repositorio*, especificándose posteriormente la dirección del repositorio del software desarrollado[2] [43] como se expone en la Figura C.2.

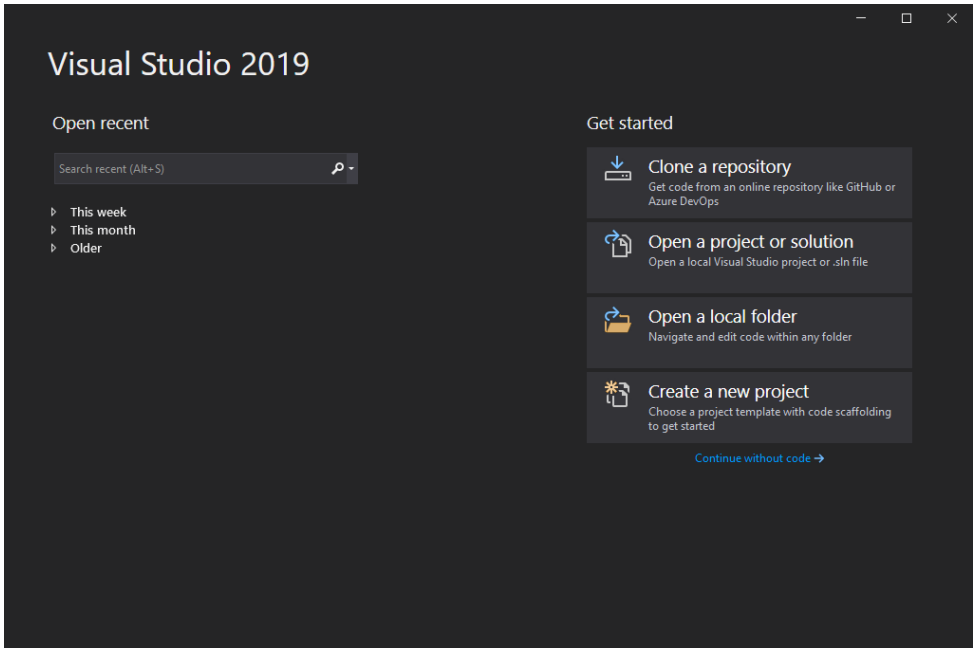


Figura C.2: Clonado repositorio [55]

A partir de este momento, la carpeta que se haya seleccionado como destino del repositorio tendrá dos subcarpetas (*VirtualLinkedDesk* y *Scripts*); la primera contendrá tres proyectos, los cuales se podrán ver desde el explorador de Visual Studio. Dicha vista se puede observar en la Figura C.3.

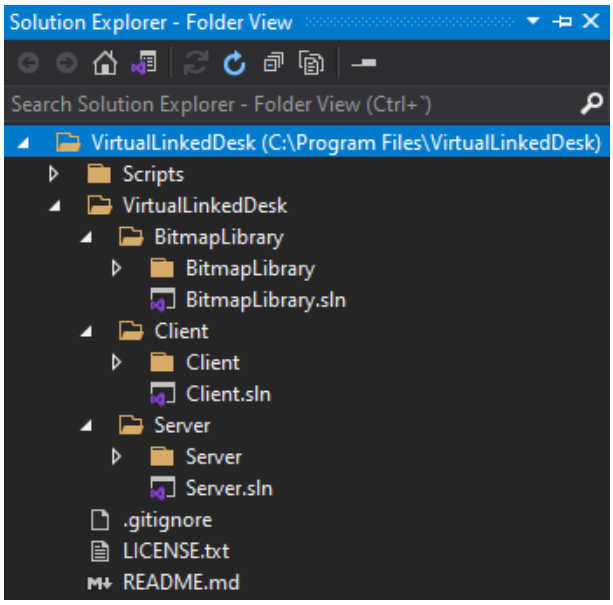


Figura C.3: Proyectos visibles del explorador [55]

[2] https://github.com/Tony450/VirtualLinkedDesk

En primer lugar, hay que dirigirse al proyecto *BitmapLibrary* y compilarlo[3], con el fin de que los otros puedan hacer uso de él. Posteriormente, hay que ir al proyecto *Client* o *Server*, dependiendo del programa que se quiera utilizar, y compilarlo.

Una vez realizados estos pasos ya se tendrá todo listo en el primer equipo. En la segunda máquina habrá que realizar exactamente los mismos pasos salvo el último, en el que después de compilar la biblioteca, se compilará el proyecto que no se llevó a cabo en el primer equipo.

- **Descargar instalador**

  En primer lugar, hay que dirigirse a la dirección del repositorio de la aplicación desarrollada [43] y clicar en la sección de Lanzamientos, igual que en la Figura C.4. En dicha sección, se descargará la última versión publicada.

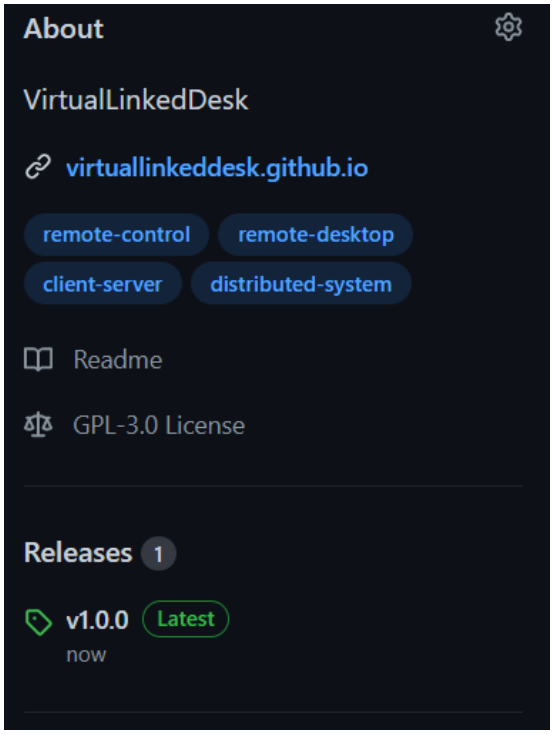


Figura C.4: Lanzamientos Github [56]

Una vez descargada, se instala siguiendo los pasos del instalador como cualquier otra aplicación.
Por último, para ejecutar la aplicación es necesario tener .NET 5.0 instalado [57]. Para esto, se puede elegir instalar el SDK, el cual incorpora tanto las herramientas de desarrollo como el entorno de ejecución, o instalar exclusivamente el entorno de ejecución.

[3]Para compilar un proyecto en Visual Studio hay que hacer click sobre sobre Build > Build Solution o teclear la combinación de teclas Ctrl + Shift + B. Si ya se tiene el IDE de Visual Studio instalado, es importante saber que la versión de .NET para la que están configurados los proyectos para compilarse (.NET 5.0), es solo compatible con la versión 16.8.0 del IDE en adelante. Se debe actualizar en caso de tener una versión más antigua.

Manual de uso

La aplicación se puede usar de dos maneras:

- **Conexión en claro**

  Al empezar la ejecución de la aplicación se mostrará una ventana como la de la Figura C.5 en el cliente y otra como la C.6 en el servidor.

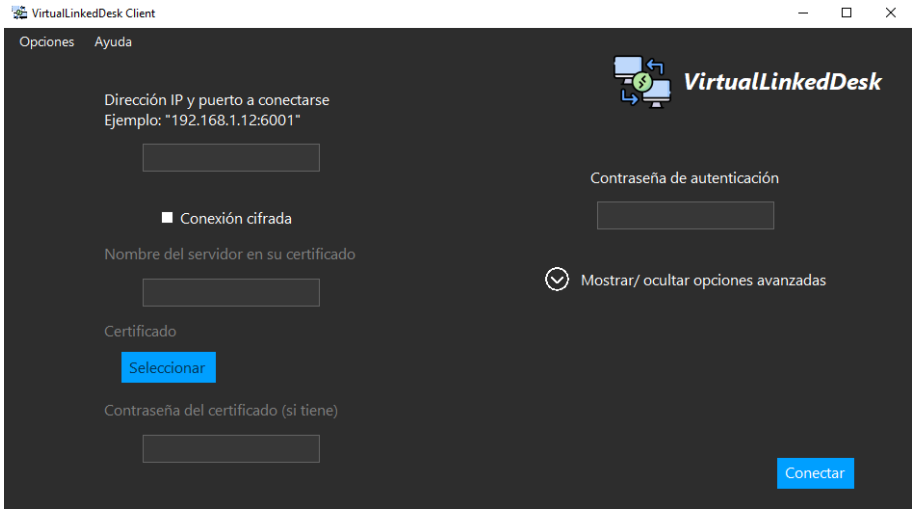


Figura C.5: Aplicación cliente inicio

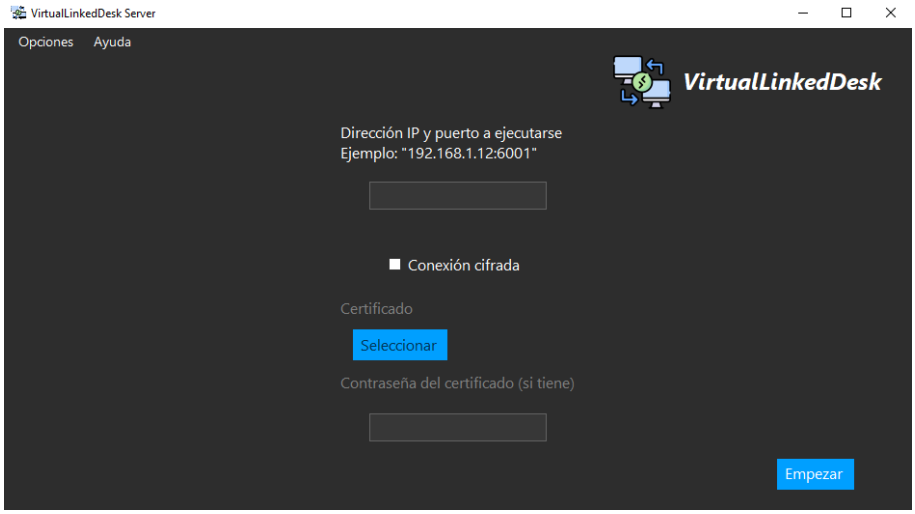


Figura C.6: Aplicación servidor inicio

Para comenzar la ejecución del servidor, en primer lugar, se ha de indicar la dirección IP de una de las tarjetas de red que tenga su equipo instaladas y el puerto donde va a empezar el servicio. Una vez se haya especificado se podrá poner en marcha clicando sobre el botón empezar.

En la aplicación cliente habrá que introducir como mínimo la dirección IP y el puerto que se ha especificado en el servidor, y la contraseña que generó este al comenzar el servicio

como medio de autenticación. Si se da click sobre *Mostrar opciones avanzadas*, se podrá introducir los FPS y también la profundidad de color deseada de la imagen recibida (lo cual implica que esta no viaje comprimida). El aspecto de la ventana con dicho desplegable se refleja en la Figura C.7.

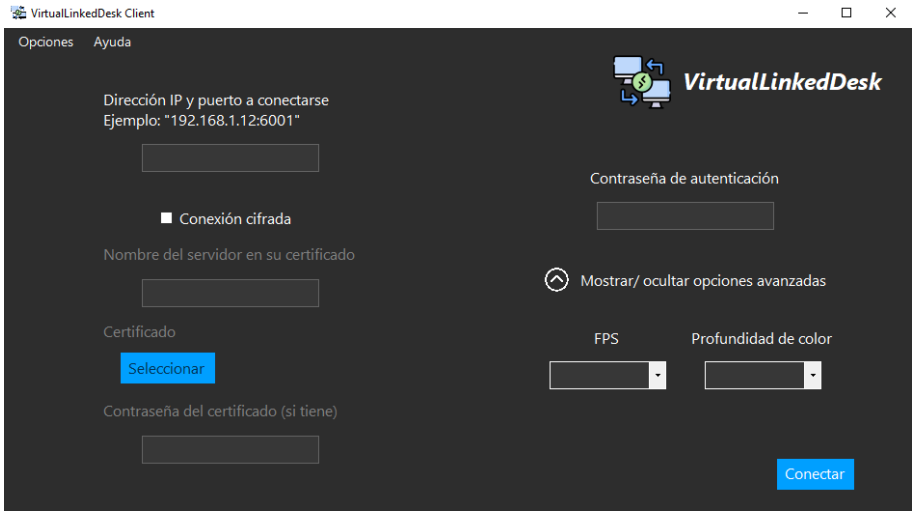


Figura C.7: Aplicación cliente - opciones avanzadas

Una vez el servidor empiece a ofrecer su servicio y el cliente se conecte a él, este empezará a mostrar el estado del escritorio del servidor con los cambios que se vayan realizando. El aspecto de las aplicaciones cliente y servidor se expone en las Figuras C.8 y C.9.

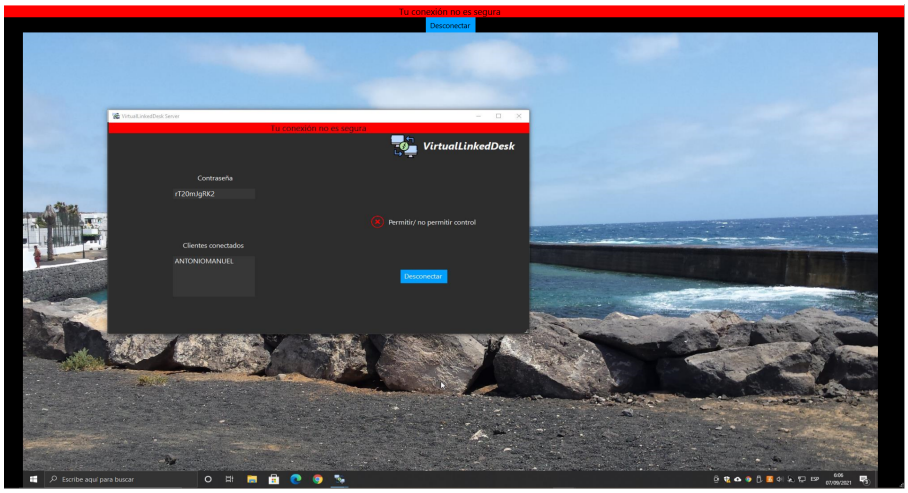


Figura C.8: Aplicación cliente

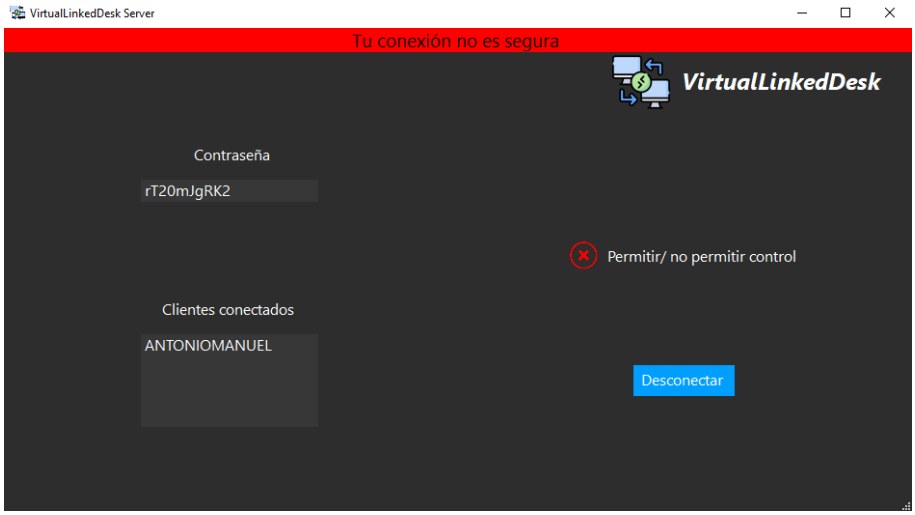


Figura C.9: Aplicación servidor

Como se puede contemplar, el cliente muestra el nombre de la máquina a la que se ha conectado y el servidor qué clientes tiene conectados. Es necesario mencionar que si el usuario no selecciona que se cifre la conexión, se mostrará una barra roja en las dos aplicaciones mostrando el aviso de que la conexión no es segura. Esto se hace con el fin de concienciar al usuario que no debería de transmitir datos sensibles en esa sesión. Si se selecciona conexión cifrada dicha barra no se muestra.

- **Conexión cifrada**

  Para cifrar los datos transmitidos y que la conexión esté autenticada, es necesario instalar la herramienta OpenSSL [58] (la versión destinada a desarrolladores) y con ella generar unos certificados. Con el fin de que en el proceso de generación de certificados no haya errores, se tendrá que configurar la variable de entorno *OPENSSL_CONF* igualándola a la ruta del directorio de instalación de OpenSSL donde se encuentra el fichero *openssl.cfg* (dentro de la carpeta donde se encuentran los binarios). Se accede a este ajuste del sistema escribiendo "Variables de entorno" en la barra de búsqueda, y clicando sobre el botón "Variables de entorno". Posteriormente, se creará la nueva variable del sistema, como en el ejemplo de la Figura C.10.

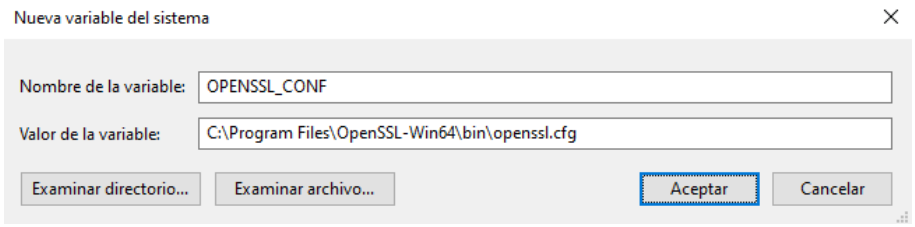


Figura C.10: Edición de las variables de entorno [59]

En algunos casos será necesario también especificar en la variable de entorno *Path* la ruta al directorio donde se encuentra el ejecutable de OpenSSL. Siguiendo el ejemplo de la Figura C.10 sería "C:\Program Files\OpenSSL-Win64\bin\".

Una vez entendida la estructura PKI expuesta en la Subsección 3.3.6, se pueden asimilar y seguir los pasos que habría que seguir en una supuesta conexión entre un cliente (Alice) y un servidor (Bob). Dichos pasos se pueden llevar a cabo ejecutando los scripts de la carpeta *Scripts* del repositorio clonado sin moverlos de sus carpetas específicas. Sus detalles se exponen a continuación:

1. **Bob genera su certificado de rol *CA* y lo instala como Autoridad de Certificación de Confianza en su consola de administración de Windows**.

   Se debe ejecutar el script *CA_Script.bat* mostrado en el Listado C.1, el cual crea una clave de 2048 bits para posteriormente generar un certificado autofirmado con esa clave. Se le pedirá al usuario una contraseña para cifrar la clave.

```
1 @echo off
2
3 openssl genrsa -des3 -out CA.key 2048
4 openssl req -new -x509 -key CA.key -out CA.cer
5
6 EXIT /B
```

Listado C.1: CA_Script.bat

Una vez se genere el certificado, se instalará en la consola de administración de Windows. Para entrar en ella se utilizará la herramienta *Ejecutar* de Windows pudiéndose abrir con las teclas Win + R. En ella se escribirá "mmc" como se especifica en la Figura C.11.

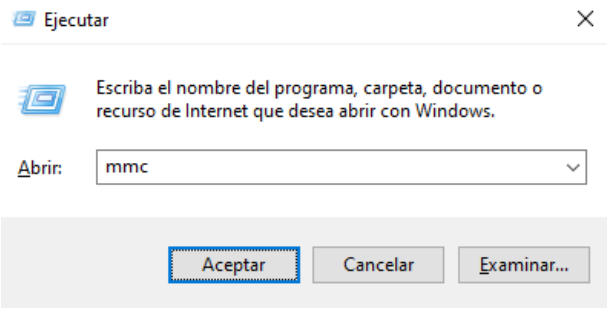


Figura C.11: Ejecutar MMC [59]

Cuando esté abierta la consola, se hará click sobre Archivo > Agregar o quitar complementos y se selecionará *Certificados* como se ve en la Figura C.12. Después se especificará *Certificado de la cuenta del equipo*.

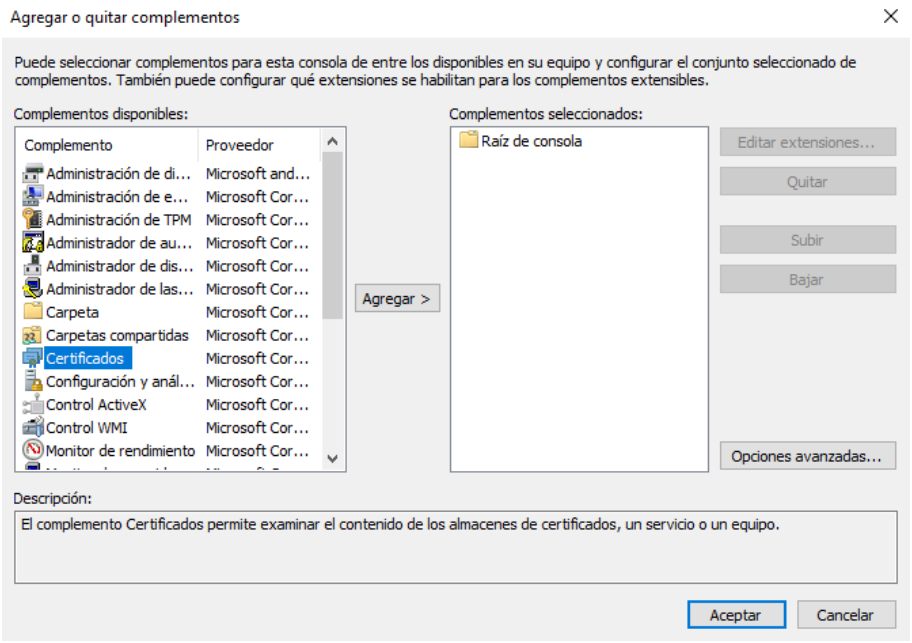


Figura C.12: Complementos MMC [59]

Por último, se dirigirá al directorio de Autoridades de Certificación de Confianza y se hará click en Acción > Todas las tareas > Importar. En este momento se abrirá un asistente como el que se muestra en la Figura C.13, que pedirá que se le especifique la ruta del certificado a importar. Después de importarlo, se guardarán los cambios y el certificado ya estará instalado.

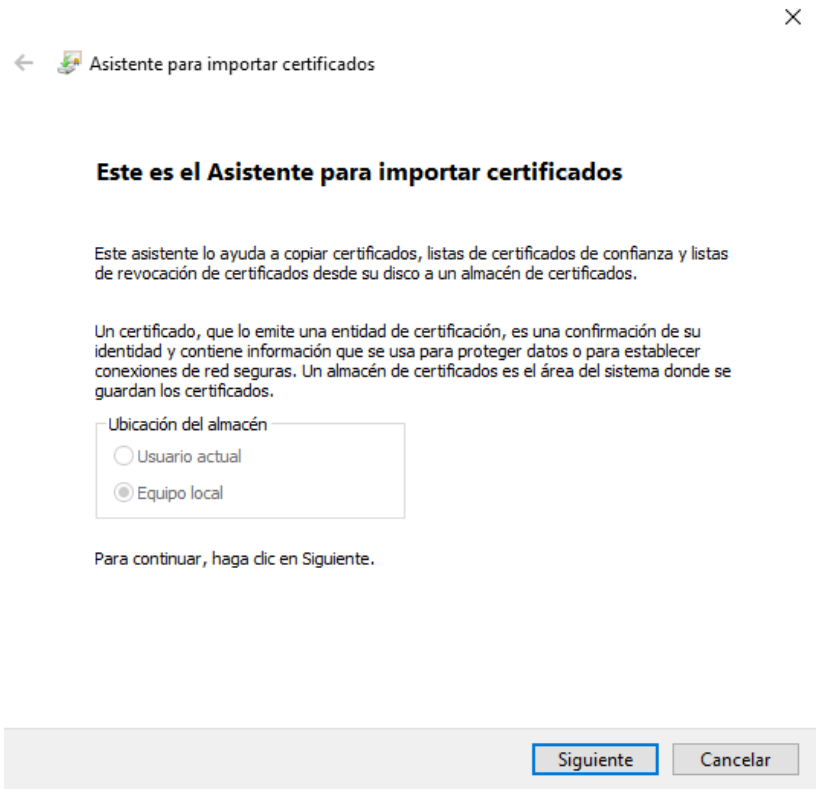


Figura C.13: Importación Certificado [59]

2. **Bob creará una solicitud de certificado de rol *Servidor* y se devuelve a sí mismo el certificado correspondiente (tiene esa capacidad debido a que es también *CA*).**

   Se ha de dirigir a la carpeta *Server* dentro de *Scripts* y ejecutar el script *Server_Script.bat* mostrado en el Listado C.2. Generará su certificado y el envoltorio PKCS #12 correspondiente, envolviendo el certificado con su clave privada.

```
@echo off

openssl genrsa -des3 -out Server.key 2048
openssl req -key Server.key -new -out Server.csr

cd ../CA
openssl x509 -req -in ../Server/Server.csr -CA CA.cer -CAkey CA.key -
    CAcreateserial -out ../Server/Server.cer

cd ../Server
openssl pkcs12 -export -in Server.cer -inkey Server.key -out Server.pfx

EXIT /B
```

Listado C.2: Server_Script.bat

3. **Alice genera una solicitud de certificado y se la envía a Bob para que este lo genere.**

   Alice se dirigirá a la carpeta *Client* dentro de *Scripts* y ejecutará el script *Client_Script_1.bat* mostrado en el Listado C.3.

```
@echo off

openssl genrsa -des3 -out Client.key 2048
openssl req -key Client.key -new -out Client.csr

EXIT /B
```

Listado C.3: Client_Script_1.bat

   Posteriormente, le enviará el fichero generado a Bob por algún canal de comunicación. Bob descargará dentro de su directorio *Client* la solicitud recibida y ejecutará el script *S_Client_Script.bat* expuesto en el Listado C.4 generando el certificado de Alice.

```
@echo off

cd ../CA
openssl x509 -req -in ../Client/Client.csr -CA CA.cer -CAkey CA.key -
    CAcreateserial -out ../Client/Client.cer
cd ../Client

EXIT /B
```

Listado C.4: S_Client_Script.bat

4. **Alice recibe de Bob el certificado que solicitó firmado, junto con el certificado de *CA* de Bob, que deberá instalarlo como Autoridad de Certificación de Confianza en su consola de administración de Windows.**

   Alice podrá recibir su certificado firmado por Bob y el certificado *CA* por cualquier canal de comunicación. Primeramente, tiene que instalarlo siguiendo los mismos pasos que se ha detallado en el paso 1. Posteriormente, se descargará su certificado firmado en su carpeta *Client*, dentro de *Scripts*, y ejecutará el script *Client_Script_2.bat* para generar el envoltorio PKCS #12 pertinente. El contenido del script está expuesto en el Listado C.5.

```
1 @echo off
2
3 openssl pkcs12 -export -in Client.cer -inkey Client.key -out Client.pfx
4
5 EXIT /B
```

Listado C.5: Client_Script_2.bat

Después de realizar los pasos para la generación de los certificados, hay que seguir las mismas indicaciones que se especificaron en la conexión en claro, con la diferencia de que habrá que marcar la casilla *Conexión cifrada* e introducir más parámetros iniciales. En el cliente hay que introducir el nombre esperado en el certificado del servidor, un certificado y la contraseña usada para cifrar la clave privada (de haberla); en el servidor se debe especificar un certificado y su contraseña (también usada, si la hay, para cifrar su clave privada). Los certificados tienen que proporcionarse envueltos con su clave privada en un fichero PKCS #12.

Apéndice D

Detalles de las clases del diseño

Debido al gran tamaño del diagrama de clases expuesto en la Subsección 3.3.1 no se incluyeron detalles de los parámetros de ninguna de las funciones. Cada clase se expone detalladamente en las siguientes figuras:

Bitmap

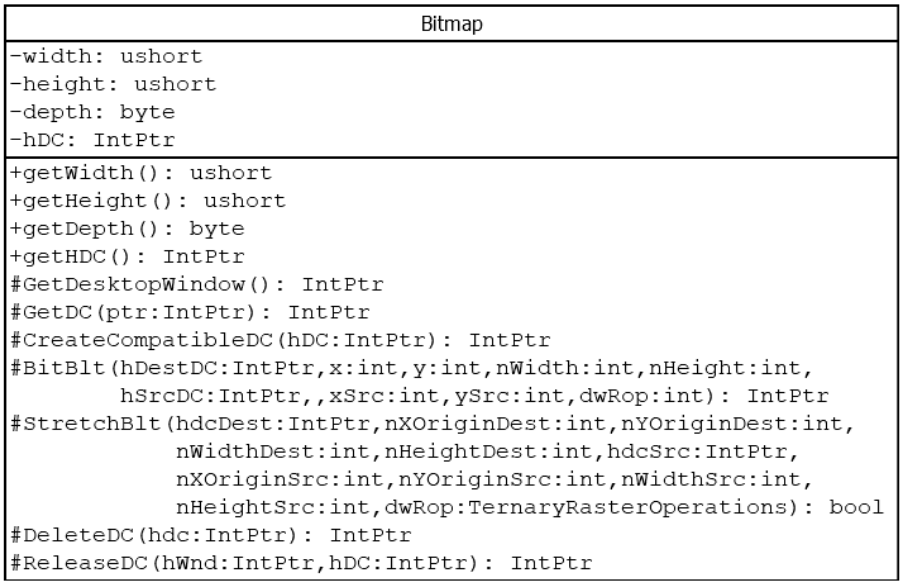


Figura D.1: Detalles clase Bitmap

Todas las funciones de esta clase están explicadas en la documentación de Microsoft, dirigirse a los correspondientes enlaces para más información:

- **GetDesktopWindow** [60].
- **GetDC** [61].
- **CreateCompatibleDC** [62].
- **BitBlt** [63].
- **StretchBlt** [64].

- **DeleteDC** [65].
- **ReleaseDC** [66].

BitmapOffScreen

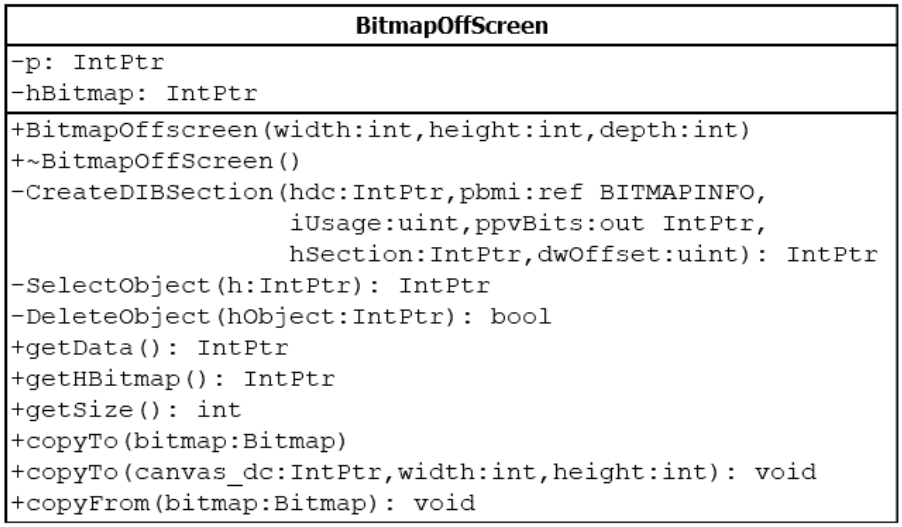


Figura D.2: Detalles clase BitmapOffScreen

A continuación, se dan detalles de cada función:

- **BitmapOffScreen**. Se encarga de construir un mapa de bits en memoria de las dimensiones (ancho x alto) y profundidad de color (en bits por píxel) especificadas.
- **CreateDIBSection** [67].
- **SelectObject** [68].
- **DeleteObject** [69].
- **getData**. Devuelve un puntero al área de datos del mapa de bits.
- **getHBitmap**. Retorna el manejador al mapa de bits que se ha creado.
- **getSize**. Devuelve el tamaño del mapa de bits.
- **copyTo**$_1$. Copia su contenido en otro mapa de bits especificado.
- **copyTo**$_2$. Copia su contenido en un contexto de dispositivo con un alto y ancho de píxeles específico.
- **copyFrom**. Copia los datos del mapa de bits especificado en el actual.

BitmapScreen

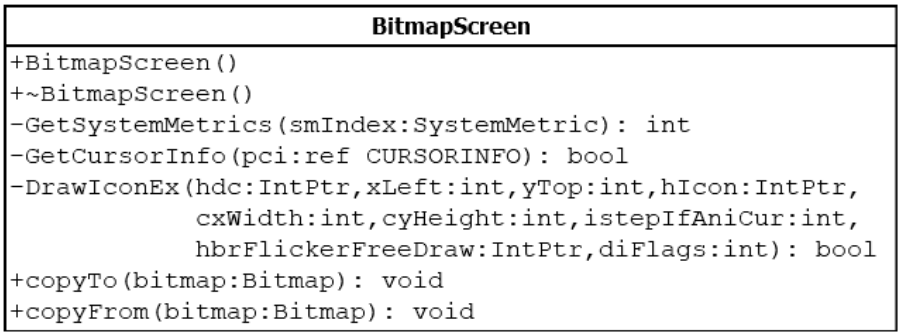


Figura D.3: Detalles clase BitmapScreen

Los detalles de las funciones se explican más abajo:

- **GetSystemMetrics** [70].
- **GetCursorInfo** [71].
- **DrawIconEx** [72].
- **copyTo**. Copia su contenido en otro mapa de bits especificado superponiendo la imagen del cursor del ratón.
- **copyFrom**. Copia los datos del mapa de bits especificado en el actual.

Client

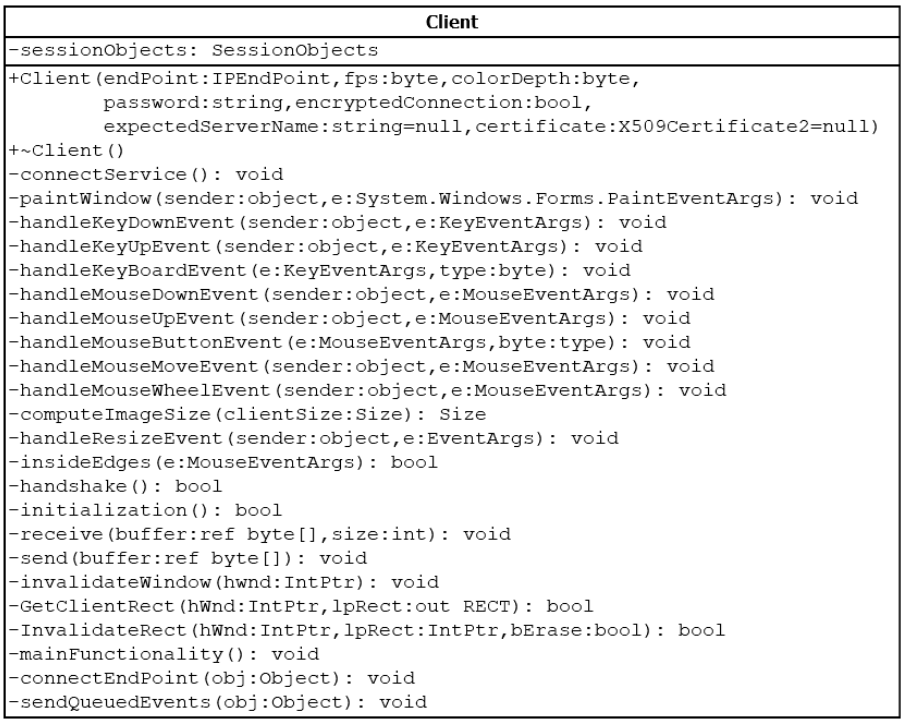


Figura D.4: Detalles clase Client

La explicación de las funciones se muestra a continuación:

- **Client**. Construye un cliente con la información proporcionada por el usuario al inicio de la ejecución. Dicha información se muestra en la Figura C.7.
- **connectService**. Se conecta al servicio del servidor llamando a la función *connectEndPoint*.
- **paintWindow**. Se encarga de repintar la ventana cuando se invaliden los datos mostrados en ella.
- **handleKeyDownEvent**. Se ejecuta cuando se detecta una pulsación de una tecla de teclado. Llamará a *handleKeyBoardEvent* especificando que ha sido una pulsación y no una liberación.
- **handleKeyUpEvent**. Se ejecuta cuando se detecta la liberación de una tecla de teclado. Llamará a *handleKeyBoardEvent* especificando que ha sido una liberación y no una pulsación.
- **handleKeyBoardEvent**. Crea la estructura necesaria para encapsular un evento de tecla de teclado.
- **handleMouseDownEvent**. Se ejecuta cuando se detecta una pulsación de un botón de ratón. Llamará a *handleMouseButtonEvent* especificando que ha sido una pulsación y no una liberación.
- **handleMouseUpEvent**. Se ejecuta cuando se detecta la liberación de un botón de ratón. Llamará a *handleMouseButtonEvent* especificando que ha sido una liberación y no una pulsación.
- **handleMouseButtonEvent**. Crea la estructura necesaria para encapsular un evento de botón de ratón.
- **handleMouseMoveEvent**. Es llamado cuando se mueve el cursor del ratón de posición. Crea la estructura necesaria para encapsular dicho evento.
- **handleMouseWheelEvent**. Se llama cuando se detecta el movimiento de la rueda del ratón. Crea la estructura necesaria para encapsular dicho evento.
- **computeImageSize**. Hace posible que la imagen recibida del servidor se adapte a cualquier tamaño de la ventana del cliente.
- **handleResizeEvent**. Adapta el contenido de la ventana a su nuevo tamaño cuando se detecta una redimensión.
- **insideEdges**. Comprueba que el cursor del ratón esté dentro del área donde es pintada la imagen recibida del servidor.
- **handshake**. Intercambio con el servidor del protocolo y la contraseña.
- **initialization**. Intercambio con el servidor de parámetros de inicialización introducidos por el usuario.
- **receive**. Se trata de una función bloqueante que recibe una cantidad de datos determinada especificados en un buffer.
- **send**. Envía todos los datos contenidos en un buffer determinado.
- **invalidateWindow**. Invalida los datos mostrados en la ventana actual llamando a las funciones *GetClientRect* e *InvalidateRect*.
- **GetClientRect** [73].
- **InvalidateRect** [74].
- **mainFunctionality**. Realiza la funcionalidad principal del programa.

- **connectEndPoint**. Se crea un socket y se conecta a la dirección IP y puerto especificados por el usuario.
- **sendQueuedEvents**. Comprueba si en la cola concurrente de peticiones hay peticiones pendientes para enviar, en caso de que haya, las envía.

Server

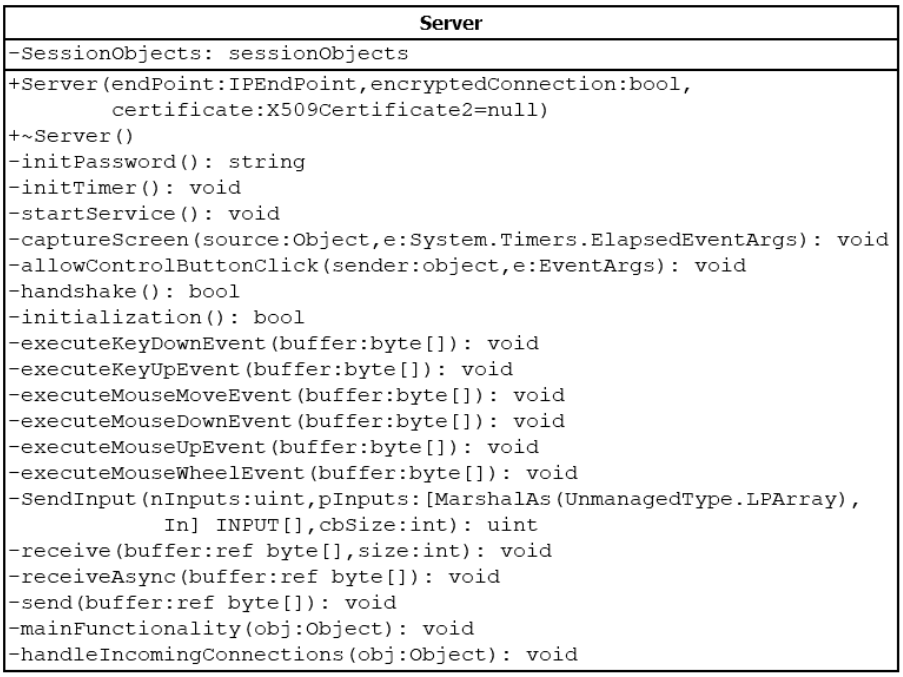


Figura D.5: Detalles clase Server

A continuación, se explican todas las funciones:

- **Server**. Construye un servidor con la información proporcionada por el usuario al inicio de la ejecución. Dicha información se muestra en la Figura C.6
- **initPassword**. Inicializa la contraseña.
- **initTimer**. Inicializa el temporizador que capturará la pantalla a una frecuencia elegida por el usuario.
- **startService**. Empieza la ejecución del servicio pudiendo manejar conexiones entrantes.
- **captureScreen**. Captura los datos de la pantalla y los copia a un buffer.
- **AllowControlButtonClick**. Activa o descativa la posibilidad de controlar el ratón o el teclado del servidor.
- **handshake**. Intercambio con el cliente del protocolo y la contraseña.
- **initialization**. Intercambio con el cliente de parámetros de inicialización introducidos por el usuario.
- **executeKeyDownEvent**. Crea la estructura necesaria para ejecutar el evento de pulsado de una tecla de teclado.
- **executeKeyUpEvent**. Crea la estructura necesaria para ejecutar el evento de liberación de una tecla de teclado.

- **executeMouseMoveEvent**. Crea la estructura necesaria para ejecutar el evento de movimiento del cursor del ratón.
- **executeMouseDownEvent**. Crea la estructura necesaria para ejecutar el evento de pulsado de un botón de ratón.
- **executeMouseUpEvent**. Crea la estructura necesaria para ejecutar el evento de liberación de un botón de ratón.
- **executeMouseWheelEvent**. Crea la estructura necesaria para ejecutar el evento de pulsado de una tecla de teclado.
- **SendInput** [75].
- **receive**. Se trata de una función bloqueante que recibe una cantidad de datos determinada especificados en un buffer.
- **receiveAsync**. Realiza un recibo de datos asíncrono encadenado en el buffer especificado.
- **send**. Envía todos los datos contenidos en un buffer determinado.
- **mainFunctionality**. Realiza la funcionalidad principal del programa.
- **handleIncomingConnections**. Se encarga de manejar toda conexión entrante que se detecte.

Apéndice E

Vista general API Win32

Para poder interactuar con las funciones internas de bajo nivel del sistema operativo de Windows es necesario hacer uso de su API Win32, la cual expone dichas funciones a través de un conjunto de bibliotecas dinámicas [76], tal y como se muestra en la Figura E.1:

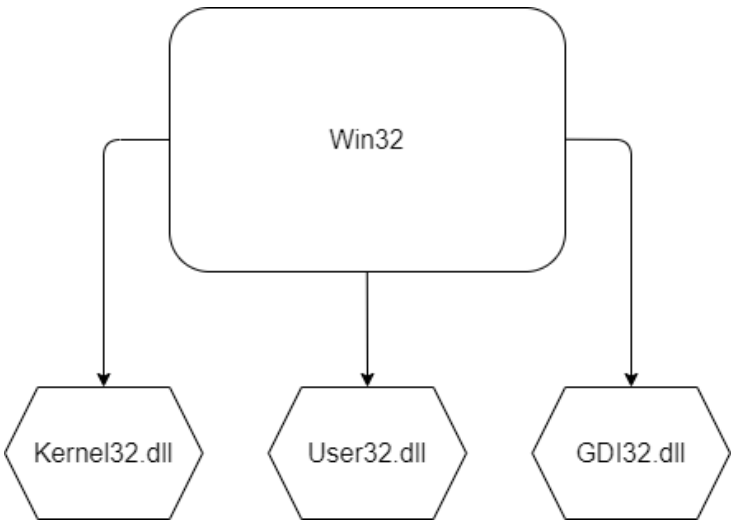


Figura E.1: Estructura Win32

Kernel32.dll

Contiene funciones de bajo nivel del sistema operativo destinadas a la administración de memoria y gestión de recursos como la creación de procesos, hilos, etc. En este proyecto las funciones de esta biblioteca no se han usado, salvo *GetLastError* [77], que ha sido muy útil para depurar debido a que devuelve el último código de error del hilo que la llama.

User32.dll

Abarca todo el control de los elementos de la interfaz de usuario del sistema operativo como las ventanas, menús, sus comunicaciones, etc. Los conceptos que se deben tener claros para

entender el uso de las funciones que ofrece esta biblioteca sobre el software son los siguientes:

- **Partes de una ventana**. Una ventana [78] es un área rectangular de una pantalla que usa una aplicación para mostrar su salida y recibir entradas del usuario. Se divide a su vez en dos importantes áreas [79], el área cliente, que es el que la aplicación maneja su comportamiento y apariencia; y el área no cliente, el cual el sistema operativo se encarga de su control. Este último está formado por la barra de título, botones de minimizado y maximizado, etc. La explicación gráfica se muestra en la Figura E.2

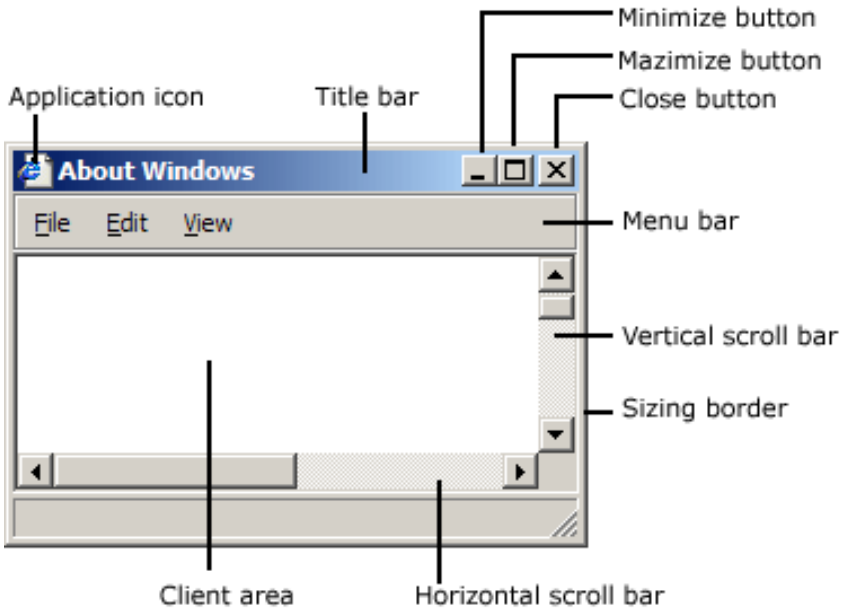


Figura E.2: Partes de una ventana [79]

- **Manejador de ventana (hWnd)**. Es una estructura que identifica de manera unívoca a una ventana [80]. Si se obtiene este manejador sobre una determinada ventana se podrá operar con ella.

- **Coordenadas**. Las coordenadas pueden ser de tipo pantalla, o de tipo cliente [81]. En ambos tipos, el origen de coordenadas es la esquina superior izquierda; la diferencia es que en las de tipo pantalla dicha esquina coincide con la de la pantalla como tal y en las de tipo cliente se refiere a la del área cliente de la aplicación ejecutándose en el momento.
  Es importante destacar que el formato de las coordenadas también puede ser absolutas normalizadas, en tal caso, su unidad no será el píxel, sino un número entre 0 y 65535.

- **Contexto de dispositivo (DC)**. Es una estructura que define objetos gráficos, sus atributos asociados y también los modos gráficos que afectan a la salida de un dispositivo (hDC) [82]. Se pueden obtener manejadores de contexto de dispositivo para realizar operaciones de dibujo sobre ellos. Existen cuatro tipos de contextos de dispositivo [83]:

  - **Pantalla**. Permiten realizar operaciones de dibujo en monitores.
  - **Impresora**. Posibilitan las operaciones de dibujo en las impresoras.

- **Memoria**. Los procedimientos de dibujo los realizan sobre un mapa de bits en lugar de sobre un dispositivo [84], tratando a esa porción de memoria como un dispositivo virtual.
- **Información**. Se utilizan para recuperar información de un dispositivo.

Para el desarrollo del proyecto serán únicamente importantes los contextos de dispositivo de memoria y pantalla.

Una vez entendidas las anteriores ideas, se pueden aplicar sobre el proyecto haciendo uso de funciones como:

- **GetDesktopWindow**. Devuelve un manejador de la ventana raíz o también llamada de escritorio [60]. Dicha ventana [79] es la que el sistema crea cuando se inicia, y sirve como raíz de la jerarquía del sistema de ventanas. Cubre, por tanto, todo el área rectangular sobre el que se expande el escritorio.
- **GetDC**. Retorna un manejador de contexto de dispositivo de una ventana específica pasándosela como parámetro [61]. Una vez se obtenga dicho manejador se podrá realizar operaciones gráficas de la biblioteca de GDI32 sobre ella.
- **ReleaseDC**. Libera un contexto de dispositivo dejando libre recursos, por ejemplo, memoria, asociados a él [66].
- **SendInput**. Hace posible la sintetización de acciones sobre el teclado y el ratón [75], tales como clicks o movimientos del cursor.
- **GetClientRect**. Devuelve las coordenadas del área cliente de una ventana pasándole su manejador correspondiente [73].
- **InvalidateRect**. Invalida el contenido de una ventana concreta [74], dicho contenido será el área encerrado por las coordenadas que se especifican. Esto hará que dicha superficie esté preparada para ser repintada.

GDI32.dll

Sus siglas vienen de Graphics Device Interface (Interfaz de Dispositivo Gráfico) y ofrece funciones de dibujo para salida de vídeo a dispositivos como son las pantallas e impresoras.

Al igual que con User32, para entender las funciones que ofrece esta biblioteca es necesario tener ciertos conceptos claros de los mapas de bits. Un mapa de bits es uno de los objetos gráficos que se pueden seleccionar en un contexto de dispositivo [85] y sirve para crear, manipular y almacenar imágenes en disco [86]. Los conceptos que hay que comprender de estos objetos son los siguientes:

- **Colores de un mapa de bits**. Las imágenes de un mapa de bits pueden ser monocromáticas o policromáticas, cada píxel de una imagen corresponde a uno o más bits en un

mapa de bits [85]. El número máximo de colores de un mapa de bits correspondiente a una imagen se calcula con la fórmula de la Ecuación E.1.

$$2^{Profundidad\ de\ color\ (en\ bits\ por\ píxel)} \tag{E.1}$$

- **Indexación de los datos**. Los datos de los mapas de bits pueden ser indexados de abajo hacia arriba o de arriba hacia abajo [87]. Si se está en el primer caso significa que el primer byte de la memoria es el píxel inferior izquierdo de la imagen, en caso del segundo el primero será el píxel superior izquierdo de la imagen.

- **Tipos de mapas de bits**. Los mapas de bits se pueden clasificar en dos tipos [88]:

  - **Independientes del dispositivo**. Se caracterizan por ser portables al poder usarse en muchas aplicaciones. Este tipo de mapas de bits contienen una tabla de colores en la que se expone la correspondencia de los valores de los píxeles del dispositivo con los valores RGB, por ese motivo son compatibles con cualquier dispositivo [89].
  - **Dependientes del dispositivo**. No son portables como los anteriores, pero son más eficientes sobre todo para ciertas tareas como capturar imágenes. La razón por la que no son portables es porque no tienen una tabla de colores [90]; en su lugar, los colores se codifican en un formato dependiente del dispositivo. Como dos dispositivos distintos pueden tener un conjunto de colores diferente, un mapa de bits de este tipo no se mostrará correctamente en otro equipo.

  Para este proyecto se utilizarán los mapas de bits independientes del dispositivo ya que se transmiten imágenes de una máquina a otra y tienen que visualizarse correctamente.

Después de comprender los conceptos anteriores es necesario estudiar las siguientes funciones que hacen uso de ellos y que son necesarias para el desarrollo de este software:

- **CreateCompatibleDC**. Crea un contexto de dispositivo de memoria especificándole por parámetro un dispositivo concreto [62].

- **CreateDIBSection**. Crea un mapa de bits independiente del dispositivo [67] en el cual las aplicaciones pueden leer/escribir directamente debido a que retorna un puntero a la dirección de los datos del mapa de bits.

- **SelectObject**. Selecciona un objeto en un contexto de dispositivo específico y lo reemplaza por el nuevo objeto del mismo tipo que se le especifica por parámetro [68].

- **BitBlt**. Su nombre viene de la operación gráfica Bit Blit (**Bit Bl**ock **T**ransfer) que consiste en la transferencia de datos entre los mapas de bits asociados a dos contextos de dispositivo (hDC) especificados. [63].

- **StretchBlt**. Al igual que la anterior, realiza la operación Bit Blit siendo capaz de estirar o comprimir la imagen para que se adapte a las dimensiones del área de destino [64].

Con relación a esta biblioteca también se ha ha hecho uso de GDI+ [91], siendo este un envoltorio que proporciona .NET sobre GDI, mejorándolo y extendiéndolo, aportándole nuevas funcionalidades. Para el proyecto se ha aprovechado este envoltorio con el fin de comprimir con JPEG las imágenes enviadas [92], ya que GDI no tiene soporte para trabajar con dicho algoritmo.

Apéndice F

Redes

Una red [93] es un conjunto de máquinas interconectadas a través de un medio con el fin de intercambiar información y compartir recursos. En toda red hay dos roles, un emisor y un receptor, las máquinas en dicha red se intercambian los roles.

Estructura de Internet

Internet (**Inter**national **net**work of computers) [94] es como su propio nombre indica una red global de ordenadores y surge de la red norteamericana Arpanet [2] en el año 1969. Se estructura en un modelo de capas de protocolos situadas unas encima de otras [95], la capa que está en un nivel inferior ofrece su servicio a la que está encima de ella para que esta pueda desempeñar su función y hacer lo mismo con la que está en un nivel superior. Dicho modelo de capas recibe el nombre de pila de protocolos de Internet y su representación OSI consta de los siguientes niveles, estando los cuatro más altos orientados a aplicaciones y los tres más bajos al transporte:

- **Nivel 7 - Capa de aplicación**. Está en contacto directo con programas de usuario final como los de correo electrónico, navegadores web, etc. En esta capa se asegura que los datos que se envíen desde un equipo a través de ella puedan ser interpretados correctamente por la capa de aplicación de otro; es decir, que si se envía un paquete desde una aplicación en concreto, el receptor sepa que es para esa aplicación también. Este tipo de programas que se ejecutan en red pueden tener arquitectura cliente-servidor, peer-to-peer (P2P) o una combinación híbrida. Algunos ejemplos de protocolos que se usan en esta capa son FTP, SMTP, HTTP, etc.
- **Nivel 6 - Capa de presentación**. Se encarga de aplicar un formato estandarizado a los datos para que sean reconocidos entre distintos equipos. Por ejemplo, tipo de cifrado, compresión o convenciones little/big endian.
- **Nivel 5 - Capa de sesión**. Posee mecanismos de gestión y control para regular el establecimiento de la conexión, su mantenimiento y su interrupción. Un protocolo que se sitúa en esta capa es SSL.
- **Nivel 4 - Capa de transporte**. Se lleva a cabo la comunicación lógica entre procesos a

través de sus sockets (dirección IP + puerto). Los protocolos de esta capa son TCP, UDP y TLS.

- **Nivel 3 - Capa de red**. Ofrece la comunicación lógica entre máquinas asignándoles una dirección IP. El encabezado que proporciona esta capa a los paquetes contendrá información de la asignación de rutas y el control del flujo de dichos datos. Los protocolos de esta capa son IP, ICMP, RIP, OSRF, etc.
- **Nivel 2 - Capa de enlace**. Proporciona transferencia fiable de datos entre elementos vecinos a través de un enlace o medio de transmisión físico. Consigue la fiabilidad a través de funciones como reconocimiento, eliminación de errores y control del flujo de datos. El acceso a dicho enlace físico está regulado por protocolos como Ethernet, 802.11i (WiFi) o PPP. Los paquetes en este nivel tienen en sus cabeceras direcciones MAC para identificar origen y destino.
- **Nivel 1 - Capa física**. Se encarga de la transformación analógica de los datos digitales para poder transmitirlos por medios de transmisión físicos como hilos de cobre, fibra o aire.

Es importante mencionar que la pila de protocolos de Internet no solamente posee la representación OSI, también tiene la representación TCP/IP que agrupa las capas de presentación y de sesión en la de aplicación. En la práctica si se necesitan estos servicios deben ser implementados en la capa de aplicación [96].

Un concepto que se debe entender es el del encapsulamiento/desencapsulación, el cual trata de que siempre que un paquete de datos se envíe y pase de una capa superior a una inferior, se le añade un encabezado con el fin de que cuando este paquete se reciba en el otro extremo, se pueda procesar y dirigir correctamente a la aplicación de destino quitando dichos encabezados al subir de nuevo de las capas inferiores a las superiores. Su representación se observa en la Figura F.1.

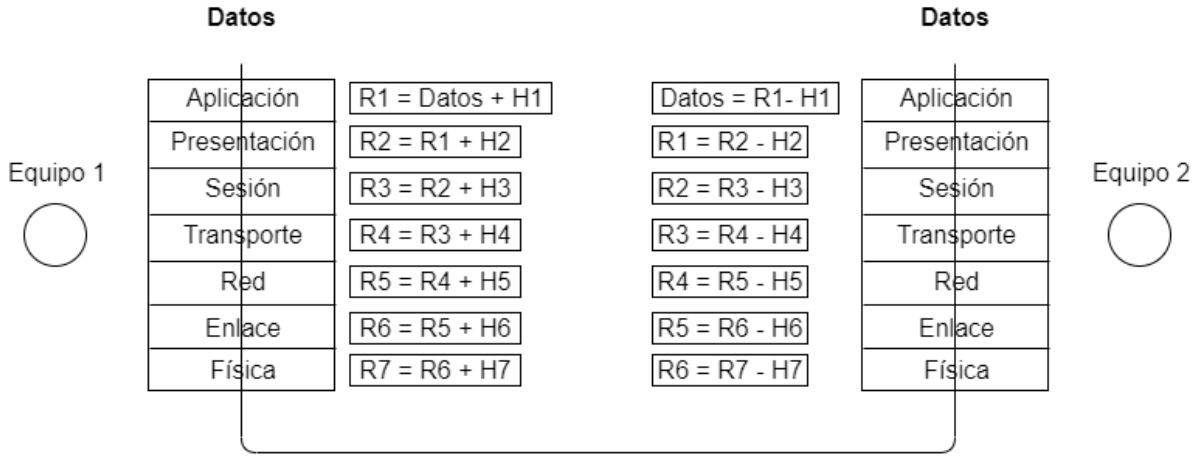


Figura F.1: Encapsulamiento y desencapsulación

Arquitecturas de aplicaciones en red

Existen varios tipos de arquitecturas de aplicaciones que se ejecutan en la red, las más relevantes se detallan a continuación [97, 98]:

- **Cliente-servidor**. Se caracterizan por la existencia de solo dos tipos de roles, el servidor que ofrece un servicio y los clientes que acceden a él. El servidor siempre estará activo con una dirección IP permantente para que los clientes se puedan comunicar con él. Este posee una gran cantidad de recursos para gestionar todas las solicitudes que le lleguen de los clientes conectados, es decir, los recursos están centralizados. Los clientes suelen tener direcciones IP dinámicas y no se comunican entre sí.
  Es importante mencionar que esta arquitectura también se puede aplicar en aplicaciones que se ejecuten en una sola máquina. Las grandes ventajas de esta arquitectura es que los clientes pueden tener pocos recursos debido a que la mayor carga de los procesos se ejecutan en el servidor y que el sistema permite la incorporación de mejoras de los dos tipos de roles por separado. La desventaja es que si el servidor cesa su ejecución debido a cualquier incidencia, los clientes no pueden acceder al servicio; por esta razón, se replican los servidores y también sus datos.

  Existen multitud de aplicaciones con esta arquitectura, como el servicio de correo electrónico Gmail, servidores de alojamiento de archivos como Mega o servidores web a los que se accede a través de un navegador web.

- **Peer-to-peer**. En este tipo no hay un servidor siempre activo como en la anterior. Todos los nodos de la red son pares o iguales y se comportan como clientes y servidores a la vez, consumiendo cada nodo recursos del resto de nodos. Como ventaja se puede destacar la gran escalabilidad del sistema, obteniendo más rendimiento cuantos más nodos haya conectados. Con la arquitectura cliente-servidor sucede lo contrario, teniendo en cuenta que un número reducido de servidores proporciona todos los recursos y el ancho de banda, y que la incorporación de abundantes clientes supondría una pérdida de rendimiento para todos los usuarios conectados. También se puede ver como ventaja la descentralización, no dependiendo de esta manera del correcto funcionamiento del servicio ofrecido por escasos servidores.
  Como desventajas se observa la dificultad de conocer la entidad que domina los recursos al existir una descentralización, y además, la compleja gestión de la red a medida que incrementan los nodos conectados.

  Existen dos tipos de redes peer-to-peer:

  - **Puras**. No existe ningún tipo de centralización, por lo que todos los intercambios de información son de usuario a usuario, pudiéndose enlazar con el apoyo de un nodo, que es otro usuario.
  - **Híbridas**. Existe un servidor central que administra los recursos y la comunicación entre los distintos nodos pero sin conocer la identidad de los mismos y tampoco almacenar información, implicando esto que no se comparten ficheros entre clientes y servidor. Es importante marcar que en el supuesto de que el servidor caiga, los nodos

conectados a él pueden seguir comunicándose debido a que también están conectados entre sí. Asimismo, se puede incluir más de un servidor a esta red.

Como ejemplos de programas peer-to-peer están los famosos de compartición de archivos como Napster, Ares o BitTorrent.

Protocolos de red más relevantes para el proyecto

A continuación, se detallan las características más importantes de los protocolos que se han usado en el proyecto:

- **SSL/TLS**. Son protocolos que proveen capacidad para establecer enlaces autenticados y cifrados entre los equipos de una red [99, 100]. Esto asegura que la información transmitida no pueda ser interceptada ni modificada por entidades no autorizadas. SSL se quedó obsoleto en el año 1999 cuando se lanzó su sucesor TLS, pero se sigue nombrando a la tecnología como *SSL* o *SSL/TLS*. Una de las diferencias a mencionar de los dos protocolos es que SSL trabaja en la capa de sesión mientras que TLS lo hace en la de transporte.

  Un ejemplo de uso de este protocolo es el de la navegación web junto con el protocolo HTTP, provocando su versión segura llamada HTTPS (HTTP sobre SSL). Para aplicarlo en las páginas web, se necesita tener instalado un certificado SSL en el servidor web.

- **TCP/UDP**. Son protocolos que posibilitan el transporte de información entre procesos [97]. Funcionan sobre el protocolo IP, de ahí viene el término comúnmente utilizado *TCP/IP* que significa TCP sobre IP. El transporte lo consiguen añadiéndole a la dirección IP suministrada por el protocolo IP de la capa de red, un número llamado puerto; dicha información agrupada se llama socket, y es la puerta de comunicaciones de un proceso. TCP y UDP se diferencian en los siguientes aspectos:

  - TCP es orientado a conexión y UDP no. Esto significa que TCP requiere una conexión inicial (negociación), que establezca un canal entre cliente y servidor y UDP no lo requiere.
  - TCP garantiza una transferencia de datos fiable asegurando que va a llegar al otro extremo todo lo enviado ordenado y sin corromperse. UDP no asegura esto. Esta característica se consigue con el reenvío de paquetes de verificación cada vez que se recibe algo, de esta manera si se recibe del otro extremo que no le ha llegado la información que esperaba en un tiempo determinado se enviará de nuevo.
  - TCP también ofrece control de flujo y de congestión para que el emisor no sature al receptor y para que se regule al emisor cuando la red se satura. UDP tampoco posee esto.

  Lo que tienen en común los dos protocolos es que ninguno proporciona temporización, ancho de banda garantizado, ni seguridad. Pese a todo, UDP es muy útil y se usa mucho para ciertas tareas como la transmisión de audio y vídeo, debido a que no vale la pena usar TCP y saturar la red con mensajes de confirmación en la transferencia de paquetes

que no es vital que lleguen absolutamente todos y con su respectivo orden, como es el caso del contenido multimedia.
Cabe descatacar que el protocolo DNS en la mayoría de las ocasiones también transmite las peticiones entre cliente y servidor utilizando UDP en vista de que es mucho más veloz.

- **IP**. Es un protocolo usado para la comunicación entre distintas máquinas de una red, la cual se consigue asignándoles una dirección identificativa a cada equipo, llamada dirección IP. La mayor parte de las redes utiliza la versión 4 de este protocolo, IPv4, la cual utiliza 32 bits en sus direcciones. Existe IPv6, que utiliza 128 bits y que se creó para paliar el problema de la estimación del fin de direcciones IPv4, pero esta versión todavía no se ha implantado completamente. La red es capaz de operar con una mezcla de routers IPv4 e IPv6 debido a la tunelización [101], que es un proceso por el cual un datagrama IPv6 es transportado como carga en un datagrama IPv4 entre los routers IPv4.

  El transporte de datos que ofrece el protocolo IP no es fiable debido a que no tiene ningún mecanismo para descubrir si un paquete ha llegado a su destino, necesita los servicios de la capa de transporte para serlo.

Direccionamiento IP en la red global de Internet

En sus comienzos, Internet fue una red muy reducida y no se diseñó pensando en las dimensiones que tiene actualmente [102]. El direccionamiento usado en el protocolo IPv4 da solo la posibilidad de conseguir $2^{32} = 4294967296$ direcciones distintas, por lo que con el aumento masivo de Internet se fueron agotando y se introdujeron distintas soluciones. La primera fue CIDR [101], la cual consigue asignar distintos rangos de direcciones a diferentes redes. Esto lo hace dividiendo el formato de las direcciones IP (con representación en bits) en dos partes, una específica, para identificar la subred en cuestión y otra para determinar los dispositivos. En formato decimal se mostraría como una dirección corriente con una barra seguida del número de bits fijo usado para definir la subred, este número recibe el nombre de máscara de subred. En la Figura F.2 se refleja un ejemplo de esta técnica.

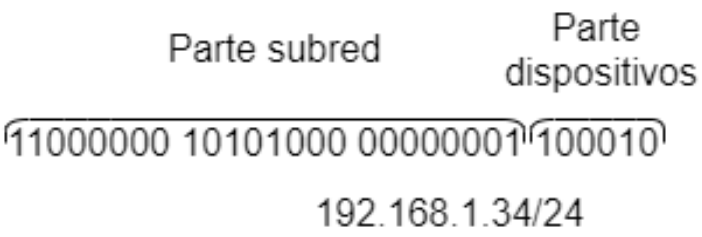


Figura F.2: Ejemplo CIDR

Posteriormente, nació la tecnología NAT [102], la cual se fundamenta en que todos los dispositivos que estén conectados a una misma red hagan uso de un rango de direcciones fijas, llamadas direcciones privadas, y que se conecten a Internet a través de un router utilizando una única dirección común para todos, llamada dirección pública. Esta idea se muestra gráficamente en la Figura F.3.

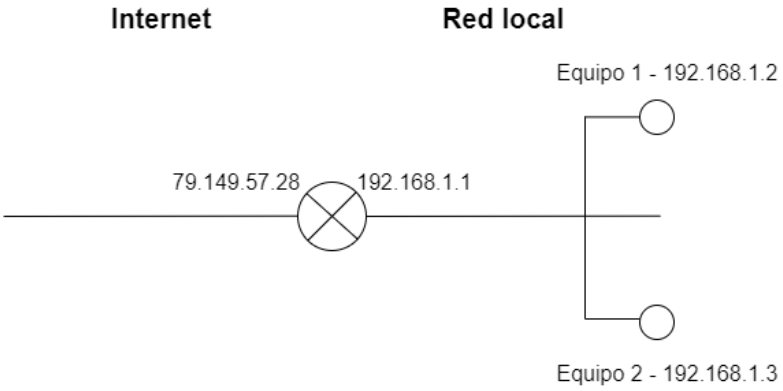


Figura F.3: Idea de la tecnología NAT

En la Figura F.4 se contempla un caso de uso de esta tecnología. Primeramente, un equipo envía un mensaje a otro especificando su dirección IP pública y un puerto. Después, la tecnología NAT se encargará de anotar en una fila de una tabla interna dos datos, estos son la dirección IP y el puerto de la máquina originaria del mensaje, vistos desde dentro de la red local, y la dirección IP y el puerto de esa misma máquina vistos desde fuera de esta red. La nueva dirección IP siempre será la del router, vista desde Internet, y el nuevo puerto se asignará de tal manera que luego se sepa que ese número corresponde a la dirección IP y puerto de la máquina de origen correspondiente, vista desde dentro de la red local.
De esta manera, el proceso que reciba el mensaje saliente tendrá que enviar su respuesta especificando como destino la dirección pública del router y el nuevo puerto asignado. Una vez el router reciba dicho mensaje, se comprobará la tabla y se sabrá que el mensaje va dirigido hacia un proceso en concreto, de la máquina que se mencionó al principio.

Tabla NAT	
Red pública	Red interna
79.149.57.28:7000	192.168.1.2:3000
:	:

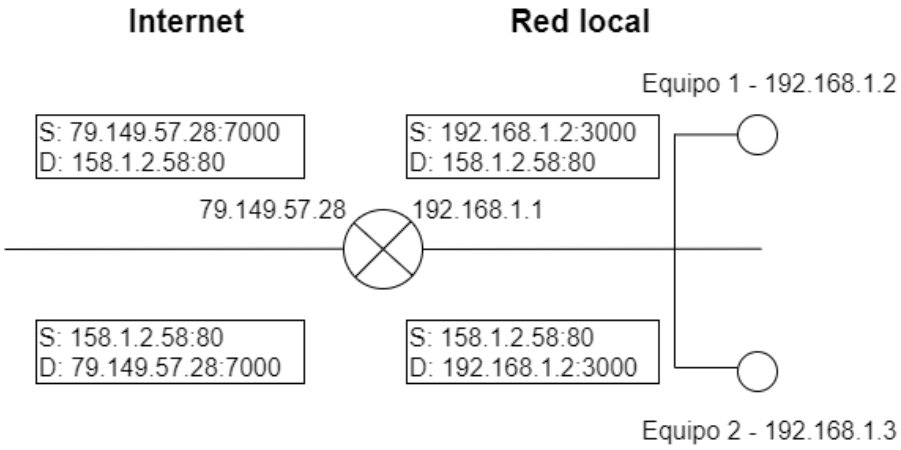


Figura F.4: Funcionamiento de la tecnología NAT

Esta tecnología tiene ciertas ventajas [101] como el enorme ahorro de las direcciones IP que supone su uso, la seguridad añadida, debido a que los equipos dentro de la red local no son visibles desde fuera, y la capacidad de modificar las direcciones IP locales sin notificar al exterior. También tiene desventajas, en vista de que hay aplicaciones o protocolos que no son compatibles con ella y que al estar los equipos de las redes locales invisibles desde fuera, no se puede contactar con ellos sin realizar configuraciones internas o utilizar algunas herramientas adicionales. Las opciones que se tienen para esto son las siguientes:

- **Apertura de puertos manual**. Se trata del proceso de configurar el NAT a mano (su tabla interna), para reenviar el tráfico especificado con la dirección pública del router y un puerto determinado, a un puerto de una máquina en concreto de la red local.
- **UPnP**. Sus siglas vienen de Universal Plug and Play y es un protocolo que sirve para realizar la apertura de puertos de forma automatizada [103]. De esta manera, cualquier programa puede enviar al router una solicitud de apetura de puertos cuando se necesite comunicar con un servidor. En el momento en el que el programa termine su ejecución, se cerrarán otra vez los puertos que haya abierto por razones de seguridad. Este protocolo es muy utilizado en los servicios online de los videojuegos.
- **STUN**. Sus siglas proceden de Simple Traversal of UDP through NAT [104], y se trata de un protocolo que saca provecho de que el protocolo UDP no está orientado a conexión, permitiendo que procesos de máquinas ejecutándose detrás de un NAT descubran cuál es su dirección IP pública, el puerto que pueden utilizar los equipos de fuera de la red local para conectarse a ellos y también, el tipo de NAT al que están siendo sometidos.
  Su funcionamiento es simple. Primeramente, el proceso del equipo que quiere ser públicamente visible contacta con un servidor STUN en Internet, preguntándole qué dirección IP pública y puerto se le asignará; una vez este le responda, el equipo tendrá esta información y podrá dársela a los clientes que deseen conectarse a él para que puedan hacerlo.

  Es importante saber que STUN no funcionará correctamente con los routers que utilicen NAT simétrico, debido a que este tipo de NAT crea un nuevo mapeo de dirección IP y puerto cada vez que una máquina de una red local se conecta con una que se encuentre fuera de la red.
- **Servidor de relevos**. En esta solución, la conexión no se realiza únicamente entre los dos nodos (cliente y servidor) que desean contactarse, sino que ambos nodos actúan como clientes de un servidor que les ofrece la posibilidad de comunicarse. Este servidor es públicamente visible para todos. Primeramente, el nodo que inicia la conexión contacta con este servidor, en vez de con el nodo que quiere comunicarse directamente; después, el otro hace lo mismo, estando en ese momento ambos enlazados. La función de este servidor es la de redirigir el tráfico entre sus clientes.

  Es importante comentar que en un entorno real se utilizarían varios servidores, en lugar de uno solo. Programas como TeamViewer utilizan este método para efectuar sus conexiones.

Apéndice G

Seguridad informática

Principios de la seguridad de la información

La seguridad de la información es el conjunto de técnicas y medidas que aplica una institución para proteger la información basándose en tres principios [105], como se muestra en la Figura G.1, popularmente conocida como triángulo CIA.

Figura G.1: Triángulo CIA

Cada principio se encarga de garantizar algo distinto:

- **Confidencialidad**. Se ocupa de que solamente las entidades autorizadas puedan acceder a la información. Para conseguirlo se utiliza el cifrado.
- **Integridad**. Se encarga de denegar la modificación no autorizada de información. Para lograrlo se hace uso de la firma digital.
- **Disponibilidad**. Trata de garantizar el acceso a la información siempre que lo necesiten los usuarios autorizados. Se consigue por medio de la autenticación.

Criptografía

Primeramente, según apareció la criptografía, se definió como la disciplina que estudia los principios, métodos y medios de transformar los datos para ocultar su significado [106], centrándose esta definición clásica solamente en cumplir el fundamento de la confidencialidad. La definición moderna de la criptografía la describe como la disciplina que estudia los principios, métodos y medios de transformar los datos para ocultar su significado, garantizar su integridad, establecer su autenticidad y prevenir su repudio [106]. Esta última definición se centra en garantizar las propiedades de la confidencialidad e integridad.

Cifrado

Es importante entender que no es lo mismo cifrar que codificar. El concepto de codificar trata de la aplicación de un algoritmo de cambio de representación de lenguaje para ocultar la información sin añadirle ninguna seguridad (por ejemplo, cambio de representación decimal a hexadecimal). Por el contrario, la idea de cifrar trata de emplear un algoritmo que necesita una clave de cifrado para funcionar, ocultando de esta manera los datos y añadiendo seguridad debido a que para luego descifrar esos datos hace falta dicha clave.

Hay distintos métodos de cifrado [107]:

- **Simétrico**. Tanto el emisor como el receptor utilizan la misma clave para cifrar y descifrar los datos que se transmiten. Este método tiene la desventaja de que es necesario el uso de un canal seguro para transmitir la clave de cifrado entre emisor y receptor al comenzar la comunicación.
- **Asimétrico**. Se utilizan dos tipos de claves distintas, una pública que es conocida por todas las entidades y una privada que solo la conoce su poseedor. El proceso consiste en que cuando el emisor quiere transmitir un mensaje al receptor, debe cifrarlo con la clave pública del destinatario, y este para descifrarlo, tiene que utilizar su propia clave privada.

  La clave pública y la privada de una entidad están relacionadas matemáticamente entre sí, siendo esta relación muy complicada de descubrir para un atacante.

Firma

Las propiedades de una firma manual son [108]:

- **Fácil y barata de producir**
- **Fácil de reconocer**
- **Imposible de rechazar por el propietario**
- **Infalsificable teóricamente**

La firma digital cumple las mismas propiedades pero no puede ser siempre la misma debido a que sería fácil de ser falsificada. Garantiza los principios de autenticidad, integridad y también el no repudio, pero no asegura la confidencialidad.

Para aplicar una firma digital se hace uso de criptografía asimétrica, la diferencia con el cifrado asimétrico es que para firmar se usa la clave privada del emisor y para que el receptor verifique la firma se utiliza la clave pública del remitente.

Infraestructura de clave pública (PKI)

Una infraestructura de clave pública [109] es un sistema de recursos, políticas y servicios que permite el uso del cifrado asimétrico con el fin de autenticar a las entidades participantes en una transacción. Por lo general, posee dos tipos de entidades:

- **Autoridad de Certificación (CA)**. Es una entidad de confianza que proporciona los servicios de emisión, validación y revocación de certificados digitales y distribución de claves públicas.
- **Autoridad de Registro (RA)**. Es una entidad que identifica de forma inequívoca al solicitante de un certificado y que verifica los datos que ha proporcionado este con el fin de, si están correctos, suministrárselos a la CA para que emita el certificado [110].

Una PKI también se puede describir como una jerarquía fiable para la gestión de certificados digitales [109]. Su representación gráfica es la reflejada en la Figura G.2.

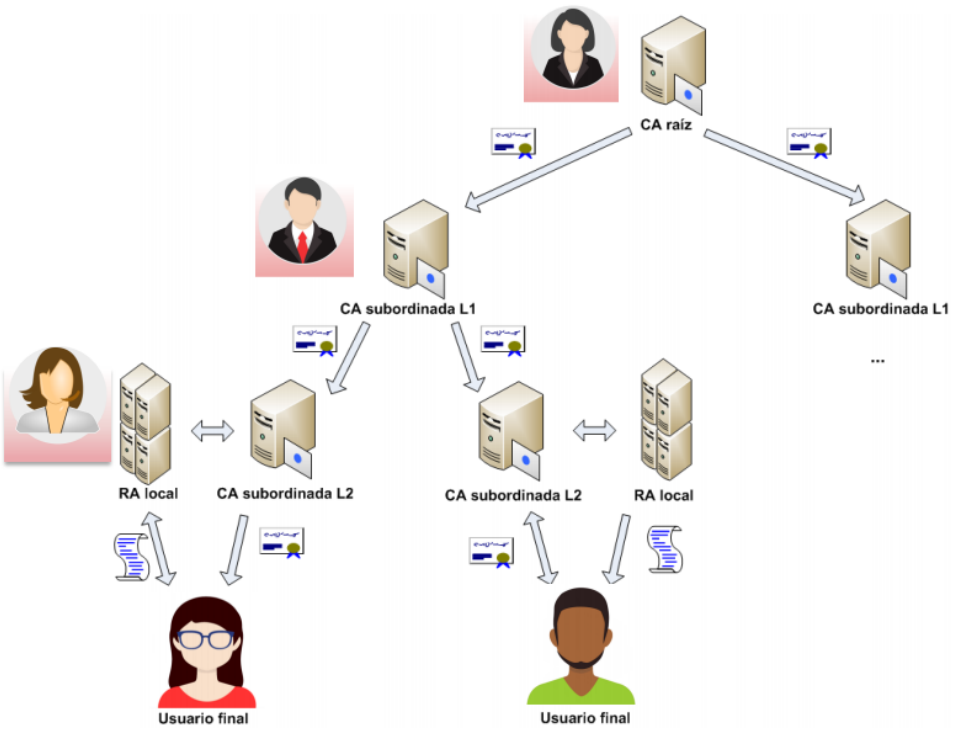


Figura G.2: Representación PKI [111]

Un certificado digital [112] es un documento que sirve para confirmar la identidad de una entidad certificando que su clave pública pertenece a ella, por lo que protegen de la impersonalización. Los certificados siempre son emitidos por una entidad emisora de certificados (CA).

X.509 [113] es un formato estándar para certificados de clave pública y es en lo que se basa la tecnología PKI. Un certificado X.509 consta de los siguientes campos [114]:

- **Versión de X.509 utilizada**.
- **Número de serie**. Un número entero positivo que identifica de forma exclusiva el certificado.
- **Algoritmo de firma utilizado**.
- **Emisor del certificado**. Da información sobre la CA que ha emitido el certificado, tal y como el nombre común y la dirección del sitio donde se encuentra dicha entidad.
- **Periodo de validez**.
- **Sujeto**. Expone el mismo tipo de información que el campo del emisor pero esta vez del sujeto titular del certificado.
- **Información de clave pública del sujeto**. Contiene el algoritmo de clave pública usado y la propia clave pública.
- **Identificador único de emisor (opcional)**.
- **Identificador único de sujeto (opcional)**.
- **Extensiones (opcional)**. Se puede utilizar para vincular el certificado a múltiples identidades.
- **Algoritmo usado para firmar el certificado**.
- **Firma digital del certificado**.

Para poder validar un certificado es necesario validar toda su cadena de certificación hasta llegar a un certificado raíz autofirmado en el que se confíe [111]. En algunos entornos, como es el caso de .NET, no solo basta con realizar esta validación, sino que se exige una verificación de entidad propia implicando con esto la posesión de la clave privada. Por ello hay que generar un fichero PKCS #12 envolviendo el certificado generado con la clave privada correspondiente.

Una vez estudiados los conceptos anteriores es necesario entender que para que las conexiones de este software estén cifradas y autenticadas es imprescindible tener una entidad de confianza (CA) instalada en los equipos que participarán en la conexión. Los usuarios que controlen estos dispositivos se encargarán de generar y enviar solicitudes de certificados a dicha CA para que posteriormente esta cree los certificados y se los devuelva.

Bibliografía

[1] J. Castromil. "Por qué el Xerox Alto es el origen de los ordenadores modernos". https://clipset.com/por-que-el-xerox-alto-es-el-origen-de-los-ordenadores-modernos/. Accedido el 15-07-2021.

[2] Digital Guide IONOS. "ARPANET: los primeros pasos de Internet". https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/arpanet-los-inicios-de-internet/. Accedido el 24-06-2021.

[3] Ordenadores y portátiles. "¿Qué es Arpanet?" https://www.ordenadores-y-portatiles.com/que-es-arpanet/. Accedido el 24-06-2021.

[4] Estréllate y Arde. "El entorno gráfico X Window System". http://www.estrellateyarde.org/el-entorno-grafico-x-window-system. Accedido el 10-06-2021.

[5] J. A. Castillo. "Telnet qué es y para qué sirve [La más completa]". https://www.profesionalreview.com/2019/01/20/telnet-que-es/. Accedido el 07-06-2021.

[6] C. Campo y C. Gacría Rubio. "Servicio de terminal remoto". https://www.cartagena99.com/recursos/alumnos/apuntes/4_telnet-rlogin_v4.pdf. Accedido el 07-06-2021.

[7] SSH. "rlogin - the legacy remote login tool". https://www.ssh.com/academy/ssh/rlogin. Accedido el 08-06-2021.

[8] D. C. "¿Cómo funciona el SSH?" https://www.hostinger.es/tutoriales/que-es-ssh. Accedido el 08-06-2021.

[9] E. C. Anna Frederick. "Looking back at Project Athena". https://news.mit.edu/2018/mit-looking-back-project-athena-distributed-computing-for-students-1111. Accedido el 10-06-2021.

[10] X.Org. "X Window System Protocol". https://www.x.org/releases/X11R7.5/doc/x11proto/proto.pdf. 1986. Accedido el 10-06-2021.

[11] S. Romero. "Introducción al sistema X Window". http://www.sromero.org/ext/articulos/xwin/xwin1.html. Accedido el 10-06-2021.

[12] The Open Group. "The X Window System". http://www.opengroup.org/tech/desktop/x-window-system/. Accedido el 10-06-2021.

[13] IP Adress Info. "The VNC family of Remote Control Applications". http://ipinfo.info/html/vnc_remote_control.php. Accedido el 12-06-2021.

[14] B. Mitchell. "The 6 Best Virtual Network Computing (VNC) Software". https://www.lifewire.com/vnc-free-software-downloads-818116. Accedido el 12-06-2021.

[15] TightVNC. "TightVNC Software". https://www.tightvnc.com/. Accedido el 12-06-2021.

[16] TigerVNC. "TigerVNC". https://tigervnc.org/. Accedido el 12-06-2021.

[17] UltraVNC Team. "UltraVNC Remote PC support". https://www.uvnc.com/. Accedido el 12-06-2021.

[18] RealVNC. "RealVNC". https://www.realvnc.com/en/. Accedido el 12-06-2021.

[19] The Remote Framebuffer Protocol. "IETF RFC 6143". https://datatracker.ietf.org/doc/html/rfc6143. 2011. Accedido el 30-11-2020.

[20] Glosario IT. "Framebuffer - Sección Informática". https://www.glosarioit.com/Framebuffer. Accedido el 30-11-2020.

[21] Digital Guide IONOS. "Terminal server: qué es y cuáles son sus ventajas". https://www.ionos.es/digitalguide/servidores/know-how/terminal-server-que-hay-tras-los-servidores-remotos/. Accedido el 16-06-2021.

[22] Tecnozero. "Terminal Server". https://www.tecnozero.com/blog/terminal-server/. Accedido el 16-06-2021.

[23] Microsoft. "Remote Desktop Protocol, Microsoft MS-RDPBCGR". https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rdpbcgr/5073f4ed-1e93-45e1-b039-6e30c385867c. 2021. Accedido el 30-11-2020.

[24] TeamViewer. "TeamViewer". https://www.teamviewer.com/es/. Accedido el 20-06-2021.

[25] Google. "Escritorio Remoto de Chrome". https://remotedesktop.google.com/home. Accedido el 20-06-2021.

[26] Microsoft. "Te damos la bienvenida a tu PC en la nube de Windows 365". https://www.microsoft.com/es-ww/windows-365. Accedido el 16-07-2021.

[27] G. González. "Linux y macOS suben, Windows baja: así de raras están las cuotas de mercado de los sistemas operativos de escritorio". https://www.genbeta.com/sistemas-operativos/linux-macos-suben-windows-baja-asi-raras-estan-cuotas-mercado-sistemas-operativos-escritorio. Accedido el 05-11-2020.

[28] Digital Guide IONOS. "El modelo en cascada: desarrollo secuencial de software". https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/. Accedido el 12-12-2020.

[29] Microsoft. "Virtual-Key Codes". https://docs.microsoft.com/en-us/windows/win32/inputdev/virtual-key-codes. Accedido el 05-09-2021.

[30] Microsoft. "MOUSEINPUT structure (winuser.h)". https://docs.microsoft.com/en-us/windows/win32/api/winuser/ns-winuser-mouseinput. Accedido el 05-09-2021.

[31] J. M. Aguilar. "10 Diferencias entre .NET Core y .NET Framework". https://www.campusmvp.es/recursos/post/10-diferencias-entre-net-core-y-net-framework.aspx. Accedido el 15-07-2021.

[32] Jobted. "Sueldo del Analista de Sistemas en España". https://www.jobted.es/salario/analista-sistemas. Accedido el 30-07-2021.

[33] Jobted. "Sueldo del Analista Programador en España". https://www.jobted.es/salario/analista-programador. Accedido el 30-07-2021.

[34] Jobted. "Sueldo del Programador en España". https://www.jobted.es/salario/programador. Accedido el 30-07-2021.

[35] Cuestiones laborales. "Cómo calcular la jornada de trabajo". https://www.cuestioneslaborales.es/como-calcular-la-jornada-de-trabajo/. Accedido el 30-07-2021.

[36] Agencia Tributaria. "Tabla de coeficientes de amortización lineal". https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml. Accedido el 30-07-2021.

[37] Comisión Nacional de los Mercados y la Competencia. "El gasto promedio de los españoles en electricidad es de 56,3 euros al mes por hogar". https://www.cnmc.es/eu/node/271580. Accedido el 30-07-2021.

[38] Movistar. "Fusión Inicia". https://www.movistar.es/particulares/fusion/inicia. Accedido el 30-07-2021.

[39] Jefatura del Estado. "Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales". https://www.boe.es/buscar/pdf/2018/BOE-A-2018-16673-consolidado.pdf. 2018. Accedido el 05-04-2021.

[40] Letslaw. "Ley de Protección de Datos ¿En qué consiste y cómo te afecta?" https://letslaw.es/ley-de-proteccion-de-datos/. Accedido el 05-04-2021.

[41] Ekon. "¿Qué es y cómo afecta el RGPD?" https://www.ekon.es/rgpd/. Accedido el 05-04-2021.

[42] Unión Europea. "Reglamento general de protección de datos". https://www.boe.es/doue/2016/119/L00001-00088.pdf. Accedido el 05-04-2021.

[43] A. M. Hernández De León. "VirtualLinkedDesk". https://github.com/Tony450/VirtualLinkedDesk. Accedido el 07-09-2021.

[44] GNU Operating System. "GNU General Public License". https://www.gnu.org/licenses/gpl-3.0.html. Accedido el 30-06-2021.

[45] Microsoft. "Programming reference for the Win32 API". https://docs.microsoft.com/en-us/windows/win32/api/. Accedido el 05-11-2020.

[46] Common Language Infrastructure (CLI). "Common Language Infrastructure (CLI), ECMA International ECMA-335". https://www.ecma-international.org/wp-content/uploads/ECMA-335_6th_edition_june_2012.pdf. 2012. Accedido el 15-07-2021.

[47] Mono Project. "Cross platform, open source .NET framework". https://www.mono-project.com/. Accedido el 15-07-2021.

[48] JPEG (Joint Photographic Experts Group). "JPEG (Joint Photographic Experts Group), ITU Recommendation T.81". https://www.w3.org/Graphics/JPEG/itu-t81.pdf. 1992. Accedido el 15-07-2021.

[49] Sustainability of Digital Formats: Planning for Library of Congress Collections. "Bitmap Image File (BMP), Version 5". https://loc.gov/preservation/digital/formats/fdd/fdd000189. Accedido el 15-07-2021.

[50] Microsoft. "Bitmaps (Windows GDI)". https://docs.microsoft.com/en-us/windows/win32/gdi/bitmaps. Accedido el 10-11-2020.

[51] Internet X.509 Public Key Infrastructure Certificate y Certificate Revocation List (CRL) Profile. "IETF RFC 5280". https://datatracker.ietf.org/doc/html/rfc5280. 2008. Accedido el 10-05-2021.

[52] PKCS #12: Personal Information Exchange Syntax v1.1. "IETF RFC 7292". https://datatracker.ietf.org/doc/html/rfc7292. 2014. Accedido el 10-05-2021.

[53] Remote Authentication Dial In User Service (RADIUS). "IETF RFC 2865". https://datatracker.ietf.org/doc/html/rfc2865. 2000. Accedido el 12-06-2021.

[54] J. López Gómez. "SRS-latex-uc3m". https://github.com/jalopezg-r00t/SRS-latex-uc3m. Accedido el 20-05-2021.

[55] Microsoft. "Visual Studio Best-in-class tools for any developer". https://visualstudio.microsoft.com/. Accedido el 06-11-2020.

[56] Github. "Where the world builds software". https://github.com/. Accedido el 03-09-2021.

[57] Microsoft. "Download .NET". https://dotnet.microsoft.com/download. Accedido el 15-07-2021.

[58] Shining Light Productions. https://slproweb.com/products/Win32OpenSSL.html. Win32/Win64 OpenSSL. Accedido el 01-05-2021.

[59] Microsoft. "New features. Updated security. Windows 10 keeps getting better all the time." https://www.microsoft.com/en-us/windows/get-windows-10. Accedido el 03-09-2021.

[60] Microsoft. "GetDesktopWindow function (winuser.h)". https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getdesktopwindow. Accedido el 25-11-2020.

[61] Microsoft. "GetDC function (winuser.h)". https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getdc. Accedido el 25-11-2020.

[62] Microsoft. "CreateCompatibleDC function (wingdi.h)". https://docs.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-createcompatibledc. Accedido el 18-11-2020.

[63] Microsoft. "BitBlt function (wingdi.h)". https://docs.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-bitblt. Accedido el 18-11-2020.

[64] Microsoft. "StretchBlt function (wingdi.h)". https://docs.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-stretchblt. Accedido el 30-04-2021.

[65] Microsoft. "DeleteDC function (wingdi.h)". https://docs.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-deletedc. Accedido el 20-03-2021.

[66] Microsoft. "ReleaseDC function (winuser.h)". https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-releasedc. Accedido el 25-11-2020.

[67] Microsoft. "CreateDIBSection function (wingdi.h)". https://docs.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-createdibsection. Accedido el 18-11-2020.

[68] Microsoft. "SelectObject function (wingdi.h)". https://docs.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-selectobject. Accedido el 18-11-2020.

[69] Microsoft. "DeleteObject function (wingdi.h)". https://docs.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-deleteobject. Accedido el 20-03-2021.

[70] Microsoft. "GetSystemMetrics function (winuser.h)". https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getsystemmetrics. Accedido el 20-03-2021.

[71] Microsoft. "GetCursorInfo function (winuser.h)". https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getcursorinfo. Accedido el 20-03-2021.

[72] Microsoft. "DrawIconEx function (winuser.h)". https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-drawiconex. Accedido el 20-03-2021.

[73] Microsoft. "GetClientRect function (winuser.h)". https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getclientrect. Accedido el 14-11-2020.

[74] Microsoft. "InvalidateRect function (winuser.h)". https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-invalidaterect. Accedido el 14-11-2020.

[75] Microsoft. "SendInput function (winuser.h)". https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-sendinput. Accedido el 20-03-2021.

[76] Microsoft. "Identifying Functions in DLL". https://docs.microsoft.com/en-us/dotnet/framework/interop/identifying-functions-in-dlls?redirectedfrom=MSDN. Accedido el 10-11-2020.

[77] Microsoft. "GetLastError function (errhandlingapi.h)". https://docs.microsoft.com/en-us/windows/win32/api/errhandlingapi/nf-errhandlingapi-getlasterror. Accedido el 03-09-2021.

[78] Microsoft. "Windows (Windows and Messages)". https://docs.microsoft.com/en-us/windows/win32/winmsg/windows. Accedido el 12-11-2020.

[79] Microsoft. "About Windows". https://docs.microsoft.com/en-us/windows/win32/winmsg/about-windows. Accedido el 12-11-2020.

[80] Microsoft. "What Is a Window?" https://docs.microsoft.com/en-us/windows/win32/learnwin32/what-is-a-window-. Accedido el 12-11-2020.

[81] Microsoft. "Windows Forms Coordinates". https://docs.microsoft.com/en-us/dotnet/desktop/winforms/windows-forms-coordinates?view=netframeworkdesktop-4.8. Accedido el 18-02-2021.

[82] Microsoft. "Device Contexts". https://docs.microsoft.com/en-us/windows/win32/gdi/device-contexts. Accedido el 14-11-2020.

[83] Microsoft. "Device Context Types". https://docs.microsoft.com/en-us/windows/win32/gdi/device-context-types. Accedido el 14-11-2020.

[84] Microsoft. "Memory Device Contexts". https://docs.microsoft.com/en-us/windows/win32/gdi/memory-device-contexts. Accedido el 14-11-2020.

[85] Microsoft. "About Bitmaps". https://docs.microsoft.com/en-us/windows/win32/gdi/about-bitmaps. Accedido el 14-11-2020.

[86] Microsoft. "Bitmaps (Windows GDI)". https://docs.microsoft.com/en-us/windows/win32/gdi/bitmaps. Accedido el 14-11-2020.

[87] Microsoft. "Top-Down vs. Bottom-Up DIBs". https://docs.microsoft.com/en-us/windows/win32/directshow/top-down-vs--bottom-up-dibs. Accedido el 16-11-2020.

[88] Microsoft. "Bitmap Classifications". https://docs.microsoft.com/en-us/windows/win32/gdi/bitmap-classifications. Accedido el 16-11-2020.

[89] Microsoft. "Device-Independent Bitmaps". https://docs.microsoft.com/en-us/windows/win32/gdi/device-independent-bitmaps. Accedido el 16-11-2020.

[90] Microsoft. "Device-Dependent Bitmaps". https://docs.microsoft.com/en-us/windows/win32/gdi/device-dependent-bitmaps. Accedido el 16-11-2020.

[91] Microsoft. "About GDI+ Managed Code". https://docs.microsoft.com/en-us/dotnet/desktop/winforms/advanced/about-gdi-managed-code?view=netframeworkdesktop-4.8. Accedido el 24-05-2021.

[92] Microsoft. "Images, Bitmaps, and Metafiles". https://docs.microsoft.com/es-es/dotnet/desktop/winforms/advanced/images-bitmaps-and-metafiles?view=netframeworkdesktop-4.8. Accedido el 24-05-2021.

[93] RedUsers. "¿Qué es una red informática?" https://www.redusers.com/noticias/que-es-una-red-informatica/. Accedido el 09-04-2021.

[94] Significados. "Significado de Internet". https://www.significados.com/internet/. Accedido el 02-09-2021.

[95] Digital Guide IONOS. "¿Qué es el modelo OSI?" https://www.ionos.es/digitalguide/servidores/know-how/el-modelo-osi-un-referente-para-normas-y-protocolos/. Accedido el 14-06-2021.

[96] J. F. Kurose y K. W. Ross (2016). "Chapter 1 Introduction". https://www-net.cs.umass.edu/kurose-ross-ppt-7e/Chapter_1_V7.01.ppt. Accedido el 14-06-2021.

[97] J. F. Kurose y K. W. Ross (2016). "Chapter 2 Application Layer". https://www-net.cs.umass.edu/kurose-ross-ppt-7e/Chapter_2_V7.01.ppt. Accedido el 14-06-2021.

[98] M. José. "Redes P2P ¿Qué son, cuales son sus ventajas y desventajas y que tipos hay?" https://internetpasoapaso.com/p2p/. Accedido el 15-06-2021.

[99] Equipo de soporte de SSL. "¿Qué es SSL?" https://www.ssl.com/es/preguntas-frecuentes/faq-que-es-ssl/. Accedido el 15-06-2021.

[100] Redalia. "Qué es el protocolo SSL/TLS". https://www.redalia.es/ssl/protocolo-ssl/. Accedido el 15-06-2021.

[101] J. F. Kurose y K. W. Ross (2016). "Chapter 4 Network Layer: The Data Plane". https://www-net.cs.umass.edu/kurose-ross-ppt-7e/Chapter_4_V7.01.ppt. Accedido el 14-06-2021.

[102] J. L. Alcoba. "NAT (Network Address Translation): Qué es y cómo funciona". https://www.xatakamovil.com/conectividad/nat-network-address-translation-que-es-y-como-funciona. Accedido el 23-06-2021.

[103] C. González. "Qué es UPnP y para qué sirve esta función del router". https://www.adslzone.net/2018/05/03/upnp-que-es/. Accedido el 23-06-2021.

[104] Ernesto. "El protocolo STUN - Parte 1". https://www.3cx.es/blog/protocolo-stun-1/. Accedido el 23-06-2021.

[105] A. Pérez. "Seguridad de la información, un conocimiento imprescindible". https://www.obsbusiness.school/blog/seguridad-de-la-informacion-un-conocimiento-imprescindible. Accedido el 02-05-2021.

[106] A. I. González-Tablas Ferreres, J. M. García-Romero de Tejada, L. González Manzano y P. Martín González. "Introducción a la criptografía". http://ocw.uc3m.es/ingenieria-informatica/criptografia-y-seguridad-informatica/material-propio%20LorenaChema/L2_Intro_criptosistemas.pdf. Accedido el 02-05-2021.

[107] A. I. González-Tablas Ferreres, J. M. García-Romero de Tejada, L. González Manzano y P. Martín González. "Criptosistemas Simétricos: Bloque". http://ocw.uc3m.es/ingenieria-informatica/criptografia-y-seguridad-informatica/material-propio-1/L5_Criptosistemas_simetricos.pdf-1. Accedido el 02-05-2021.

[108] A. I. González-Tablas Ferreres, J. M. García-Romero de Tejada, L. González Manzano y P. Martín González. "Esquemas de firma digital". http://ocw.uc3m.es/ingenieria-informatica/criptografia-y-seguridad-informatica/material-propio-2/L11_Esquemas_firma_digital.pdf. Accedido el 02-05-2021.

[109] IBM. "Infraestructura de claves públicas (PKI)". https://www.ibm.com/docs/es/ibm-mq/7.5?topic=ssfksj-7-5-0-com-ibm-mq-sec-doc-q009900--htm. Accedido el 02-05-2021.

[110] Fábrica Nacional de Moneda y Timbre. "1028 - ¿Qué es una Autoridad de Registro?" https://www.sede.fnmt.gob.es/preguntas-frecuentes/otras-preguntas/-/asset_publisher/1RphW9IeUoAH/content/1028-que-es-una-autoridad-de-registro-?inheritRedirect=false. Accedido el 07-05-2021.

[111] A. I. González-Tablas Ferreres, J. M. García-Romero de Tejada, L. González Manzano y P. Martín González. "Infraestructuras de clave pública". http://ocw.uc3m.es/ingenieria-informatica/criptografia-y-seguridad-informatica/material-propio-2/L12_Infraestructuras_Clave_Publica.pdf. Accedido el 03-05-2021.

[112] IBM. "Certificados digitales". https://www.ibm.com/docs/es/ibm-mq/7.5?topic=ssfksj-7-5-0-com-ibm-mq-sec-doc-q009820--htm. Accedido el 03-05-2021.

[113] Equipo de soporte de SSL. "¿Qué es un certificado X.509?" https://www.ssl.com/es/preguntas-frecuentes/%C2%BFQu%C3%A9-es-un-certificado-x-509%3F/. Accedido el 03-05-2021.

[114] Packtpub. "PKI certificate standards for IIoT". https://subscription.packtpub.com/book/business/9781788832687/3/ch03lvl1sec31/pki-certificate-standards-for-iiot. Accedido el 03-05-2021.