Portarretratos inteligente, acceso a servicios en red desde dispositivos de bajas prestaciones

RESUMEN

El acceso a los diferentes tipos de información que ofrece Internet ha pasado de ser algo habitual a una necesidad continua, gracias al auge de las redes sociales. Esto genera una demanda de nuevas formas de acceder a la información, lo que representa una oportunidad para desarrollar nuevas soluciones.

El portarretratos inteligente, o *Smart Photo Frame*, es un dispositivo que muestra de forma continua información proveniente de diferentes servicios en red. Su formato es similar al de un marco de fotos digital, con una mayor interactividad.

Nosotros planteamos una plataforma para portarretratos inteligentes, que:

- Permite el acceso a Internet sin restricciones desde cualquier dispositivo
- Es capaz de funcionar desde dispositivos con hardware muy limitado
- Evita la obsolescencia, no dependiendo del software del dispositivo para acceder a nuevos tipos de contenidos

Su arquitectura está basada en la transformación en imágenes fijas de los contenidos accedidos a través de un navegador. Estas imágenes son generadas por un agente externo al dispositivo cliente, cuyo único requisito computacional será representar la información en su pantalla e interactuar con el usuario.

Con el fin de demostrar el funcionamiento y viabilidad de la solución, se ha implementado el proxy completo, junto con un emulador de portarretratos para PC y una implementación de demostración plenamente funcional sobre la videoconsola Nintendo DS.

En este marco de desarrollo, hemos realizado un estudio de viabilidad de diferentes formatos de representación de imágenes, con el fin de encontrar el óptimo para nuestra solución.

Finalmente, hemos comprobado que es una plataforma completa y versátil, que ofrece múltiples oportunidades para el desarrollo de servicios interactivos en red para dispositivos de bajas prestaciones.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1
Introducción

En este capítulo se presentan los objetivos del proyecto, así como el proceso de trabajo y la estructura de la memoria.

1.1. Motivación

La idea de este proyecto, surge de la creciente necesidad de las personas por permanecer conectadas en todo momento. Desde hace un tiempo, consultar el e-mail o la información meteorológica desde los dispositivos móviles se ha convertido en algo muy habitual, por esto, cada vez son más los dispositivos tipo teléfono, que facilitan el acceso a los servicios disponibles en la Web.

La gráfica de la Figura 1.1, muestra los resultados porcentuales de uso de Internet por parte de la población general (eje de ordenadas) obtenidos año a año (eje de abscisas) en el Estudio General de Medios (EGM) realizado por la Asociación para la Investigación de Medios de Comunicación (AIMC). Se observa claramente cómo el uso de Internet en España no ha parado de crecer en los últimos años. A su vez, puede comprobarse cómo cada vez más personas utilizan Internet diariamente.

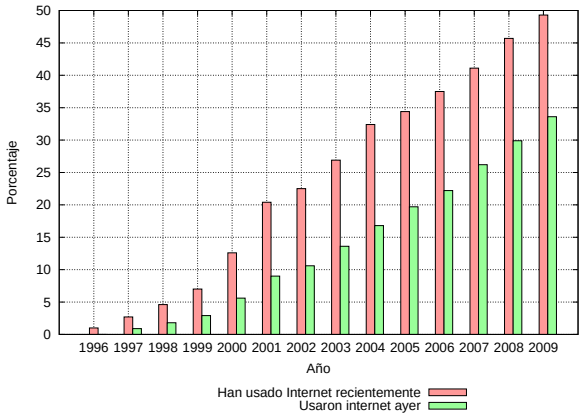


Figura 1.1: Estudio de la AIMC sobre el uso de Internet

.

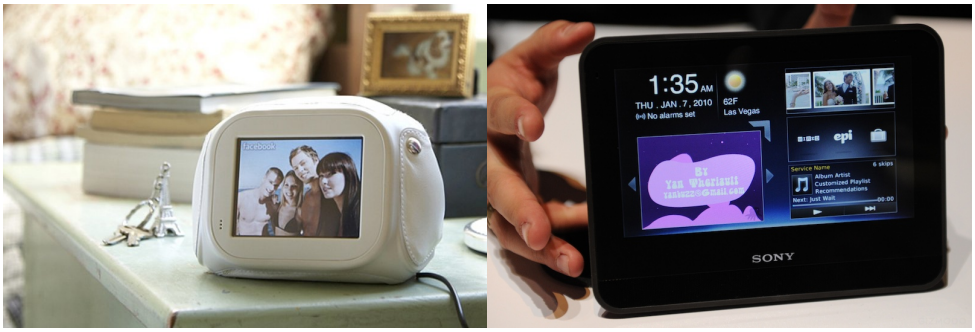
El conjunto de dispositivos que permiten este tipo de acceso es muy heterogéneo. Actualmente están integrados desde en productos tan populares dentro de la electrónica de consumo como son los teléfonos móviles, hasta en productos únicamente destinados a ofrecer este tipo de

contenido. De estos últimos destacan en el mercado dos: Chumby y Sony Dash.

Chumby es un dispositivo creado en 2006. El aspecto externo es similar a un despertador con pantalla táctil. Internamente ejecuta un sistema Linux sobre un procesador ARM, incorporando también conectividad Wi-Fi y USB. Está diseñado para ejecutar widgets (pequeña aplicación gráfica que ofrece información específica o diversas funcionalidades) gracias a una versión de Adobe Flash Player incluida.

Sony Dash es similar a Chumby, siendo compatible con los mismos widgets. Salió al mercado en mayo de 2010 y todavía no cuenta con demasiadas aplicaciones exclusivas. El hecho de que un marca como Sony comercialice un portarretratos inteligente, refleja el gran potencial de este mercado.

La Figura 1.2 muestra la apariencia de estos dos últimos dispositivos al mismo tiempo que da una idea de las dimensiones de cada uno.



(a) Chumby **(b)** Sony Dash

Figura 1.2: *Smart Photo Frames* actualmente en el mercado

Los teléfonos móviles de última generación, también llamados *smartphones*, ofrecen diversos servicios de acceso a contenidos específicos de Internet, tales como redes sociales o noticias. Los *smartphones* más potentes son capaces de ejecutar navegadores web completos. Sin embargo, los teléfonos móviles son dispositivos "de bolsillo", es decir, una restricción de diseño es que su tamaño sea el mínimo posible. Un efecto lateral de esta restricción es el tamaño reducido de sus pantallas, que puede provocar que la navegación no sea agradable para el usuario.

Existen navegadores como el "Opera Mini", desarrollados para dispositivos móviles. Este navegador utiliza la plataforma Java ME, cuya presencia es requisito indispensable en el dispositivo destino. El método de navegación es el siguiente: se solicita la información a través de un servidor de la compañía, el cual procesa las páginas y las reduce, antes de enviarlas de vuelta al dispositivo. Versiones actuales de este sistema permiten una compresión de datos de hasta un 90 %.

Como hemos visto, la variedad de dispositivos hace que el software de los mismos sea variado y muy complejo, ya que requieren de tecnologías como son Java o Flash que pueden no estar disponibles para ellos.

Las tecnologías web cada vez son más dinámicas. Mantener un dispositivo actualizado y preparado para reproducir cualquier tipo de contenido es costoso y puede dar lugar a errores.

Esto, sumado a que podemos disponer de varios dispositivos, puede resultar demasiado costoso para un usuario que deba administrar varios de ellos, reinstalando actualizaciones para cada uno.

Así pues, a la hora de llevar a cabo nuestra solución debemos tener en cuenta los siguientes puntos:

- **Capacidad de cálculo**
  Como hemos visto, para este tipo de dispositivos portátiles es habitual el uso de programas basados en Flash u otras plataformas. Sin embargo, esta opción puede incrementar sensiblemente el coste, además de no ser válida para cualquier tipo de máquina. Si pensamos en dispositivos portátiles como, por ejemplo, la Nintendo DS, su potencia de cálculo es limitada, y por tanto, este tipo de programas no podrían funcionar de manera cómoda. Se requiere restar todo el trabajo posible al cliente con el fin de poder adecuarlo a cualquier máquina o reducir los costes.

- **Variedad de plataformas**
  Existe un conjunto muy amplio de plataformas disponibles actualmente, como Java o Flash. Estas plataformas son muy heterogéneas y no necesariamente interoperables. La elección de plataforma no es trivial, ya que un sistema puede servir para unos dispositivos, pero no ser posible su uso en otros. Esto crea la necesidad de diferentes versiones de un mismo proyecto dependiendo del dispositivo, lo que genera un aumento en los costes de desarrollo y mantenimiento si se pretenden cubrir múltiples plataformas.

- **Variedad de dispositivos**
  Cada vez son más los dispositivos desde los que se quiere acceder a Internet para mostrar información, desde portarretratos hasta pantallas integradas en electrodomésticos. Desarrollar un navegador propio para cada uno de ellos es impensable.

- **Tecnologías usadas por páginas web**
  Estas tecnologías no son fijas, están en continuo desarrollo, por lo tanto se requieren actualizaciones de software para poder adaptarse a los nuevos cambios con cierta frecuencia. Actualizaciones que pueden ser desde el propio navegador hasta una animación en Java de una página en concreto.

  El tener que descargar una nueva versión del programa, además de requerir conocimientos por parte del usuario, afecta a la comodidad, ya que supone tener que acceder a algún sitio, descargarse el software actualizado e instalarlo de nuevo en su dispositivo. Además, las operaciones a bajo nivel sobre dispositivos aumentan la probabilidad de que se produzcan deterioros, como es el caso de las escrituras sobre memorias *flash*, o queden inutilizados de forma accidental. El usuario quiere tener siempre disponible la información independientemente de los cambios en las páginas, y que los cambios se solucionen de manera transparente para él.

  También debemos mencionar, que los navegadores son cada vez más complejos, lo que implica mayores costes de desarrollo, así como una capacidad de cálculo superior para los dispositivos en los que se ejecutan.

1.2. Objetivos del proyecto

Los objetivos planteados al realizar este proyecto son los siguientes:

- El objetivo principal es la creación de un sistema de acceso a Internet desde dispositivos de muy bajo coste y prestaciones, basado en un proxy que adecue los datos al dispositivo.
- El dispositivo no debe precisar de actualizaciones de firmware, dejándole el peso del trabajo al proxy.
- Diseñar y evaluar un protocolo de comunicaciones que permita una latencia baja con un bajo coste computacional.
- Evaluar y seleccionar diferentes formatos de imagen para su uso en las comunicaciones del dispositivo y el cliente.

Con el fin de experimentar y demostrar el funcionamiento de la arquitectura, como parte del proyecto se implementará la solución completa para dos dispositivos de ejemplo, la Nintendo DS y un ordenador portátil emulando un portarretratos inteligente.

1.3. Estructura de la memoria

En el Capítulo 2 se describe la arquitectura general de la solución, profundizando posteriormente en los elementos que la componen y el protocolo de comunicaciones utilizado.

A continuación, el Capítulo 3 recoge el conjunto de dispositivos tenidos en cuenta para llevar a cabo el sistema de demostración, incluyendo una descripción de la Nintendo DS, utilizado finalmente. Además, se incluye una breve referencia de las herramientas utilizadas durante el proyecto.

El Capítulo 4 incluye las tareas realizadas para llevar a término el proyecto, describiendo el diseño y la implementación de los principales elementos de la solución. También se resume el análisis de viabilidad de diferentes formatos gráficos, realizado para optimizar las comunicaciones entre el cliente y el proxy.

El Anexo A recoge la planificación y el tiempo empleado en desarrollar el proyecto.

El Anexo B extiende el estudio de viabilidad de los diferentes formatos gráficos planteados, incluyendo una descripción de sus características principales y modos de operación, así como resultados de las diferentes pruebas realizadas.

El Anexo C documenta las aplicaciones desarrolladas en el proyecto, a modo de manual de usuario.

El Anexo D recoge la especificación del protocolo de comunicaciones utilizado entre el proxy y el cliente, con el fin de facilitar el diseño de nuevos clientes compatibles.

Capítulo 2
Arquitectura

En este capítulo se propone y defiende la arquitectura propuesta para la solución. Esta arquitectura es la desarrollada en el Capítulo 3 y el Capítulo 4.

2.1. Descripción

Nuestra solución pretende permitir el acceso a servicios presentes en Internet desde cualquier dispositivo, con el único requisito de poseer una pantalla y conexión a la red. Posibles clientes de este servicio pueden ser, entre otros: televisiones, electrodomésticos o marcos de fotos digitales. El rango de dispositivos posibles es muy amplio.

Para que nuestra solución sea viable para cualquier dispositivo con esas características, y teniendo en cuenta los problemas descritos en el capítulo anterior, se decide trabajar sobre una solución orientada al envío de capturas de pantalla.

2.1.1. Envío de capturas

Con la intención de evitar la sobrecarga de los dispositivos de bajas prestaciones y, por otro lado, evitar problemas de actualizaciones, se decide que la navegación real del cliente se ejecute fuera de él. Para ello, debe existir un proxy conectado a Internet, que reciba las peticiones del cliente y ejecute sus comandos, renderizando el contenido web y enviándole las capturas del navegador al cliente, dándole la sensación de ejecución en el propio sistema.

2.1.2. Diseño de la solución

Las páginas web están normalmente diseñadas para ser visualizadas en resoluciones convencionales de pantalla de ordenador como puede ser 1024x768. Sin embargo, estas resoluciones no coinciden con las habituales de los dispositivos hacia los que nos orientamos. Es necesario realizar una adecuación de la navegación, que en nuestro caso realizará un proxy diseñado para este proyecto, explicado en la Sección 2.2.2.

La arquitectura de la solución es la representada en la Figura 2.1. En la parte izquierda de la figura se encuentran ejemplos de los diferentes tipos de cliente: un marco digital, un televisor, etc. Vemos como éstos se comunican con nuestro proxy que hace de puente entre el cliente e Internet. El proxy es el encargado de interactuar con el usuario gestionando sus peticiones y

sirviéndole la información. Para ello, el proxy renderiza la información que el usuario solicita y se la envía en forma de capturas. Dichas peticiones se realizan habitualmente a diferentes servidores de Internet. Además, interpreta los comandos que le llegan del cliente en su navegador local.

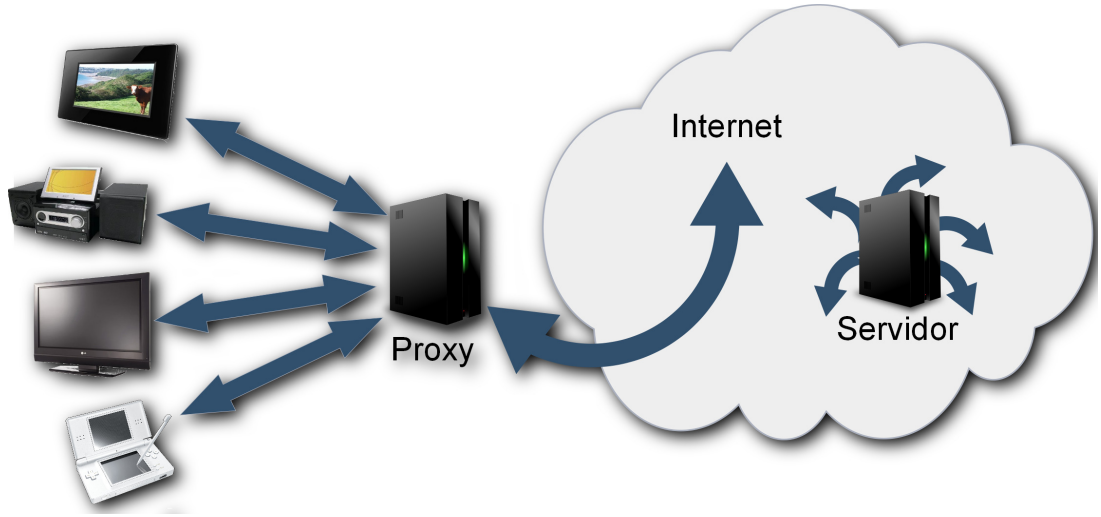


Figura 2.1: Arquitectura de la solución

2.1.3. Valoración de la solución

A priori, se plantea que la solución tiene las siguientes ventajas:

- **Independencia del cliente:** Como ya se ha visto, en nuestro esquema de comunicación el cliente está exento de todo el trabajo de adecuación de contenidos. Por lo tanto, las nuevas actualizaciones o cambios que se puedan producir en las páginas web no le afectarán de manera alguna, siendo transparente para él la representación de los datos. Tan sólo un visor de imágenes será suficiente para disponer de todo tipo de contenidos.
- **Bajo coste en el dispositivo cliente:** No se precisa de un hardware específico sino que cualquier dispositivo con conexión a Internet, pantalla y algún método de interactuar con ella puede ser parte de esta solución. Gracias a esto, cualquier persona que posea un dispositivo con estas características, desde una Nintendo DS hasta una televisión, podrá reutilizarlo para este fin, no siendo necesario que adquiera un aparato más moderno.

También podemos distinguir el siguiente conjunto de inconvenientes:

- **Generación de capturas:** El inconveniente principal de esta solución es la relación entre la calidad de la imagen que se envía y el tiempo en enviarla. Para dar la sensación de interactividad la latencia debe ser mínima, sin descuidar el ancho de banda, por lo que es necesario un compromiso entre calidad, tamaño y tiempo de transferencia.
- **Complejidad:** El conjunto de operaciones que hay que realizar para que el cliente vea las páginas web es más complejo que en una navegación normal, ya que se necesita una capa intermedia de transformación de páginas a imágenes. Esto genera un retraso notable que se observa especialmente cuando son necesarias tasas de refresco altas, como es el caso de la reproducción de vídeo.

Tal y como se detalla en el Anexo B, se han realizado numerosos experimentos de variación del formato y calidad de la imagen, ya que es un factor determinante a la hora de conseguir una navegación fluida.

2.2. Elementos

2.2.1. Cliente

El cliente solicita los contenidos que desea ver al proxy. Éste le envía la captura correspondiente para que el cliente interactúe con ella como si se tratara de su propia página web. También envía eventos derivados de la navegación al proxy, como pueden ser un clic de ratón o una introducción de texto. Estos comandos son interpretados por el proxy en el navegador local y los efectos derivados de estos eventos son enviados en forma de nuevas capturas en las peticiones siguientes.

Las opciones aparentemente limitadas del cliente ofrecen una gran ventaja: sólo ha de instalarse la solución una vez. El peso del trabajo es llevado a cabo por el proxy pese a que la navegación parezca que se esté llevando a cabo localmente. En todo momento esto se realiza de manera transparente y con independencia de la versión del proxy.

2.2.2. Proxy

El proxy puede dar servicio a varios clientes, proporcionándoles la información que soliciten. Además, gestiona las peticiones que se le envían, solicitando la información a los servidores y ajustándola de acuerdo con sus necesidades.

El proxy puede estar accesible desde Internet o desde una red local. El acceso desde red local es preferible, ya que maximiza la transferencia de imágenes con el dispositivo cliente. Se espera que el volumen de tráfico entre el proxy e Internet sea menor que entre el proxy y el cliente, por lo que este esquema permite mantener una baja latencia y un ancho de banda adecuado a cada elemento.

Este mecanismo permite que el cliente sea completamente independiente de las tecnologías necesarias para visualizar contenidos web. Todo el procesamiento necesario para poder interpretar una página web, lo realizará el proxy, que deberá incorporar versiones actualizadas de navegadores y *plug-ins*.

El proxy realiza diferentes funciones para lograr sus objetivos, que se detallan en los apartados siguientes.

Envío de información

La comunicación entre el proxy y el cliente se realiza utilizando UDP[1]. Por lo tanto, será válida cualquier tecnología de comunicaciones que soporte UDP con un ancho de banda suficiente.

[1] User Datagram Protocol

Ejemplos de tecnologías soportadas son: Wi-Fi, 3G, Ethernet o PLC.

Los tipos de información que se envían entre estos dos elementos son:

- **Identificación de cliente. Se envía en sentido de cliente a proxy**
  El cliente, al iniciar la sesión, proporciona sus datos para que el proxy pueda servirle los contenidos que él anteriormente ha solicitado. De esta manera, el cliente no tiene que decir qué contenidos desea ver al iniciar la sesión, gracias a que sus preferencias están guardadas.
- **Capturas de pantalla. Se envía en sentido de proxy a cliente**
  El proxy provee al cliente de las capturas conforme navega por los contenidos solicitados.
- **Actividad de usuario. Se envía en sentido de cliente a proxy**
  El usuario envía al proxy su interacción en el transcurso de la navegación para que éste interprete los comandos en su navegador local y realice las peticiones externas que sean necesarias.

El tipo de envío que genera más cantidad de información en tránsito son las capturas de pantalla. Las capturas se fragmentan en segmentos, que son recogidos en el destino y unidos para completar la imagen, tal y como se detalla en el protocolo. Dicho protocolo de envío se describe exhaustivamente en la Sección 2.4.

Interpretación de eventos

El proxy recibe los eventos generados en la navegación por parte del cliente, tales como clics de ratón o movimientos dentro de la pantalla. Él es el encargado de identificarlos y ejecutarlos, enviándole el resultado de estos eventos al cliente a modo de capturas.

Renderizado

El envío de las capturas al cliente, supone un renderizado anterior. Este trabajo es llevado a cabo por el proxy.

El proxy puede renderizar desde dos fuentes diferentes según sea necesario:

- **Control remoto del ordenador:** Si el usuario está utilizando la opción de control remoto del escritorio, la captura la toma del escritorio de la máquina en la que esté instalado el propio proxy.
- **Navegación web:** Si la captura es para una navegación normal como puede ser mirar el correo web, se captura la imagen de un navegador local del proxy en el cual se simula la navegación del cliente.

Una vez renderizada la imagen, se ajusta según el tipo de imagen que se desee enviar. Un ejemplo de este ajuste se encuentra en la Figura 2.2. Se puede requerir enviar una página entera tal cual está en el navegador o el escritorio completo, para lo que es necesario escalarla al tamaño de la pantalla del dispositivo cliente. También puede enviarse una sección del total de la captura a tamaño real, para ser visualizada en el cliente.

Figura 2.2: Ejemplos de captura

.

Una vez que se ajusta la imagen según sea necesario, se envía la captura al dispositivo cliente.

El tamaño de la captura que es enviada depende del dispositivo, ya que se ajusta a la resolución que pueda tener el cliente, y el formato depende del usuario, que puede elegir entre JPEG y GIF según sus necesidades, como se explica en el Anexo B.

2.3. Conexión proxy-cliente

La conexión entre el proxy y el cliente tiene las siguientes funcionalidades y características:

- **Identificación de usuario:** Para ofrecer funcionalidades independientes a los clientes, se utiliza un sistema sencillo de cuentas. Una base de datos en el servidor almacena los clientes y sus preferencias, que son accedidas cuando entran en el sistema. Gracias a esto, un cliente puede tener una configuración de servicios personalizada.

- **Cifrado:** El cifrado en la comunicación proxy-cliente es fundamental cuando se realiza a través de medios no seguros.

- **Desconexión:** Cuando se supera un tiempo preestablecido sin recibir tramas por parte del cliente, el servidor cierra la conexión.

2.4. Protocolo de comunicaciones proxy-cliente

Uno de los principales objetivos en el diseño de este protocolo es reducir la latencia en la comunicación con el cliente, de forma que la navegación resulte lo más fluida posible. Como sabemos, el usuario tiene la ilusión de estar llevando a cabo una navegación local y espera de manera inmediata los cambios en las páginas que está visitando.

2.4.1. Funcionamiento

El protocolo, cuya representación gráfica puede consultarse en la Figura D.1, funciona de la siguiente manera para los envíos de las capturas:

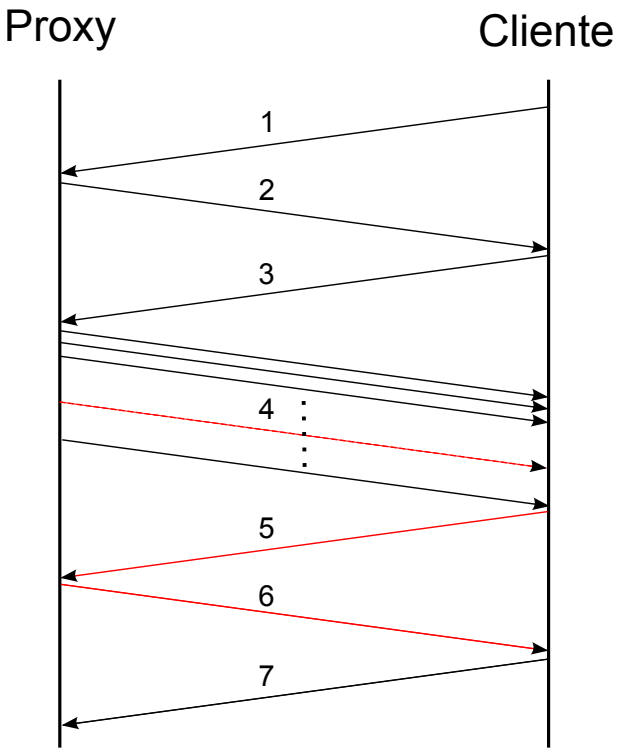


Figura 2.3: Protocolo proxy-cliente

.

1. El cliente solicita una nueva imagen. Para esto le envía una trama.

2. El proxy le envía una trama informándole de que ha recibido la petición y le especifica el número de bloques en los que se divide la imagen que le enviará.

3. El cliente guarda sitio para recibir la imagen completa y la inicializa. Además, envía una trama al proxy para informarle de que ya puede empezar a enviar los bloques de la imagen.

4. El proxy, envía las diferentes partes que componen la imagen seguidos mediante tramas. Cada una de ellas tiene el número de secuencia para poder reconstruirla en recepción.

5. Si hay algún bloque que no se ha recibido, o llega con error, se envía inmediatamente un NAK con el número de trama que le falta al proxy.

6. El proxy reenviaría una trama con el bloque perdido o dañado.

7. Cuando recibe todas las tramas, se envía un ACK para informar al proxy de que todos los bloques de la imagen han sido recibidos.

En el Anexo D se detalla el formato de las tramas enviadas para todos los tipos de mensajes presentes en la comunicación proxy-cliente.

2.4.2. Ventajas respecto a otros protocolos

El diseño de este protocolo está influido por la necesidad de almacenar completamente la imagen en el cliente antes de ser mostrada. No tenemos restricciones en cuanto a tamaños de ventanas de recepción, ya que siempre podremos albergar, todos los segmentos de imagen que nos lleguen. El hecho de que una trama llegue dañada, o no llegue, no es un impedimento para seguir recibiendo tramas, lo que se traduce en una comunicación más rápida que la de otros protocolos como los de ventana deslizante con límite de tamaño o *Stop & Wait*.

Capítulo 3
Planteamiento y entorno de trabajo

En este capítulo se describe el planteamiento inicial para la selección del dispositivo a utilizar, y las posibles alternativas. Además, se incluye una relación de las herramientas utilizadas para el desarrollo del proyecto

3.1. Planteamiento

Para implementar la aplicación se barajaron varias posibilidades con el fin de determinar qué dispositivos cliente utilizar. El cliente ideal sería un marco de fotos digital con acceso a Internet. Éstos tienen una pantalla bastante grande en comparación con móviles u otros dispositivos de tamaño reducido.

Debemos tener en cuenta, que puede haber dispositivos de múltiples tamaños, desde un despertador con una pantalla muy pequeña, hasta una televisión.

3.2. Dispositivos utilizados

Hemos trabajado sobre varios dispositivos para dar forma a nuestra solución. Por un lado, se ha hecho uso de un ordenador de sobremesa para albergar en él el proxy que da servicio a los diferentes dispositivos cliente. Los requisitos mínimos para que un ordenador de sobremesa pueda ejecutar el proxy no son elevados, ya que cualquier ordenador moderno con Microsoft Windows, acceso a Internet y un navegador puede hacerlo. Por supuesto, un ordenador con una potencia elevada será capaz de reducir los tiempos de compresión y dar como resultado una navegación más fluida.

Como dispositivos cliente, de entre todos los que pueden formar parte de la solución, se han elegido dos de ellos para demostrar el funcionamiento de la aplicación y llevar a cabo las medidas de rendimiento: la consola Nintendo DS y un ordenador portátil a modo de emulador de un portarretratos inteligente.

La Nintendo DS, como se explica en la Sección 3.4, se utiliza por ser un dispositivo económico y por el atractivo de sus dos pantallas para mostrar el tipo de información al que va dirigida nuestra solución.

El uso del ordenador portátil, que actúa como emulador de portarretratos, se ha implementado y además ha servido de referencia para medidas de capturas para resoluciones mayores de

pantalla. Cualquier ordenador portátil convencional puede disponer de un navegador completo y, por lo tanto, nuestra solución no tiene sentido para este tipo de dispositivo. El programa creado tan sólo hace uso de su pantalla, ratón, teclado y conexión a Internet, obviando el resto, con el fin de comprobar cómo responde el envío de capturas de pantalla cuando los tamaños son mayores.

3.3. Tecnología de comunicaciones

La conexión a una LAN o a Internet es fundamental en nuestra solución, ya que todo el contenido que visualiza el cliente va a llegarle a través de esta vía.

Por conveniencia y disponibilidad, se ha utilizado Wi-Fi de forma exclusiva durante la realización de este proyecto. Por supuesto, no deben descartarse otras opciones, como 3G, PLC o Ethernet.

La tecnología Wi-Fi proporciona una gran libertad, ya que gracias a ella y sin necesidad de cables, los dispositivos pueden acceder a Internet desde cualquier lugar habilitado.

3.4. Nintendo DS

La Nintendo DS es una videoconsola portátil desarrollada por Nintendo, que se comenzó a comercializar en el año 2004. Su característica principal es la presencia de dos pantallas independientes, una de ellas táctil. A lo largo de los años, su diseño se ha ido refinando y adaptando a diferentes necesidades y mercados, dando lugar a los siguientes modelos:

- Nintendo DS (2004): primera versión comercializada.
- Nintendo DS Lite (2006): versión reducida con misma funcionalidad que la original.
- Nintendo DSi (2008): iteración de la DS Lite que incluye cámara de vídeo y permite el uso de diferentes servicios en Internet.
- Nintendo DSi XL (2009): una versión de mayor tamaño de la Nintendo DSi.

En este proyecto se ha desarrollado sobre la Nintendo DS Lite, que comparte el hardware básico de toda la familia DS. La diferenciación principal respecto a la Nintendo DS original es su menor volumen, a pesar de que aumenta el tamaño de las pantallas.

La videoconsola cuenta con dos procesadores: un ARM9, a una frecuencia de 66MHz, y un ARM7, a 33MHz. En general, los programas de usuario deben correr siempre sobre el ARM9, utilizando el ARM7 únicamente desde llamadas de biblioteca.

La gestión de memoria es compleja. Incorpora una caché de 4KB para datos y 8KB para instrucciones, junto a 32 KB de TCM (memoria rápida controlada por software) para instrucciones y 16 KB de TCM para datos.

Dispone de dos pantallas LCD con una resolución de 256x192 píxeles, una de ellas táctil. El motor de aceleración de gráficos en dos dimensiones está disponible para las dos pantallas

Figura 3.1: Nintendo DS Lite

simultáneamente, mientras que el motor de tres dimensiones sólo puede utilizarse de forma efectiva sobre una.

Además de las pantallas LCD, la DS soporta dieciséis canales de sonido, incorporando micrófono, dos altavoces con control de volumen analógico y toma de auriculares. Así mismo, dispone de varios botones y cursores, que junto a la pantalla táctil hacen de la DS una videoconsola con un gran potencial para todo tipo de aplicaciones interactivas.

La Nintendo DS utiliza Wi-Fi como medio de comunicación para juegos en red. La versatilidad que ofrece la tecnología Wi-Fi ya ha sido aprovechada por la propia Nintendo a partir de la DSi, ampliando los servicios disponibles para la consola. Utiliza el estándar IEEE802.11b en la banda de 2.4GHz, dando soporte a encriptación WEP.

Existe un navegador para la Nintendo DS y DS Lite creado por Opera, basado en su motor Presto. Este navegador presenta serias limitaciones, como no ser capaz de reproducir contenido flash o ser demasiado lento, a pesar de incluir una ampliación de memoria para la consola. La Nintendo DSi incorpora una versión más moderna de este navegador.

La Nintendo DS es una videoconsola extremadamente popular, con más de ciento veinte millones de unidades vendidas en todo el mundo. El kit de desarrollo oficial no es de libre distribución, por lo que existen varias comunidades de *homebrew* (desarrollo de aplicaciones no oficiales para videoconsolas) que ofrecen bibliotecas y utilidades para crear software para la DS. Este software debe ser ejecutado utilizando un dispositivo especial, que debe adquirirse de forma separada.

Su gran interactividad y baja potencia la hacen la candidata ideal como sistema de desarrollo y demostración para este proyecto. Sus dos pantallas ofrecen nuevas oportunidades de mostrar el contenido.

3.5. Herramientas utilizadas

Se han utilizado varias herramientas para implementar la solución. A continuación, se detallan las principales.

3.5.1. Borland Delphi 7

Borland Delphi es un entorno de desarrollo de aplicaciones para Microsoft Windows, comercializado por Borland en el año 2002. El lenguaje utilizado es un derivado de Pascal, con varios añadidos que incluyen orientación a objetos.

En este proyecto se ha utilizado Delphi para la programación del proxy, de forma que se ha podido reutilizar código ya existente de capturas de pantalla con la Nintendo DS. También se ha utilizado para implementar el emulador de portarretratos en PC.

La amplia biblioteca de componentes y su integración con la API de Windows, junto a las potentes características de creación de menús, han sido características muy ventajosas a la hora de realizar este desarrollo. El principal inconveniente es la dificultad a la hora de portar el software generado a otras plataformas.

3.5.2. devkitARM release 21

Para el desarrollo de software sobre la Nintendo DS se ha utilizado la plataforma devkitARM. El entorno es gratuito y está bajo licencia GPL[1], por lo que todo el código fuente está disponible.

devkitARM forma parte del proyecto devkitPro, que ofrece diferentes entornos de compilación cruzada para desarrollar software para diversas videoconsolas. Concretamente, devkitARM permite desarrollar programas para procesadores ARM, e incluye soporte específico para GameBoy Advance y Nintendo DS.

El entorno para usuario final de devkitARM incluye un *toolchain* completo de compilación cruzada para ARM. Este *toolchain* incorpora los compiladores de C y C++ de GCC 4.1.2, además de utilidades de generación de binarios e imágenes para la Nintendo DS.

Dentro del mismo entorno se encuentra también "libnds", una biblioteca de bajo nivel que permite controlar el hardware de la Nintendo DS. Por desgracia, al ser una implementación no soportada por Nintendo, no cubre todas las características de la máquina.

La programación para Nintendo DS resulta compleja por las limitaciones de su arquitectura y la falta de soporte para las bibliotecas con las que contamos. Como problema adicional, no hemos podido utilizar ninguna herramienta de depuración: aunque existen varios emuladores de la arquitectura, ninguno simula correctamente la comunicación Wi-Fi, necesaria en este proyecto.

[1] General Public License

3.5.3. Apache

La navegación desde algunos de los dispositivos que tratamos está muy limitada en comparación con la navegación desde ordenadores convencionales. Las bajas resoluciones de algunos de estos dispositivos hacen que sea conveniente adaptar las páginas web antes de presentárselas al usuario.

En algunos casos (Facebook o Wikipedia, por ejemplo), la propia página tiene versiones adaptadas a dispositivos móviles que podemos usar en nuestra aplicación. Para otro tipo de páginas, como el propio Facebook o Flickr entre otras, existen APIs públicas que permiten crear páginas web con la misma información que ofrece la página oficial.

Por eso, dentro de la arquitectura de la aplicación se incluye un servidor web, desde el que el usuario pueda comenzar su navegación con un menú personalizado, y visitar páginas adaptadas si así lo desea. Esta labor la realiza el servidor web Apache.

El servidor HTTP de Apache es uno de los servidores web más utilizados en Internet. Además de ser un software maduro y estable, es libre. Uno de sus puntos fuertes para su elección en este proyecto, es su integración inmediata con gran número de lenguajes de programación de páginas web dinámicas.

Capítulo 4
Trabajo realizado

Este capítulo pretende ofrecer una visión general del funcionamiento y las características del software implementado para este proyecto. Se incluye un resumen del trabajo de análisis de formatos gráficos realizado para optimizar la transferencia de imágenes.

4.1. Desarrollo de software

La implementación de la solución implica la creación de software para cada uno de los elementos presentes en la arquitectura. En primer lugar se necesita la creación de una aplicación que, instalada en un ordenador, actúe de proxy. Los dispositivos cliente que deseen acceder a la solución, se conectan a través de dicho proxy, que será el encargado de suministrarles la información.

Así mismo es necesario generar el software que debe residir en los dispositivos cliente. En nuestro caso ha sido necesario realizar dos implementaciones separadas, ya que el desarrollo de software para la Nintendo DS es muy diferente al desarrollo para un portátil. Por supuesto, la dificultad de la implementación del software es muy inferior a la de la creación de un navegador completo.

En los siguientes apartados se detalla la función de los diferentes elementos de la solución.

4.1.1. Software de cliente

Como ya hemos visto, se hace uso de dos dispositivos cliente diferentes: un ordenador portátil y una consola Nintendo DS. Ambos utilizan una aplicación similar que corresponde a la que se detalla a continuación.

La aplicación que reside en la consola Nintendo DS, ha sido implementada en C++ utilizando devkitARM, descrito en la Sección 3.5.2.

Para la aplicación cliente del ordenador portátil se ha elegido Delphi, de forma que se ha podido reutilizar parte del código del proxy.

Como se ha expuesto en la Sección 1.2, se ha procurado que la aplicación que implementa el dispositivo cliente soporte el mínimo trabajo posible del conjunto de la solución. Gracias a esto, la aplicación para este dispositivo no precisa de actualizaciones. De esta manera se consigue que

el cliente tenga una aplicación que no deba ser modificada una vez instalada. Cualquier cambio que afecte a las tecnologías de los servicios que ofrece nuestra solución no afectan en absoluto al cliente.

Las funciones que realiza el software en el dispositivo cliente son:

- **Conexión a la red**: Al inicio de la aplicación, ésta debe conectarse a una red. El usuario ha de seleccionar su red e introducir la contraseña para identificarse, si es que la hubiera.
- **Identificación**: Una vez aceptada la red a la que conectarse, el dispositivo cliente, de manera transparente al usuario, envía al proxy una identificación, única para el dispositivo, para que éste le muestre los contenidos que el usuario previamente ha solicitado.
- **Cifrado de la comunicación:** El dispositivo cifra toda su comunicación con el proxy utilizando una contraseña compartida.
- **Recepción de capturas y presentación**: Una vez que el cliente se ha identificado, esta aplicación comenzará a recibir la información en forma de capturas. Deberá recibirlas y mostrarlas por pantalla conforme sean recibidas.
- **Envío de peticiones**: A consecuencia de la interacción del usuario con estas capturas, se generan peticiones como puede ser un clic de ratón o un movimiento dentro de la pantalla. Estos eventos son enviados al proxy para que los interprete y devuelva el resultado en forma de una nueva captura.

En la figura 4.1 se presenta una imagen de la ejecución de la aplicación cliente sobre la Nintendo DS.

Figura 4.1: Ejecución del software de cliente en NDS

4.1.2. Software de proxy

Como se ha descrito en los objetivos del proyecto, la aplicación que implementa el proxy es la que lleva a cabo la mayor parte del trabajo, y por tanto, la que resulta más interesante conocer. Se ha implementado en Delphi por las razones descritas en la Sección 3.5.1.

A continuación se presentan las funciones llevadas a cabo por el proxy instalado en un PC:

- **Identificación**: El proxy es el encargado de mediar entre el cliente y el servidor en la identificación inicial del cliente.
- **Envío de capturas**: La información como y hemos visto, se envía a modo de capturas. El proxy simula la navegación del cliente en un navegador local y renderiza el contenido para enviarlo al cliente.
- **Cifrado de la comunicación:** El proxy cifra toda su comunicación con el dispositivo utilizando una contraseña compartida.
- **Interpretación de eventos**: Además de solicitud de capturas, el proxy recibe eventos generados en la navegación por parte del usuario como clics de ratón en enlaces o introducción de texto. El proxy ha de interpretar estos eventos en su propio navegador local y enviar sus resultados de vuelta al cliente a modo de capturas.
- **Suministro de información**: El proxy es el encargado de suministrar la información al cliente obteniéndola de la fuente que sea necesaria. Bien de Internet o bien de un servidor en donde se encuentra contenido específico para la aplicación. Una vez recogida la información, la visualiza en su navegador local de donde la renderizará para ofrecérsela al cliente.

En la Figura 4.2 se observa una captura de la aplicación proxy.

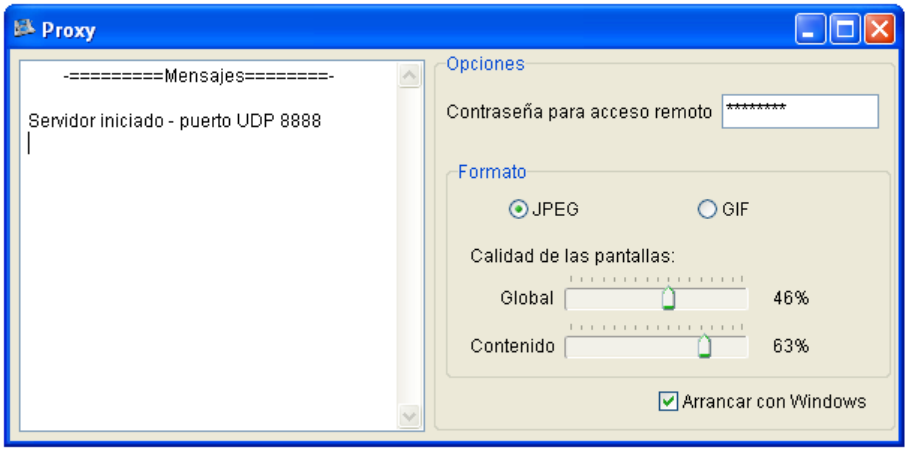


Figura 4.2: Aplicación proxy

4.1.3. Servidor

Aunque se espera que el cliente solicite páginas web remotas, se ha implementado un servidor específico de la solución. Su función principal es la de proporcionar los servicios que se requieran que no puedan ser recogidos directamente de Internet. Es decir, adapta los contenidos de Internet

antes de proporcionárselos al proxy.

El servidor lleva a cabo las siguientes funciones:

- **Identificación del usuario**: En el servidor se encuentra una base de datos con la información de cada usuario. Cuando un usuario accede al sistema, se identifica. Esta clave es la que llega al servidor, que busca en su base de datos qué información solicita este usuario y se la envía al proxy para que la prepare.
- **Adaptación la información**: Al ser posible utilizar como dispositivo cliente máquinas con resoluciones de pantalla reducidas, las resoluciones para las que están destinadas las páginas web normales resultan demasiado grandes para poder ser visualizadas cómodamente. El servidor ofrece, para algunas páginas web, versiones adaptadas a diferentes resoluciones. Esto se consigue creando páginas a partir de APIs públicas de algunos servicios o por conversión directa del contenido solicitado. Según el servicio solicitado utilizará diferentes métodos, como se explica en el Anexo C.

4.1.4. Protocolo de comunicación

El protocolo de comunicación empleado para el envío de información, especialmente para el envío de capturas, es uno de los puntos críticos del proyecto. La rapidez en el envío de la información, es uno de los pilares para el buen funcionamiento de la solución.

En líneas generales, cuando una captura es solicitada, el proxy es el encargado de suministrarla. Para ello, divide la captura en segmentos de tamaño fijo y los envía en secuencia al cliente. Por su parte, el cliente recibe estos segmentos y atendiendo a su número, contenido dentro de la trama, reconstruye la imagen y la muestra por pantalla. En caso de faltar algún segmento, como el cliente conoce su identificador, se lo solicitara al proxy, que se lo reenviará.

Es un protocolo de ventana sencillo pero muy conveniente para nuestra solución. En lugar de comprobar cada segmento tras su recepción, se solicitan al final los que faltan, ya que, por lo general, no se producen pérdidas.

En la Sección 2.4 se muestra una descripción detallada del protocolo. En el AnexoD se presenta una especificación completa del mismo.

4.2. Capturas

Una parte importante de nuestra solución son las capturas de pantalla que realiza el proxy y le envía al cliente. Toda la información que recibe el cliente, le llega de esta forma. Por tanto, es necesario estudiarlas en profundidad para elegir las características óptimas que generen los mejores resultados para el conjunto del proyecto.

4.2.1. Obtención

Utilizamos dos tipos de capturas en nuestras aplicaciones de demostración. Como se ha visto en la Sección 2.2.2, las capturas pueden obtenerse de toda la pantalla del proxy, si se accede en modo de escritorio remoto, o de una zona controlada del navegador local utilizado para acceder

a Internet, si se accede en modo de navegación web.

Para el renderizado de la navegación web, el navegador local del cual se obtiene la imagen utiliza de fondo Internet Explorer. Se ha elegido este navegador ya que presenta las mejores compatibilidades para las diferentes páginas web. De esta manera, cualquier actualización de contenidos web se lleva a cabo en Internet Explorer sin afectar al proxy.

4.2.2. Formatos

Se han estudiado tres de los formatos más populares de codificación de imágenes: PNG, GIF y JPEG. Cada uno de estos formatos tiene diferentes parámetros configurables, por lo que se ha realizado un amplio conjunto de medidas estudiando el impacto de diferentes ajustes. Un estudio detallado de estos formatos se puede encontrar en el Anexo B.

4.2.3. Tipos de captura

El tamaño de la captura a enviar por parte del proxy, influye directamente en el tiempo de envío de la captura. Cuanto mayor sea su tamaño, mayor será el tiempo empleado en realizar un envío completo.

Como es lógico, se pretende aprovechar al máximo las pantallas de los dispositivos cliente que se conecten a la solución, luego las capturas enviadas tendrán el tamaño correspondiente a la pantalla del dispositivo que solicite la captura.

Sin embargo, no siempre es necesario enviar toda la captura. Podemos valernos de técnicas para reducir la cantidad de información a enviar dependiendo del escenario en el que nos encontremos. A continuación se detallan algunas de estas técnicas:

- **Envío de pantalla completa**
  No se trata de una opción óptima, ya que supone enviar la captura completa. Este método es necesario durante un inicio de sesión o un cambio de servicio, ya que la captura ha de ser renderizada en una página distinta y la imagen cambia completamente respecto a la anterior, si es que la había.

- **Desplazamiento**
  Por lo general, al navegar por una pagina web, los movimientos más comunes de desplazamiento son los realizados en sentido vertical. Si imaginamos el estado de la pantalla antes y después del desplazamiento, observamos como hay parte de la imagen que se mantiene sin variaciones respecto a la captura anterior, aunque desplazada hacia arriba o hacia abajo según el sentido del movimiento, mientras que la parte nueva ocupa el resto de la pantalla. Lo mismo ocurre con el desplazamiento horizontal. Esta nueva sección de imagen puede ser de desde una sola línea de píxeles hasta casi toda la imagen.

  Para aprovechar esta circunstancia, se decide enviar únicamente la sección de imagen que ha cambiado, desplazando la captura anterior y sobrescribiendo la nueva sección donde corresponda.

Gracias a esta optimización, se puede reducir considerablemente la cantidad de información a transmitir por la red a un cliente que realice este tipo de desplazamientos.

Esta optimización se implementó sobre nuestra solución, por lo que pudimos comprobar que los tamaños de las imágenes a enviar eran menores, ya que tan sólo suponen una porción del tamaño total de la pantalla. El tiempo empleado en renderizar la imagen era el mismo, ya que se necesita capturar toda la pantalla para comprobar si ha habido variación en la pantalla antes de enviar tan solo una sección. Esto es así ya que es posible que además del desplazamiento, se haya producido un cambio en la parte fija y deba ser enviada de nuevo toda la imagen. Por otro lado, este proceso conlleva más trabajo por parte del proxy, ya que debe mantener índices de las posiciones y tamaños enviados anteriormente para calcular el punto a partir del cual enviar la imagen, así como calcular el tamaño de la nueva sección para enviarle todos estos datos al cliente. Esto es necesario para que el cliente sepa dónde colocar la sección de imagen que le llega.

- **Cambios localizados**
  Además de los desplazamientos en la pantalla, podemos encontrarnos con variaciones localizadas dentro de la misma. Imaginemos por ejemplo un anuncio en una página web que se trata de contenido en flash que está en continuo movimiento. Enviar toda la imagen puede ser excesivo, ya que sólo ha cambiado una sección.

  Una posible optimización para beneficiarnos de esta circunstancia es la de enviar solamente los píxeles que hayan sido modificados de la imagen. Esta optimización, es similar a la anterior, ya que sólo enviamos parte de la imagen.

  La diferencia se encuentra en que en el caso anterior, el movimiento venía dado por un evento del cliente, mientras que en este caso, es el proxy el que debe identificar que sólo ha variado una parte antes de enviarla. Esto puede localizarse mediante diferentes técnicas como pueden ser:

  - **Resta de capturas.**
    Éstas deberán estar guardadas en formato BMP para poder comparar píxel a píxel, ya que por ejemplo el formato JPEG no codifica las imágenes en mapa de bits. Si los cambios se localizan dentro de un área determinada de la imagen, tan sólo se enviará ese área.

  - **Análisis de metadatos.**
    Se puede analizar el código en HTML en busca de contenido flash o susceptible de movimiento.

  Esta optimización también ha sido implementada y estudiada con el fin de analizar la posible mejora. Es una optimización más compleja, ya que supone trabajo por parte del proxy, que ha de realizar una comparativa de los BMPs y cada uno de sus píxeles. El tiempo invertido en esta comparación es elevado y aumenta considerablemente el tiempo de preparación de la captura. Una vez seleccionados los píxeles se envía de forma absoluta la menor área de la imagen que contenga todos los píxeles modificados. Para pantallas pequeñas no se obtienen beneficios. Por este motivo, a pesar de haber sido implementado, se dejó sin aplicar en la versión de demostración.

- **Incremental**
  Por último, puede darse el caso de que se produzcan pequeños cambios en píxeles dentro de la página que no se encuentren localizados en un área.

  El proceso a llevar a cabo en estos casos es similar al anterior. Se ha de recurrir a la resta de BMPs (píxel a píxel) con el fin de localizar los píxeles que han sido modificados. Sin embargo, esta opción no puede emplearse en todos los tipos de formatos. Debido a que el formato JPEG, por ejemplo, codifica las imágenes en conjunto y no con mapa de bits, resulta imposible cambiar un píxel suelto. Esta optimización es válida para formatos como por ejemplo GIF, con la restricción de emplear una paleta fija para todas las imágenes, con el fin de que los índices accedan al color correcto.

  Esta optimización se implementó y su estudio no produjo una mejora significativa en los tiempos. Por un lado, la comprobación de cada píxel supone un gasto de tiempo elevado. Por otro lado, el uso de una paleta fija, hace que los colores obtenidos difieran de la realidad y ocasionen una calidad visual de la imagen muy pobre. En el único caso que puede ser viable es para las imágenes con sólo texto y aún así, el tamaño en formato GIF es mínimo, con lo que carece de sentido práctico. Otros factores como la restricción de formato impuesta, la sobrecarga para el proxy (mayor cuantos más clientes concurrentes) y la influencia del tamaño de la pantalla, han sido decisivos para prescindir de esta optimización en nuestra aplicación.

Las optimizaciones presentadas ganan importancia a medida que aumenta la resolución del dispositivo cliente. En el caso de la Nintendo DS, con su escasa resolución, las optimizaciones no influyen en gran medida.

4.2.4. Medidas de rendimiento

Se han llevado a cabo medidas de rendimiento para conseguir minimizar el tiempo empleado en mostrar las imágenes por pantalla. Con nuestra arquitectura se pretende que el usuario tenga la ilusión de navegación local, así que las capturas han de ser recibidas de manera fluida. Por tanto, será necesario conseguir reducir la latencia al mínimo posible y garantizar un ancho de banda suficiente.

Las dos medidas que hemos tenido en cuenta han sido el tiempo de compresión de la captura en un formato de imagen y su tamaño, que afecta al tiempo de envío. A continuación se detalla cómo se llevan a cabo estas medidas y los resultados que arroja su estudio.

Se ha de tener en cuenta el compromiso que existe entre calidad y tamaño: a mayor calidad, mayor tamaño de imagen. Se pretende encontrar el punto que equilibre este compromiso y genere resultados óptimos.

Método

La solución se verifica y estudia utilizando los dos dispositivos clientes nombrados en la primera sección de este capítulo, con el fin de generar envíos de capturas, la operación más costosa en tiempo y la que tiene un mayor potencial para minimizar la latencia. El procedimiento llevado a cabo para este estudio es la creación de una traza de imágenes a un tamaño acorde para cada dispositivo, que nos permita medir de forma consistente diferentes factores utilizando

ejecución directa.

En esta traza se encuentran tres tipos de imágenes:

- Imágenes con sólo texto
- Imágenes con sólo foto
- Imágenes mixtas (texto y foto)

Esta separación de tipos de imagen se debe a sus diferentes gamas de color, que tras varios análisis previos pudimos comprobar que es el factor determinante para obtener tamaños de fichero reducidos con diferentes formatos. Es importante disponer de estos tres tipos para poder calcular datos fiables, en media, de cómo reaccionan los dispositivos a un tamaño determinado de imagen independientemente del tipo de imagen que se esté enviando.

Ya que nuestro objetivo es obtener datos consistentes de todos estos bloques, hemos definido un amplio conjunto de imágenes para cada uno, que nos permitirán obtener promedios fiables de las características estudiadas. El conjunto de todas estas imágenes lo denominaremos banco de imágenes.

Se han realizado las siguientes actividades para cada formato estudiado:

1. **Codificación del banco de imágenes en el formato correspondiente**

   Se codifica cada imagen del banco de imágenes, explicado anteriormente, con el fin de medir el tiempo de conversión.

   Como ya hemos dicho, cada formato tiene características configurables que pueden hacer variar los tamaños y tiempos que se pretenden estudiar, luego han de ser tenidos en consideración. Por ello, se realizan experimentos con todas las configuraciones que permite cada formato, atendiendo al compromiso entre calidad, tamaño y tiempo. Según el tipo de formato, son:

   - **JPEG**: Variación de la calidad, de 0 a 100 % en intervalos de 10 %.
   - **GIF**: El formato GIF, al ser un formato que hace uso de paletas dentro de la imagen, ofrece la posibilidad de utilizar diferentes paletas ya predefinidas o incluso generar una propia. Además, permite elegir entre diferentes algoritmos de *dithering*. Las paletas estudiadas son tres y los algoritmos son siete.
   - **PNG**: Variación del ratio de compresión, de 0 a 9.

2. **Envío al dispositivo cliente**

   El siguiente objeto de estudio, una vez obtenidos unos parámetros de codificación razonables, es la transferencia de imágenes al dispositivo cliente. Como ya se ha dicho, se utilizan dos dispositivos cliente para realizar estas pruebas con el fin de probar diferentes resoluciones de imagen.

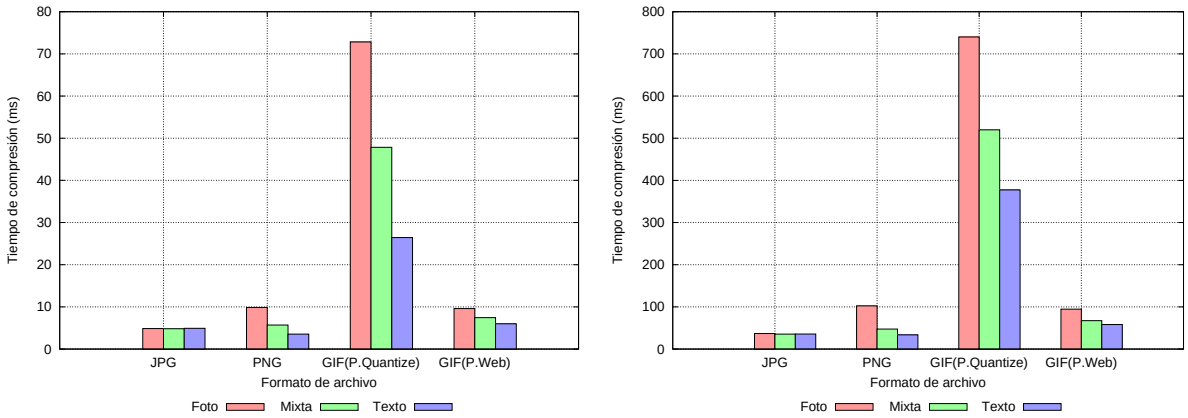
   Este paso busca calcular el tiempo de envío de cada imagen. Se comprueba que el tiempo de envío es proporcional al tamaño de la imagen enviada, por lo que va a ser un factor determinante en la elección del formato a utilizar.

Resultados y conclusiones

Analizando los resultados se llega a una idea clara del tiempo necesario para codificar las imágenes dependiendo del formato elegido y del tipo de imagen. Se analiza no sólo los tiempos obtenidos y los tamaños, sino la calidad visual de la imagen generada, ya que, en definitiva, es lo que es lo que marcará la experiencia de usuario.

El Anexo B incluye más conclusiones sobre el estudio que se ha realizado de los formatos gráficos disponibles. A continuación se presentan los resultados y conclusiones finales de dicho estudio.

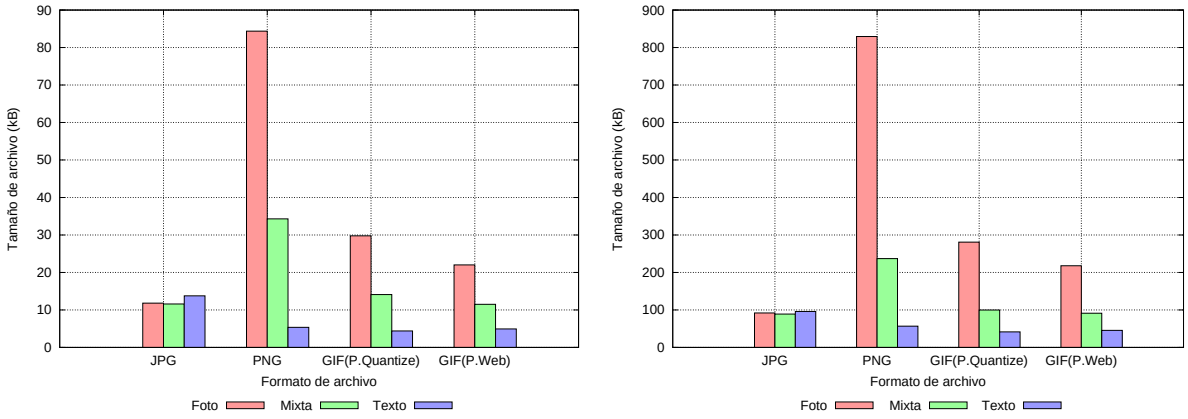
De entre las distintas configuraciones posibles para cada uno de los formatos, se selecciona la que genera mejores resultados. Eligiendo las características óptimas para cada formato, la Figura 4.3 presenta los resultados obtenidos para el tiempo de compresión.



(a) Nintendo DS **(b)** Ordenador portátil

Figura 4.3: Tiempos de compresión

Los tamaños de imagen generados, que afectan directamente al tiempo de envío de las capturas, son los recogidos en la Figura 4.4 a modo de resumen.



(a) Nintendo DS **(b)** Ordenador portátil

Figura 4.4: Tamaños de archivo

Como se desprende de la Figura 4.3, el tiempo que se dedica a la compresión de las imágenes es, en media, menor en el formato JPEG que en cualquiera de los demás formatos. Si nos centramos únicamente en las imágenes con sólo texto, PNG es una opción algo mejor. Además vemos como, en las pruebas a mayor tamaño, la relación sigue siendo proporcional a los resultados de las pruebas en la Nintendo DS.

En la Figura 4.4 que recoge el tamaño de los archivos generados según el formato utilizado, se observa cómo, en media, los resultados son mejores nuevamente para el formato JPEG. También en esta ocasión, los tamaños resultantes de la conversión son peores para JPEG cuando la imagen es únicamente texto. En este caso concreto, los formatos GIF y PNG tienen una clara ventaja.

Teniendo en cuenta el conjunto de los resultados obtenidos se decide implementar el envío de capturas utilizando como formato JPEG con una calidad del 70 %. Sin embargo, se deja implementada la posibilidad de usar el formato GIF si lo que se va a visualizar es texto, en cuyo caso, como hemos visto, los resultados relativos al tamaño y al tiempo de compresión son mejores para este formato.

Capítulo 5
Conclusiones

Tal y como se planteaba en los objetivos del proyecto, se ha diseñado una arquitectura viable para el acceso a Internet desde dispositivos de baja capacidad computacional. La arquitectura permite el acceso desde el cliente a cualquier servicio que sea capaz de mostrar el proxy, no siendo necesaria la actualización de su software a medida que aparecen nuevas tecnologías y estándares.

Para lograr estos objetivos, se ha definido un protocolo de comunicaciones y se ha implementado el servidor proxy. Además, se ha realizado una implementación de demostración como dispositivo cliente sobre la videoconsola Nintendo DS con funcionalidad completa y un dispositivo cliente en PC emulando un portarretratos.

En el marco de este desarrollo, se ha realizado un estudio de la influencia del uso de diferentes formatos gráficos en el rendimiento de la solución. Para dicho estudio se han utilizado las dos implementaciones de cliente desarrolladas. Los resultados finales del estudio se han incluido satisfactoriamente en la solución.

Trabajo futuro A continuación, se presentan diferentes propuestas que pueden ser interesantes en futuros desarrollos de este proyecto:

- **Detectar en proxy el formato adecuado para el envío.**
  En la solución actual, es el usuario quien decide en qué formato enviar las imágenes mediante un selector en el proxy. En general, se prevé utilizar JPEG como formato habitual, pero se mantiene GIF como una opción para capturas de sólo texto. La mejora consistiría en que el proxy fuera capaz de, según el contenido que se esté visualizando, emplear el formato que produzca mejores resultados, realizando análisis sobre la imagen, comprobando el tamaño tras realizar una conversión directa, o aplicando conocimientos sobre el contenido HTML.

- **Streaming**
  Podría incorporarse a la solución un sistema de *streaming* de vídeo, de forma que el dispositivo cliente pudiese reproducir música o vídeo desde el servidor. Esto permitiría una navegación más completa, ya que no se perdería funcionalidad en páginas de vídeo. Nuestra aplicación consigue una velocidad adecuada para la navegación pero no para la reproducción de vídeo.