UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

COMPUTACIÓN

TRABAJO FIN DE GRADO

iCROM: Plataforma para la Gestión Integral de Presentaciones

José Alberto Granados Perea

Julio 2016

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Tecnologías y Sistemas de Información

COMPUTACIÓN

TRABAJO FIN DE GRADO

iCROM: Plataforma para la Gestión Integral de Presentaciones

Autor: José Alberto Granados Perea

Director: Carlos González Morcillo

Julio 2016

iCROM

Teléfono: 650918687
E-mail: josealberto.work@gmail.com

[1] http://www.gnu.org/licenses/gpl-3.0.txt

TRIBUNAL:

Presidente: _____

Vocal: _____

Secretario: _____

FECHA DE DEFENSA: _____

CALIFICACIÓN: _____

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

A mi madre,
sin ella, todo esto
no hubiese sido posible.
Gracias.

Resumen

Desde la primera versión de Presenter, precursor de PowerPoint, en 1985, el número de presentaciones multimedia realizadas ha crecido exponencialmente. Según datos de Microsoft, a diario se realizan unos 30 millones de presentaciones en PowerPoint. Las presentaciones multimedia son una herramienta muy eficaz a la hora de comunicar información.

El término multimedia implica la utilización de texto, imágenes y animaciones de forma conjunta. Sin embargo, muchos estudios coinciden en que el uso de multimedia no siempre consigue transmitir la información de un modo eficaz. Esto es debido a la acción, directa o indirecta, de ciertos factores que forman parte de la presentación y que pueden causar que el público pierda el interés y deje de prestar atención.

Las herramientas de edición de presentaciones existentes en el mercado se centran en dar un soporte adecuado para la edición y composición de diapositivas. Sin embargo, cuentan con importantes carencias en tres ámbitos de trabajo. En primer lugar, no ofrecen un soporte completo y de bajo nivel en la gestión de elementos multimedia. En algunos casos, no soportan todos los formatos existentes y cuentan con pocas opciones de adaptación. En segundo lugar, no ofrecen mecanismos de interacción bidireccional, de modo que no se utiliza el canal de retorno del público asistente, logrando un aprendizaje puramente pasivo. En tercer lugar no potencian el uso de dinámicas de gamificación, que tan buenos resultados ofrecen para captar la atención y plantear retos.

El presente Trabajo Fin de Grado surge de la necesidad de proporcionar un conjunto de herramientas para el despliegue efectivo de presentaciones multimedia que cubran los tres ámbitos de trabajo que no se satisfacen con las aplicaciones existentes, como el soporte de elementos multimedia de bajo nivel, opciones avanzadas de interactividad y aplicando novedosos paradigmas de gamificación.

Abstract

Since the first version of Presenter, the precursor of PowerPoint, the number of multimedia presentations have grown exponentially. According to Microsoft, every day 30 million of PowerPoints has made. Nowadays, multimedia presentations represent a very effective tool to communicate information.

The multimedia term implies the simultaneous use of text, images and animations. However, many scientific studies agree that multimedia is not always used properly for communicating information in an effective way. This is due to the action of certain factors that are part of the presentation and can cause that the public loses interest and stops paying attention.

Current presentation tools focus on giving adequate support for editing and composing slides. However, they have strong deficiencies in three areas of work. First, these tools do not give a complete support when dealing with multimedia elements. In some cases, they do not support all the existing multimedia formats and they only have a few options for adapting new content. Second, these tools do not provide bidirectional interaction mechanisms. For this reason, the feedback channel of the audience is not used in multimedia presentations, generating a strictly passive learning. In the third place, these tools do not strenght the use of gamification mechanisms. Many researches agree that these mechanisms provide good results by boosting audience's attention.

This final degree essay provides a set of tools for deploying multimedia presentations that cover the three areas of work that are not fulfilled with the existing tools, such as low-level support for multimedia elements, advanced interactivity options and new gamification paradigms.

AGRADECIMIENTOS

Hay tantas personas a las que agradecer este trabajo que no sabría ni por donde empezar, por lo que intentaré ser conciso y breve. En primer lugar me gustaría agradecer a mi familia el haber estado ahí siempre que he necesitado su apoyo. En segundo lugar, a mis compañeros de carrera, en especial a **Abel Lorente**, **Juan Alfredo García** y **Darío Merino** por todas las experiencias que hemos vivido juntos. En tercer lugar, a mis compañeros del Centro de Tecnologías y Contenidos Digitales, con los que he vivido tanto en tan poco tiempo. Por último, pero no por ello menos importante, me gustaría agradecer la bondad, la paciencia y el apoyo que me ha ofrecido **Esperanza Hidalgo** este último año. A todo ellos, y a los que me dejo en el tintero, os doy las gracias de corazón.

José Alberto Granados Perea

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE LISTADOS

CAPÍTULO 1

INTRODUCCIÓN

En el mercado actual, existen multitud de herramientas para la producción de presentaciones y su puesta en escena. Sin embargo, la mayoría cuentan con opciones similares de edición y despliegue de diapositivas multimedia, olvidando por completo el canal de retorno del público objetivo de la presentación. La interacción en una presentación puede llegar a ser una gran aliada a la hora de potenciar las capacidades de enseñanza-aprendizaje en docencia presencial. De este modo, es posible utilizar dispositivos específicos (como "clickers" de votación) o de propósito general (como smartphones o tablets) como elementos dinamizadores de la presentación, obteniendo la respuesta del público en tiempo real.

La interacción con el público en una presentación se puede conseguir de muchas maneras y más si se combina con nuevos paradigmas de comunicación [12] y gamificación [17]. Es posible integrar herramientas existentes de participación basadas en micro-post, como por ejemplo mostrando *tuits* que se han hecho con un cierto *hashtag* durante la presentación, o un pequeño juego de preguntas en el cual el público compita por equipos o de forma individual por conseguir la máxima puntuación y que los resultados se muestren en tiempo real, etc.

En el ámbito de la gamificación, se pueden utilizar diversas mecánicas de los juegos del ámbito educativo y de las presentaciones multimedia, con el fin de conseguir mejores resultados en términos de interiorizar mejor ciertos conocimientos, mejorar la atención o recompensar el uso del canal de retorno en la comunicación. Este tipo de técnicas ganan terreno en las dinámicas de formación por su esencial carácter lúdico, que facilita la asimilación de conocimientos de un modo mucho más divertido y generando una experiencia positiva en el usuario [14].

Por este motivo el presente Trabajo Fin de Grado diseña y desarrolla *iCROM*, una plataforma para la gestión integral de presentaciones que cubre aspectos de configuración y proyección de presentaciones multimedia avanzadas. *iCROM* se centra en completar los aspectos que el software existente de producción de presentaciones multimedia no ofrece en los siguientes tres ámbitos de trabajo: adaptar elementos multimedia de bajo nivel, ofrecer mecanismos de interacción avanzada y dotar de mecánicas de gamificación aplicadas a las presentaciones.

1.1 PRESENTACIONES MULTIMEDIA

Las presentaciones habitualmente incluyen elementos multimedia, combinando información visual (imágenes, vídeos y animaciones) con información verbal (habitualmente texto escrito) y en algún caso elementos sonoros. El uso adecuado de ambos canales de comunicación (visual y verbal) de forma combinada potencia el aprendizaje y la retención de conceptos [24]. En la actualidad existen multitud de herramientas para la generación de presentaciones (como PowerPoint, Keynote, Prezi, Impress, Swipe, etc) que permiten añadir elementos multimedia. Sin embargo, en muchos casos estas herramientas no permiten adaptar a bajo nivel la visualización de los elementos multimedia y están limitadas a un pequeño subconjunto de formatos soportados.

1.1.1 Objetivo de las Presentaciones Multimedia

Las presentaciones multimedia son muy útiles debido a su carácter informativo, facilitando la transmisión de contenidos de una manera rápida, directa y eficaz. Aún siendo muy útiles en esta tarea, de nada sirve transmitir información si ésta es descartada o parcialmente asimilada por las personas que la reciben. Según los trabajos del Dr. Medina [16], el cerebro no presta atención a la información que no le resulta interesante por algún motivo, o en aquellos casos que se supera la capacidad de la memoria de trabajo. Así, el contenido multimedia debe ser correctamente presentado para evitar los efectos de la pérdida de atención, principalmente debidos a que existe una alta tasa de ruido visual, información redundante en el canal visual y verbal, elementos distractores, etc.

La interactividad durante una presentación puede ser una gran aliada a la hora de solventar esta problemática, debido a que está demostrado que las personas aprenden más haciendo que viendo o escuchando [29]. Si el ponente de la presentación consigue que exista esa interacción bidireccional durante la misma presentación, la información que se transmita al público será más fácil de procesar, entender y almacenar en la memoria a largo plazo. Esta interacción se puede conseguir fácilmente mediante la gamificación de las presentaciones, ya que numerosas investigaciones sostienen que el uso de las mecánicas ampliamente estudiadas en el mundo de los videojuegos aporta gran cantidad de efectos positivos en la asimilación de conocimiento [14].

1.1.2 Problemática

Las herramientas de generación de presentaciones existentes en el mercado actual se centran en dar un soporte adecuado a la edición y composición de diapositivas. Sin embargo, cuentan con importantes carencias en tres ámbitos de trabajo:

- **Gestión de elementos multimedia**. El soporte ofrecido para la carga y representación de elementos multimedia está limitada a un subconjunto de formatos soportados. El software de creación de diapositivas no suele permitir la integración de reproductores externos o la incorporación de nuevos formatos multimedia. Además, las opciones de reproducción de los formatos soportados no permiten operaciones de bajo nivel, tales como recortar fragmentos del elemento multimedia o la reproducción forzada en cierto intervalo.

- **No ofrecen mecanismos de interacción bidireccional**. En la mayoría de los casos, el software de diseño de diapositivas no permite incorporar el canal de retorno del público ni incorporar saltos condicionales en el contenido de la presentación. Así, el aprendizaje conseguido en la presentación es puramente pasivo.

- **No ofrecen opciones de gamificación**. Existe una tendencia a utilizar las dinámicas de videojuegos que captan la atención de los jugadores en diversos ámbitos que no son puramente lúdicos. Las presentaciones igualmente pueden beneficiarse de este tipo de técnicas, siempre que el software ofrezca un soporte directo para su utilización. *iCROM* permite añadir ciertas dinámicas de gamificación con concursos, retos y estadísticas en tiempo real que facilitan añadir este tipo de comportamientos.

1.2 IMPACTO SOCIO-ECONÓMICO

El uso de las presentaciones multimedia se realiza a lo largo y ancho del planeta. Este uso abarca casi cualquier ámbito, ya que el uso de presentaciones multimedia es una manera fácil, rápida y sencilla de transmitir conocimientos a un cierto público.

Según un estudio realizado por T. Escudero y A. Correa en innovación educativa [30], la proyección diaria de presentaciones multimedia en el ámbito educativo se cuenta por millones, y no es de extrañar, ya que esta tecnología revolucionó hace ya unos años la forma de dar las clases magistrales. En 2002, ya existían 4 millones de presentaciones multimedia de clases magistrales en formato de *PowerPoint*, cifra que ha ido aumentando exponencialmente según han ido pasando los años. PowerPoint se puede considerar como la herramienta ofimática que más éxito ha tenido hasta el momento a la hora de crear y reproducir presentaciones multimedia, título que defienden sus casi 20,6 millones de usuarios.

El uso de este software de presentación tan extendido no se limita únicamente al ámbito docente, sino que a diario se utiliza para realizar presentaciones de ámbito profesional o comercial, por lo que el impacto de las presentaciones en la sociedad es extremadamente alto.

1.3 ESTRUCTURA DEL DOCUMENTO

Este documento se ha estructurado según las indicaciones de la normativa de trabajos de fin de grado de la Escuela Superior de Informática de la Universidad de Castilla-La Mancha, incluyendo los siguientes capítulos:

- **Capítulo 2. Objetivos**: en este capítulo se describen cada uno de los objetivos y subobjetivos a alcanzar por la plataforma *iCROM*.
- **Capítulo 3. Antecedentes**: en este capítulo se hace un repaso de las tecnologías y conceptos teóricos estudiados para el diseño e implementación del presente trabajo fin de grado.
- **Capítulo 4. Método de Trabajo**: en este capítulo se describe la metodología empleada en el desarrollo, así como los recursos software y hardware empleados.
- **Capítulo 5. Arquitectura**: en este capítulo se describe el diseño y la implementación de la plataforma, detallando inconvenientes surgidos durante el proceso de desarrollo y las soluciones aplicadas a los mismos. Se descompone la plataforma en aplicaciones y éstas en módulos y submódulos, describiendo en detalle cada una de las partes por separado.
- **Capítulo 6. Evolución y Costes**: en este capítulo se describe la evolución de la plataforma durante su desarrollo siguiendo la metodología elegida para ello. También se dan cifras acerca del coste del desarrollo, así como algunas estadísticas del mismo.
- **Capítulo 7. Resultados y Conclusiones**: en este último capítulo se hace un breve resumen de los objetivos alcanzados, dando una serie de resultados y pruebas realizadas, así como ideas para continuar el trabajo y una conclusión personal.

CAPÍTULO 2

OBJETIVOS

En este capítulo se enumeran y describen cada uno de los objetivos y subobjetivos planteados para el presente Trabajo Fin de Grado.

2.1 OBJETIVO GENERAL

El principal objetivo de *iCROM* es el diseño e implementación de una plataforma para la gestión integral de presentaciones. Este objetivo general se desglosa en una serie de objetivos funcionales y no funcionales que se detallan con más profundidad en la siguiente sección.

2.2 OBJETIVOS ESPECÍFICOS

Los objetivos y subobjetivos específicos de *iCROM* se pueden catalogar en base a una taxonomía de **tres** ámbitos de trabajo:

- A. *Gestión de elementos multimedia*. El objetivo principal de este ámbito de trabajo es proporcionar controles avanzados de despliegue de elementos multimedia. En concreto:
  - A.1 Diseño e implementación de mecanismos para incorporar elementos multimedia con soporte para el despliegue de bajo nivel (al menos modo de predicción de relación de aspecto y zonas de recorte).
  - A.2 Soporte para el uso de presentaciones diseñadas con otras herramientas ofimáticas de creación y edición de diapositivas.
  - A.3 Extensibilidad. El sistema deberá estar basado en estándares libres y multiplataforma.
  - A.4 Diseño e implementación de herramientas auxiliares para facilitar el despliegue y distribución de presentaciones multimedia.

  - A.5 Diseño de la estructura de los archivos de flujo y configuración de las presentaciones multimedia basados en pequeños lenguajes de marcado y serialización.

- B. *Mecanismos de interacción bidireccional*. La plataforma proporcionará mecanismos para capturar el canal de retorno del público asistente de la presentación, así como opciones de interacción entre el ponente y la presentación. Específicamente:

  - B.1 Habilitar diferentes canales de comunicación con el público.
  - B.2 Proporcionar mecanismos de interacción basados en estándares libres multiplataforma. En concreto, la interacción podrá realizarse desde cualquier navegador web existente en PC o dispositivos portátiles.
  - B.3 Optimizar el uso de datos para que sea posible la interacción con bajo ancho de banda y alta latencia.
  - B.4 Habilitar mecanismos para la definición de teclas personalizadas que permitan realizar ciertas acciones en la presentación (al menos pantalla en negro, pantalla en blanco y un modo mosaico).
  - B.5 Proporcionar opciones de envío de eventos tales como pulsaciones de teclas y traducciones entre la plataforma y otras aplicaciones. De este modo, no será necesario salir de la aplicación para enviar eventos a aplicaciones auxiliares y todo el control se podrá realizar con un mando inalámbrico.

- C. *Dinámicas de gamificación*. La plataforma dará soporte para la integración de dinámicas de gamificación. En concreto:

  - C.1 Proporcionar mecanismos para la gamificación de presentaciones mediante la inclusión de ciertos elementos multimedia extensibles con los que se pueda interactuar.
  - C.2 Diseño e implementación de una aplicación basada en estándares libres multiplataforma, fácil de instalar y que haga uso de una dinámica de concurso sencilla y con la que se pueda interaccionar mediante el uso de diferentes tipos de dispositivos (al menos mediante el uso de dispositivos móviles o mediante el uso de PC).
  - C.3 Diseño de pequeños archivos de secuencia y configuración para la definición de preguntas, respuestas y archivos multimedia (al menos imágenes y vídeos).

CAPÍTULO 3

ANTECEDENTES

El cumplimento de los objetivos propuestos para *iCROM* (**Capítulo 2**) requiere de un conocimiento específico en ciertas áreas de la informática, tales como la reproducción de contenido multimedia, la ingeniería del software, las bases de datos, la representación de gráficos 3D y la visualización por computador, etc.

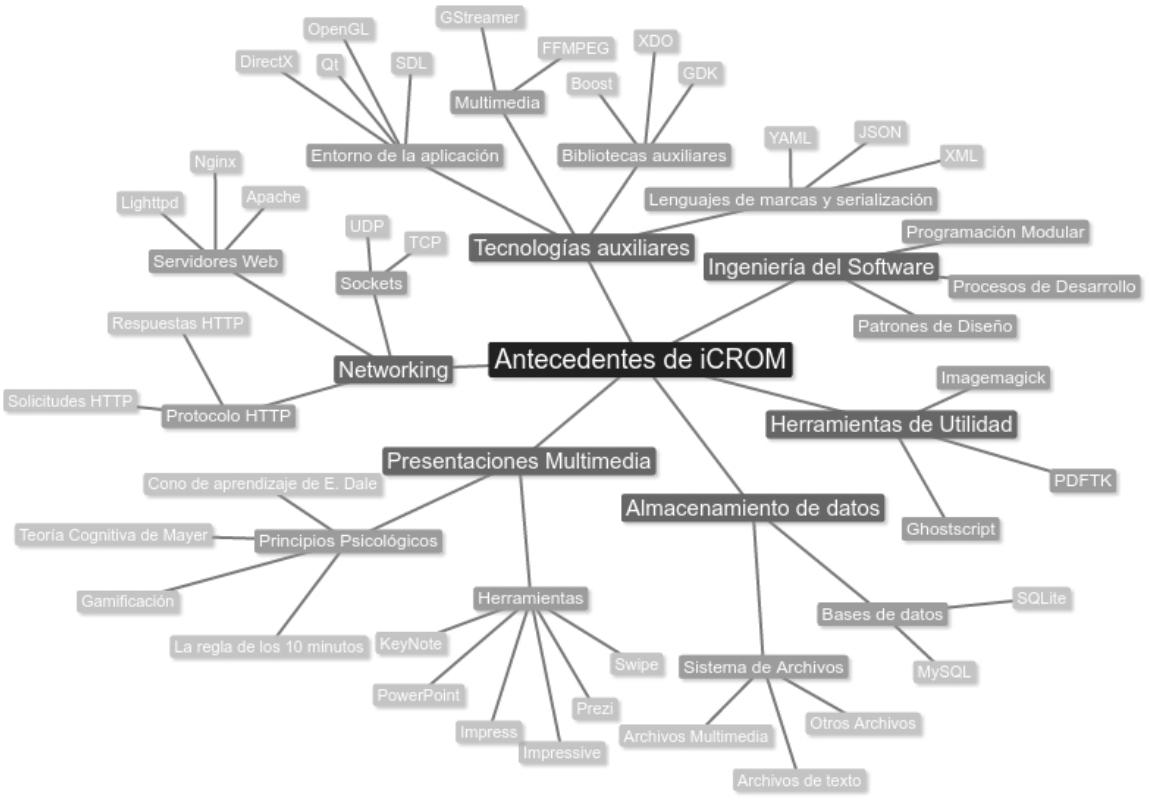


Figura 3.1: Mapa conceptual de antecedentes de *iCROM*.

En este capítulo se enumeran los conceptos estudiados para la realización del presente Trabajo Fin de Grado. Estos conceptos se pueden ver a modo de mapa conceptual en la figura 3.1. Siguiendo dicho mapa conceptual, el estado del arte se realizará describiendo por partes cada uno de los conceptos estudiados, analizando en detalle su utilidad a la hora de

alcanzar los objetivos propuestos, así como ventajas y desventajas de cara a tecnologías o conceptos similares a los elegidos para el desarrollo de la plataforma.

3.1 PRESENTACIONES MULTIMEDIA

Con el fin de crear una herramienta única para el despliegue de presentaciones efectivas, se han buscado y estudiado aquellas herramientas actualmente disponibles para el despliegue de presentaciones multimedia, con el fin de obtener ideas, evitar fallos conocidos, etc. Además, como esta plataforma pretende ofrecer herramientas para que este tipo de presentaciones no se hagan aburridas y monótonas, también se han estudiado una serie de principios psicológicos relacionados con este tema. A continuación, se describirán cada uno de los estudios realizados en este ámbito.

3.1.1 Principios psicológicos

Para poder ofrecer una serie de herramientas útiles a la hora de hacer las presentaciones más asequibles, entretenidas y educativas, se ha hecho un estudio acerca de cómo el ser humano escucha y aprende, de cómo procesa mejor la información que recibe del entorno, qué elementos le permiten adquirir conocimientos de forma más sencilla, etc.

Teoría Cognitiva del Aprendizaje Multimedia

Esta teoría defiende que el uso de diferentes formas de representación de la información facilita el aprendizaje. Richard E. Mayer, psicólogo educacional de la Universidad de California, ha contribuido enormemente en esta teoría. Para Mayer, el cerebro aprende más fácilmente si se integran palabras e imágenes que si sólo se utilizan palabras o imágenes por separado.

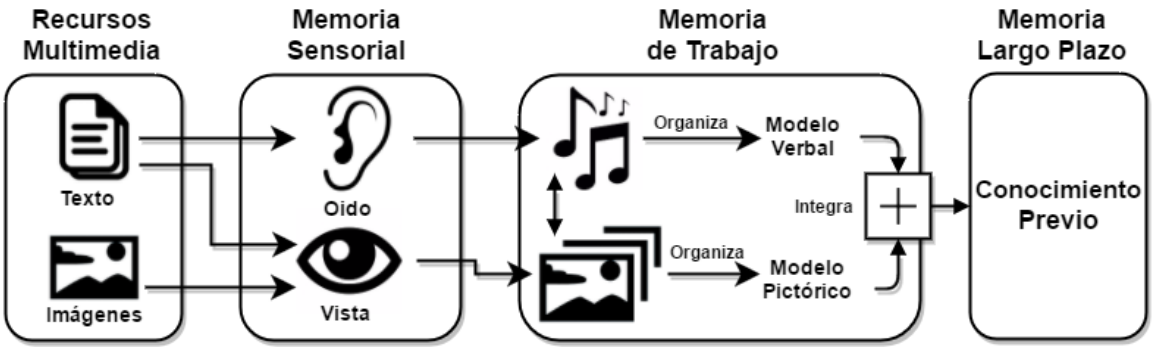


Figura 3.2: Esquema del aprendizaje multimedia [24].

Para conocer mejor cómo se procesa la información obtenida a partir de cierto contenido multimedia es necesario conocer como trabaja la mente humana y de que manera ha de presentarse esta información para que la memoria la almacene mejor a largo plazo. Atendiendo a la Figura 3.2, se puede observar que partiendo de una serie de recursos multimedia (imágenes estáticas o dinámicas, así como texto narrado o escrito) se originan ciertos estímulos sensoriales. Estos estímulos son captados por la memoria sensorial a través de la vista y el oído. Esta memoria descarta continuamente aquellos estímulos que considera que no son relevantes en ese momento. Los estímulos que se consideran relevantes son procesados en una cantidad de tiempo relativamente pequeña con el fin de pasarlos lo más rápidamente posible a la memoria de trabajo para que ésta genere a su vez una serie de representaciones consistentes a partir de los mismos datos procesados (modelo verbal y modelo pictórico). Por último, estas representaciones se integran en la memoria a largo plazo estableciendo relaciones con los conocimientos previos adquiridos, con el fin de afianzar el recuerdo y obtener un aprendizaje mayor [24].

Richard E. Mayer plantea que hay dos propósitos principales en el aprendizaje:

- *Recordar*, permite retener la nueva información que se desprende del material multimedia presentado.
- *Entender*, permite construir una representación mental coherente de dicho material multimedia.

Para Mayer, el *aprendizaje multimedia* es aquel aprendizaje en el que el sujeto logra construir conocimiento mediante la creación de representaciones mentales de la información obtenida de cierto contenido multimedia de una presentación. La utilización de varios estímulos de forma simultánea tiene efectos muy positivos a la hora de integrar el conocimiento adquirido en la memoria de largo plazo. Para demostrar esto, Richard E. Mayer realizó un experimento con dos grupos de alumnos [23]. Al primer grupo se le presentó información mediante un único canal (auditivo o visual), mientras que al segundo grupo se le presentó información mediante el uso de los dos canales.

El resultado que se obtuvo tras someter a los dos grupos a una serie de preguntas sobre la información presentada fue el siguiente: los alumnos del segundo grupo (G2) acertaron el doble de preguntas que el primero (G1), demostrando así, que el cerebro aprende mejor mediante el empleo de los canales auditivo y visual juntos que por separado.

Atendiendo a estos resultados (Fig. 3.3), Mayer propuso una serie de principios para el diseño de contenido multimedia con el fin de generar material eficaz para la presentación de información a través de este tipo de contenido [25].

Estos principios se basan en tres tipos de procesamiento que realiza el cerebro a la hora de adquirir información desde un cierto contenido multimedia:

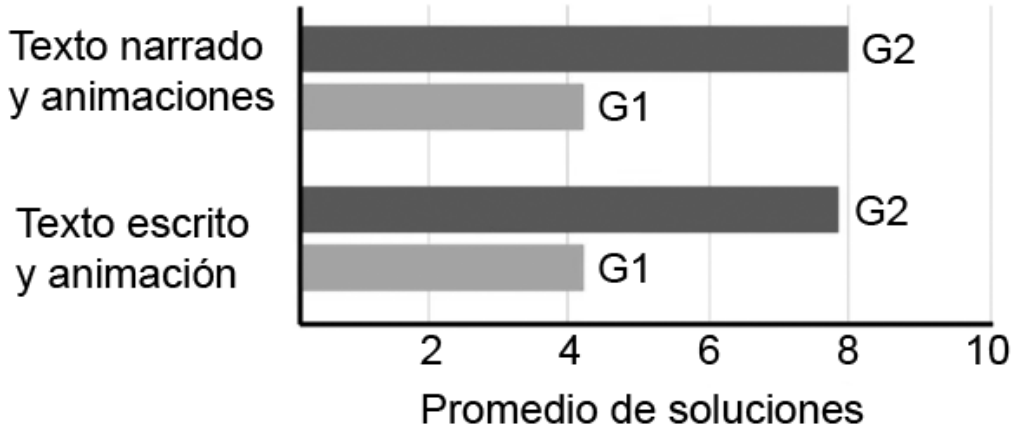


Figura 3.3: Resultado del experimento realizado a los grupos de alumnos [23].

- **Procesamiento Externo**: Tiene lugar cuando el diseño del material multimedia no es el adecuado. El objetivo para este procesamiento es minimizarlo mediante el empleo de los siguientes acciones:
  - Eliminar todos aquellos datos que no son necesarios o no están relacionados con la información que se está transmitiendo (*Coherencia*).
  - Indicar aquellos datos de los que se está informando, ocultando o mitigando aquellos que no son necesarios durante ese momento (*Señalar*).
  - Eliminar datos repetidos (*Redundancia*).
  - Intentar que los elementos relacionados estén cercanos en el espacio (*Continuidad Espacial*).
  - Conseguir que la información esté alineada en el tiempo, en el sentido de que las referencias a datos ya transmitidos no se alejen mucho, temporalmente hablando (*Continuidad Temporal*).
- **Procesamiento Esencial**: Este tipo de procesamiento está relacionado con la complejidad del material multimedia que se presenta. El objetivo en este caso es gestionar de manera correcta este tipo de procesamiento utilizando para ello los siguientes acciones:
  - Dividir el contenido multimedia en unidades más asequibles (*Segmentación*).
  - Habilitar mecanismos para la adquisición de los conocimientos previos necesarios a un determinado contenido multimedia (*Pre-training*).
  - Anteponer el texto narrado al texto escrito (*Modalidad*).
- **Procesamiento Generativo**: Este procesamiento tiene por objetivo dar un significado a la información obtenida del material multimedia presentado. Potenciar este tipo de procesamiento permite mejorar la adquisición de conocimiento a largo plazo. Para conseguir esto, se pueden aplicar los siguientes acciones:

- Comunicar la información de manera personal y directa (*Personalización*).
- Anteponer la voz humana a la voz sintetizada por computador (*Voz*).
- Usar preferiblemente cuerpos con aspecto humanoide (*Cuerpo*).
- Evitar imágenes estáticas innecesarias que generen ruido visual (*Imagen estática*).

Cono de aprendizaje de Edgar Dale

Para poder entender mejor como aprende el ser humano, resulta muy común utilizar el cono de aprendizaje de Edgar Dale para ilustrar qué acciones ayudan más al ser humano en el aprendizaje de nuevos conocimientos y qué acciones ayudan menos.

Edgar Dale fue un famoso pedagogo estadounidense que realizó una serie de contribuciones a la instrucción visual y educativa. Una de sus contribuciones fue el famoso *Cono de Aprendizaje*, también conocido como *Cono de la Experiencia*. Este cono representa la profundidad del aprendizaje realizado con la ayuda de diversos medios.

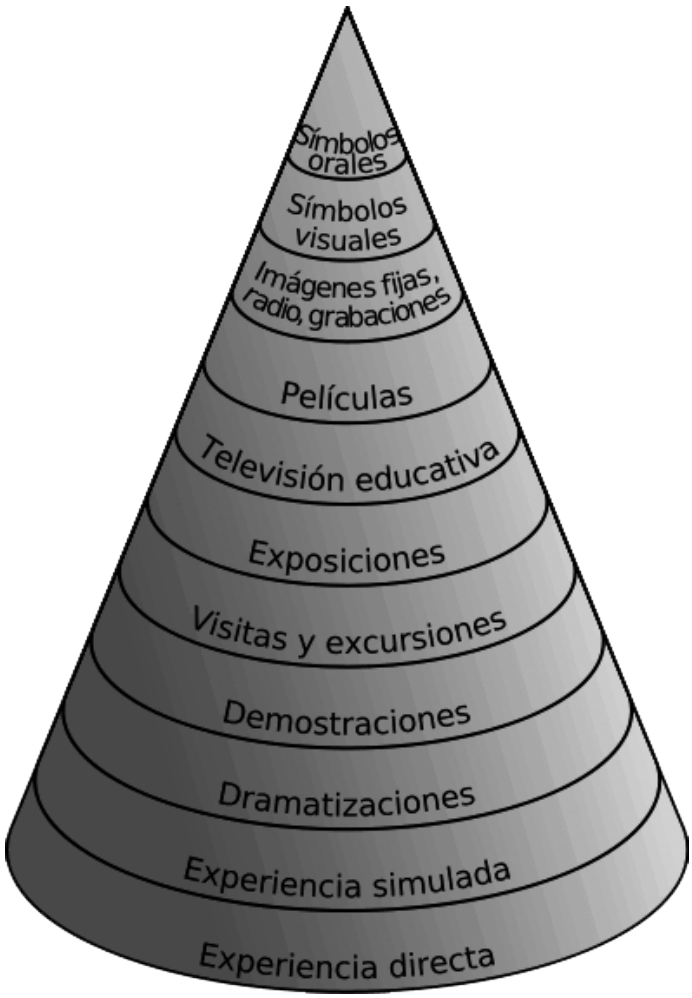


Figura 3.4: Cono de aprendizaje de Edgar Dale.

En la figura anterior se puede observar el cono de la experiencia de Edgar Dale, traducido de la versión original que éste creó en 1969 donde ya incluía la televisión como una categoría más. Dale advirtió en más de una ocasión contra el mal uso de este cono, afirmando que las categorías que aparecían en el mismo no se debían considerar inflexibles ni como una jerarquía de distintos rangos. Lo que Edgar Dale quería dejar claro con esta estructura es que para él la mente humana almacenaba mejor la información de aquellas cosas que requerían de actividad o participación directa. Esto se conoce como *aprendizaje activo* y se representa como la base del cono. Por el contrario, Dale afirmaba que el ser humano retenía peor aquellas acciones que conllevaban un *aprendizaje pasivo*, aquel resultante de actividades verbales o escritas (parte superior del cono).

Si lo anterior se aplica al campo de las presentaciones multimedia, se puede deducir que de una presentación (siguiendo las categorías del cono de la experiencia) se obtiene un aprendizaje pasivo. Para fortalecer este aprendizaje y hacer que el conocimiento que se intenta dar durante la presentación multimedia llegue mejor al público de la misma, es responsabilidad del ponente pasar el aprendizaje pasivo a un aprendizaje activo haciendo que el público de la presentación participe activamente en la misma de alguna manera.

La regla de los 10 minutos

Uno de los problemas de las presentaciones multimedia extensas es que éstas normalmente conllevan una perdida de atención progresiva del público asistente desde el inicio de la presentación hasta el final de la misma. John J. Medina, biólogo molecular especializado en el estudio del cerebro, publicó una serie de reglas para sacar el máximo partido al cerebro [16]. Una de estas reglas, la número cuatro, establece que el ser humano por naturaleza no presta atención a las cosas aburridas.

Esto se debe a que el cerebro descarta constantemente información que no le es relevante, focalizándose exclusivamente en un punto en concreto. Es en este punto donde entra en juego la regla de los 10 minutos. El cerebro es capaz de mantener la concentración en un punto determinado entre un intervalo de tiempo de 7 a 10 minutos. Una vez pasados esos minutos, el cerebro empezará a descartar información procedente de ese punto porque ya estará focalizándose en otro distinto.

Por este motivo, en las presentaciones multimedia extensas es muy importante hacer una serie de pausas cada 7 o 10 minutos con el fin de mantener siempre la atención del público activa en la presentación. Estas pausas no tienen por que ser descansos como tales, pueden ser otro tipo de acciones que lleven al público de la presentación a cambiar su foco de atención a algo que el ponente de la presentación elija. Para conseguir esto, existen numerosos métodos y técnicas. Unos ejemplos podrían ser los siguientes:

- Descansos breves de no más de 5 minutos si la presentación supera los 60 minutos de duración.

- Acciones interactivas. Esto se refiere a hacer pequeños cuestionarios, preguntas sueltas o ejercicios relacionados con la información transmitida hasta ese momento.

- Cambios de posición en caso de haber estado más de diez minutos en el mismo lugar o en la misma zona.

- Introducción de temas de conversación relacionados con el tema de la presentación (anécdotas, historias vividas, etc).

Atendiendo a la figura 3.5, se puede observar como la atención del público de una presentación de una hora de duración decrece hasta que pasan los diez primeros minutos, manteniéndose en esa línea hasta el final de la presentación.

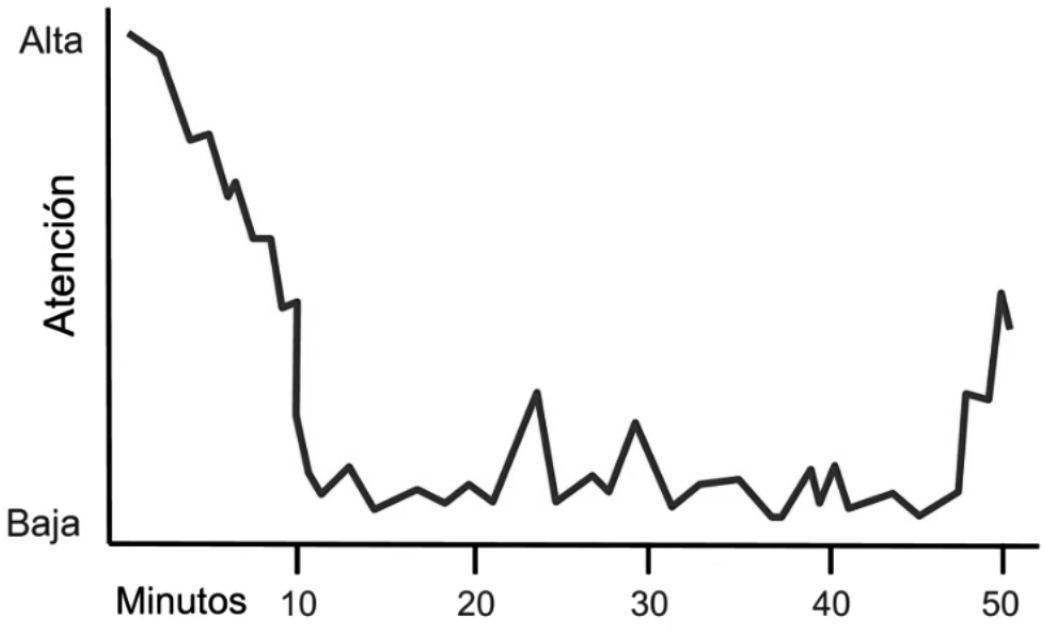


Figura 3.5: Modelo de atención en una clase magistral de unos 60 minutos de duración [16].

Mediante el empleo de este tipo de técnicas, una presentación multimedia extensa puede resultar más asequible y provechosa para el público asistente al conseguir mantener el máximo tiempo posible la atención que estos puedan prestar, pudiendo evitar así casos como el anterior.

Gamificación en Presentaciones Multimedia

El concepto de gamificación no se basa en la creación o el uso de videojuegos, sino en aprovechar ese componente adictivo que se desprende de éstos, el cual se puede aplicar en diferentes entornos con el fin de atraer al usuario y lograr que realice ciertas acciones de forma satisfactoria [6]. La gamificación intenta satisfacer algunas de las necesidades humanas básicas, tales como: *reconocimiento*, *recompensa*, *logro*, *competencia*, *colaboración*, *autoexpresión* y *altruismo*.

Para satisfacer estas necesidades, en la gamificación se utilizan distintos elementos que junto a la estética del juego forman la experiencia del jugador. Para Kevin Werbach [18], los elementos que se utilizan son las dinámicas, las mecánicas y los componentes.

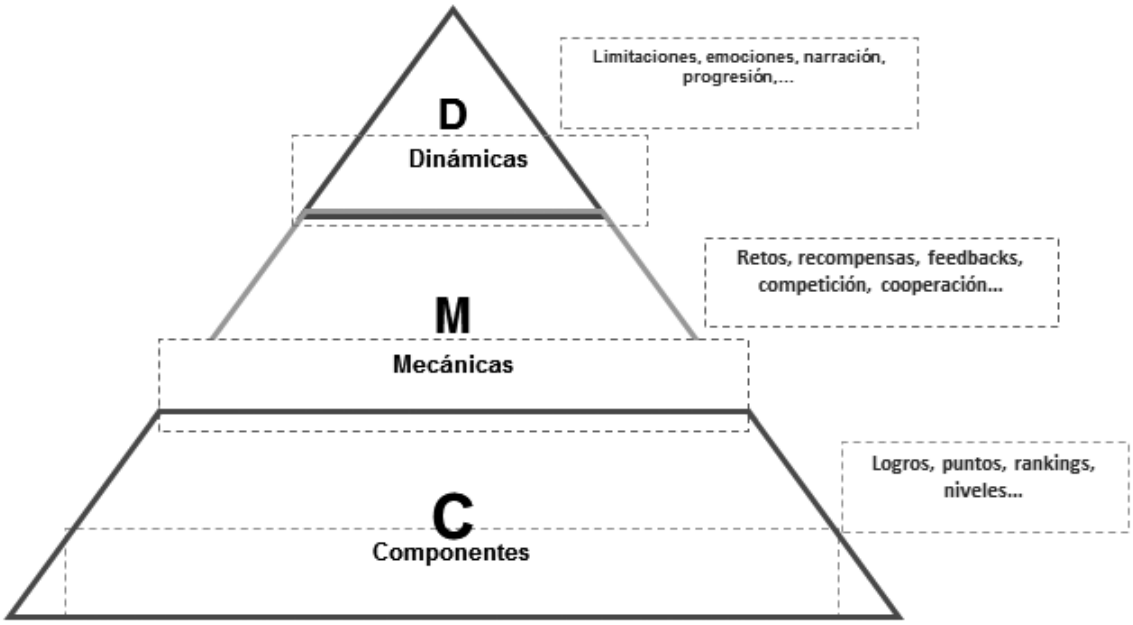


Figura 3.6: Pirámide de los elementos de gamificación [6, 18].

- *Dinámicas*: representan el concepto, la estructura implícita del juego.
- *Mecánicas*: son los procesos que provocan el desarrollo del juego. Se pueden distinguir tres tipos: mecánicas sobre el comportamiento, mecánicas de retroalimentación y mecánicas de progresión.
- *Componentes*: representan las representaciones específicas de las dinámicas y las mecánicas.

El uso de estos elementos por si mismos no hacen un juego. Un bueno juego se hace en la forma en la que estos elementos se entrelazan para hacer que el jugador final se divierta.

Numerosas investigaciones mantienen que el uso de videojuegos pueden aportar gran cantidad de efectos positivos en la enseñanza. James Paul Gree defiende que los buenos videojuegos son *máquinas para aprender* [14] porque incorporan algunos de los principios de aprendizaje más importantes establecidos por la ciencia cognitiva actual [6].

Los buenos videojuegos:

- Proporcionan la información demandada por los usuarios según sea necesario.
- Presentan a los usuarios tareas desafiantes pero al mismo tiempo sencillas de realizar.
- Convierten a los usuarios en creadores y no en meros receptores.
- Constan de niveles iniciales que se han diseñado específicamente para proporcionar a los usuarios los conocimientos básicos necesarios para afrontar problemas más complejos.
- Crean un "ciclo de maestría" sobre una tarea específica.

Teniendo en cuenta los conceptos estudiados hasta este punto, los videojuegos pueden ser grandes aliados en las presentaciones multimedia debido a que:

- Requieren de un pequeño receso para poder ser jugados. Este receso podría servir de descanso siguiendo la regla de los 10 minutos.
- Requieren de la participación activa del público de la presentación, fortaleciendo el aprendizaje pasivo con el aprendizaje activo.
- Permiten fortalecer el aprendizaje a largo plazo de los conocimientos transmitidos durante la presentación multimedia hasta ese momento, al establecer relaciones entre lo escuchado y visto durante la presentación con las experiencias obtenidas de la participación en el juego.

3.1.2 Herramientas para Presentaciones Multimedia

Por último, se han en cuenta aquellas herramientas ya implementadas para la creación y despliegue de presentaciones multimedia. Bien es sabido que con este Trabajo Fin de Grado no se pretende reinventar la rueda, si no mejorar el diseño que ya existe de la misma. Para mejorar estas herramientas en algún sentido, primero ha sido necesario estudiarlas para comprender y anotar cada una de las funcionalidades que éstas ofrecen. A continuación se hablará de cada una de estas herramientas, de sus funcionalidades y de sus ventajas en su uso frente a otras herramientas similares.

- **Microsoft PowerPoint**[1]: Desarrollado por *Microsoft*, PowerPoint representa la herramienta para presentaciones multimedia por excelencia. A pesar de ser una herramienta que requiere de una licencia de pago para su uso, está presente en casi cualquier ordenador doméstico o de oficina. Viene junto a un paquete con más herramientas ofimáticas y se puede ejecutar tanto en Windows como en Mac OS.

  Como ya se ha dicho, esta herramienta permite la creación y visionado de presentaciones multimedia, y entre sus características se pueden encontrar las siguientes:

  - Uso de plantillas propias o creadas por el usuario.
  - Introducción de campos de texto totalmente editables, desde el contenido al color, pasando por animaciones, tamaño de letra, etc.
  - Inserción de imágenes prediseñadas o alojadas en el almacenamiento interno. Permite también crear animaciones con las mismas.
  - Permite añadir efectos visuales para las transiciones entre diapositivas.
  - Permite realizar gráficos, así como añadir vídeos y audio.

  PowerPoint sólo permite presentar archivos creados bajo la misma herramienta, los cuales tienen por extensión PPT, aunque sí permite exportarlas a diferentes formatos (como por ejemplo PDF).

- **Impressive**[2]: Esta herramienta es más sencilla que la anterior pero no por ello menos potente. A diferencia de la anterior, esta herramienta no requiere de licencia de pago al ser una herramienta de código libre. Su principal característica es que permite el visionado de archivos PDF a modo de presentación. Además, permite cambiar entre diapositivas utilizando efectos 3D. Está programada en Python y utiliza la librería PyGame para crear la ventana de la aplicación, capturar los eventos de teclado y ratón, etc. Una de sus funcionalidades que no se implementa en otras herramientas similares es su *modo mosaico*, el cual permite ver una gran cantidad de diapositivas en miniatura de la presentación al mismo tiempo, dando la capacidad al usuario de acceder rápidamente a una diapositiva en concreto en un momento dado.

- **OpenOffice Impress**[3]: Esta herramienta pertenece a un conjunto de herramientas ofimáticas distribuidas bajo el paquete de OpenOffice. Similar a PowerPoint, esta herramienta permite la creación y visionado de presentaciones multimedia, dando la capacidad al usuario de añadir imágenes prediseñadas o alojadas en el almacenamiento interno, reproducción de vídeo y audio, efectos entre diapositivas, etc. Aunque tiene su propio formato de archivo, Impress permite visionar y editar presentaciones creadas

[1] https://office.live.com/start/PowerPoint.aspx
[2] http://impressive.sourceforge.net/
[3] https://www.openoffice.org/es/producto/impress.html

con PowerPoint, además también permite exportar las presentaciones creadas con esta herramienta al formato PPT.

- **Prezi**[4]: Esta herramienta es una herramienta web para la creación y visionado de presentaciones multimedia. A diferencia de otras herramientas para el mismo propósito, Prezi se caracteriza por crear presentaciones muy dinámicas, donde los efectos entre diapositivas toman un papel muy importante. Gracias a su interfaz basada en zoom, Prezi permite a los usuarios disponer de una visión diferente que no pueden conseguir con otra herramienta para el visionado de presentaciones multimedia, dando la capacidad de poder crear presentaciones diferentes a lo habitual.

- **KeyNote**[5]: Esta herramienta para presentaciones multimedia ha sido desarrollada por Apple, por lo que representa otra opción para hacer y visionar presentaciones en este tipo de dispositivos. Al igual que PowerPoint, esta herramienta requiere de una licencia de pago para poder usarse. Una de sus características es que permite exportar a diferentes formatos, incluyendo PPT de PowerPoint. Además, tiene su propio formato para la definición de archivos de presentación (**.key**), y para la definición de temas creados por el usuario (**.kth**), ambos basados en *XML*.

- **Swipe**[6]: Esta herramienta es una herramienta web que permite la creación de presentaciones multimedia de forma gratuita o de pago, dependiendo de las necesidades del usuario. Entre sus principales características se encuentra el poder compartir las presentaciones, indistintamente del dispositivo o del lugar, con el fin de que el público de una presentación pueda interaccionar y participar en ellas. Swipe es de las pocas herramientas ofimáticas para la creación de herramientas multimedia que permiten aprovechar el canal de retorno del público de una presentación.

3.2 INGENIERÍA DEL SOFTWARE

Para poder desarrollar la plataforma *iCROM*, se requieren de conocimientos sobre ingeniería del software. Ésta se puede definir como *la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación asociada requerida para desarrollar, operar y mantenerlos* [4]. Siguiendo esta definición, es natural utilizar ingeniería del software a la hora de diseñar y construir proyectos software de cierta envergadura. De esta forma se pueden obtener resultados con una cierta calidad y en un cierto periodo de tiempo limitado. Para conseguir esto, la ingeniería del software dispone de ciertas

[4]https://prezi.com/
[5]http://www.apple.com/mac/keynote/
[6]https://www.swipe.to

técnicas, metodologías y procedimientos para obtener desde el *ciclo de vida* del proyecto hasta los diagramas necesarios para el diseño y la construcción del mismo.

A continuación, se hablará de los principales conceptos de la ingeniería del software estudiados para el presente Trabajo Fin de Grado.

3.2.1 Proceso de desarrollo

El proceso de desarrollo de un determinado proyecto consiste en la consecución de una serie de pasos que involucran ciertas actividades, restricciones, recursos, herramientas y técnicas para conseguir producir una determinada salida. De esta forma, el proceso de desarrollo software se puede definir como el conjunto de actividades necesarias para transformar los requisitos obtenidos de los usuarios en un sistema software [13].

Los procesos de desarrollo software se pueden dividir en *pesados* y *ligeros*, siendo los primeros aquellos que requieren de una mayor generación de documentación, mientras que los segundos favorecen la comunicación entre los miembros del equipo de desarrollo y los clientes. En general, los procesos de desarrollo software cuentan con las siguientes características: *Comprensión*, *Visibilidad*, *Fiabilidad*, *Robustez*, *Mantenibilidad* y *Rapidez*.

Para el desarrollo de este proyecto se han tenido en cuenta el *Proceso Unificado de Desarrollo* y *SCRUM*.

Proceso Unificado

El Proceso Unificado de Desarrollo, o simplemente Proceso Unificado, es un marco de desarrollo que destaca por:

- *Ser iterativo e incremental*: El sistema software resultante de este proceso de desarrollo se obtiene a través de una serie de iteraciones e incrementos, donde las iteraciones se pueden definir como secuencias de pasos en el flujo de trabajo mientras que los incrementos hacen referencia al crecimiento del proyecto. El Proceso Unificado divide el proyecto en subproyectos, los cuales son representados como iteraciones que resultan en un incremento.

- *Estar dirigido por casos de uso*: El resultado del proceso de desarrollo es debido a una serie de de necesidades por parte de los usuarios, las cuales se deben recoger al principio del desarrollo y reciben el nombre de requisitos funcionales. El conjunto de todos los requisitos representa el modelo de casos de uso del proyecto, el cual representa a su vez la funcionalidad total del sistema que se ha de obtener como resultado.

- *Estar centrado en la arquitectura*: La arquitectura de un software se describe mediante diferentes vistas en las que se incluyen las características más importantes del sistema. La arquitectura debe permitir el desarrollo de todos los casos de uso, los cuales deben evolucionar junto a la arquitectura en paralelo.

Este marco de desarrollo es un marco genérico que puede especializarse, permitiendo su aplicación en multitud de sistemas software, multitud de organizaciones o tamaños de proyecto [13]. Además, utiliza UML (*Lenguaje Unificado de Modelado*) para la realización de los diagramas.

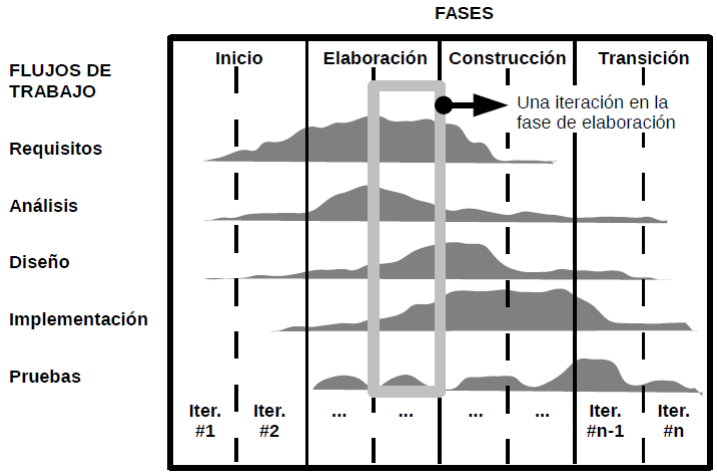


Figura 3.7: Fases, flujos de trabajo e iteraciones en el Proceso Unificado [13].

Al ser un proceso iterativo e incremental, se obtiene una notable reducción de riesgos en comparación con otros procesos de desarrollo. Además, favorece el incremento en eficiencia y permite refinar los requisitos de usuario iteración tras iteración.

Como se puede observar en la figura 3.7, el Proceso Unificado consta de cuatro fases en las cuales intervienen cinco flujos de trabajo por iteración. Las fases que constituyen este proceso son las siguientes:

- *Inicio*: En esta fase se obtienen a partir de los usuarios los primeros requisitos funcionales del sistema. Además, se hacen los primeros diseños, bocetos y diagramas de la arquitectura del sistema, junto con un coste y tiempo estimado para su desarrollo.
- *Elaboración*: En esta parte se especifican el resto de casos de uso y se termina de diseñar la arquitectura del sistema.
- *Construcción*: En esta fase se lleva a cabo la implementación del sistema siguiendo los requisitos recogidos y los diagramas y bocetos diseñados en las fases anteriores.
- *Transición*: En esta última fase se pone a prueba el producto resultante de la fase anterior, utilizando para ello un grupo reducido de usuarios, con lo que se busca que se

hagan pruebas y salgan a la luz posibles errores para poder así subsanar los mismos y poder entregar un producto sin errores ni defectos. También se pueden proponer mejoras del producto que serían implementadas en esta misma fase.

3.2.2 SCRUM

SCRUM es *un proceso de desarrollo en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente en equipo, y obtener así el mejor resultado posible de un proyecto* [27]. Al igual que el Proceso Unificado, SCRUM realiza entregas parciales y regulares del producto final, priorizando aquellas entregas que mayor beneficio aporten al receptor del proyecto.

Este proceso de desarrollo se recomienda para los proyectos que cumplen alguno de los siguientes requisitos:

- Proyectos en entornos complejos con necesidad temprana de resultados.
- Proyectos donde existe mucha incertidumbre.
- Proyectos donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.
- Proyectos en los que las entregas se alargan demasiado.
- Proyectos donde los costes se disparan.

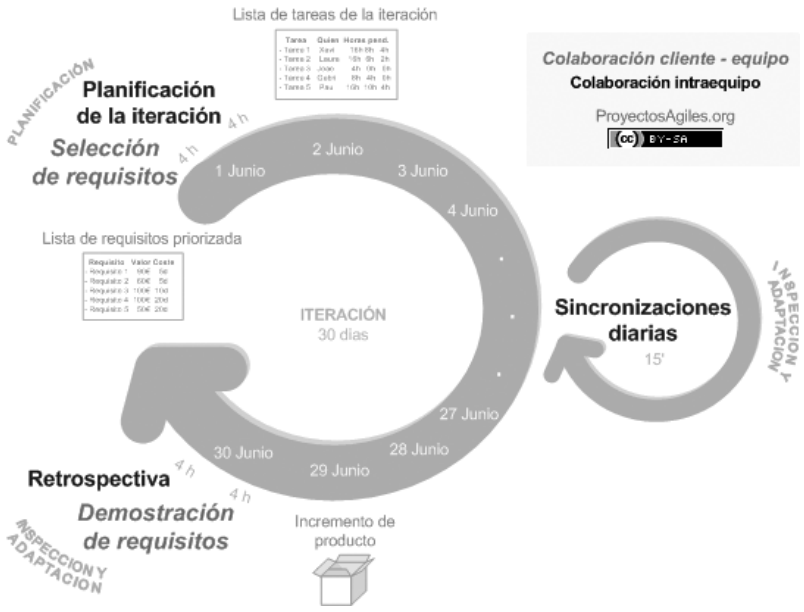


Figura 3.8: Diagrama del proceso de desarrollo SCRUM [27].

SCRUM se ejecuta en bloques temporales cortos y fijos (iteraciones). Cada uno de estos bloques temporales debe resultar en un incremento del producto final, ofreciendo un resultado completo que pueda ser entregado al cliente en caso de que éste lo necesite. Al igual que el Proceso Unificado, SCRUM parte de una serie de requisitos/objetivos que han de ser priorizados con el fin de generar un plan de proyecto. Para esta tarea se tiene muy en cuenta al cliente, siendo éste el que prioriza dichos requisitos/objetivos, repartiendo el trabajo en iteraciones y entregas.

Las actividades que se realizan en SCRUM por iteración son las siguientes [27]:

- **Planificación de la iteración**: En esta parte, el cliente selecciona una serie de requisitos, priorizando aquellos que él considere que aportan más valor con respecto al coste. El equipo pregunta dudas que puedan surgir acerca de los requisitos elegidos, siendo el cliente el encargado de responder a dichas dudas. Por último se planifica la iteración elaborando una serie de tareas necesarias para desarrollar los requisitos elegidos por el cliente, estimando esfuerzos y asignando tareas.

- **Ejecución de la iteración**: Alcanzado este punto, cada miembro del equipo inspecciona el trabajo que el resto está realizando con el fin de hacer las adaptaciones necesarias que permitan cumplir con los requisitos. Durante la iteración, el cliente junto al equipo refinan la lista de requisitos, cambiando y reformulando los objetivos del proyecto si hiciera falta.

- **Inspección y adaptación**: En esta última parte, se realiza una reunión de revisión donde se presenta el resultado conseguido por el equipo de desarrollo al cliente, el cual realiza las adaptaciones necesarias de manera objetiva replanificando el proyecto. Por último, el equipo se reúne y comenta la forma de trabajar que han seguido y los problemas que podrían impedirle progresar adecuadamente, todo con el fin de poder mejorar la productividad.

La principal diferencia con el Proceso Unificado es que éste es un proceso de desarrollo tradicional, mientras que SCRUM representa un proceso de desarrollo ágil donde se tiene muy en cuenta el cliente. El resto de diferencias entre ellos son varias (como ya se ha podido observar), aunque ambos utilizan iteraciones e incrementos para obtener un resultado final.

La metodología elegida para el presente Trabajo Fin de Grado ha sido el Proceso Unificado. Aún siendo más pesada que SCRUM, se ha elegido esta metodología debido a que los requisitos de entrada estaban bien definidos desde el principio, mientras que las metodologías ágiles funcionan mejor en proyectos donde existe mucha incertidumbre, es decir, donde los requisitos no se conocen con exactitud. Además no se contaba con la suficiente disponibilidad del cliente como para hacer que éste revisara constantemente los resultados, o replanificara el proyecto en función a los objetivos/requisitos completados.

3.2.3 Patrones de diseño

La parte que concierne al diseño de un sistema software es una tarea muy importante en el ciclo de vida del mismo. El problema se encuentra a la hora de buscar un procedimiento que permita crear el mejor diseño para dicho sistema software, ya que no existe procedimiento lo suficientemente claro o sistemático para esta tarea. Un diseño correcto debe permitir ser *escalable*, *extensible* y *crear componentes reutilizables* [10].

La solución a este problema se puede encontrar en la experiencia, ya que gracias a ésta se pueden evitar aquellas decisiones de diseño que llevaron a error y reutilizar aquellas que supusieron un acierto. En este punto es donde entran los **patrones de diseño**. Un patrón de diseño son formas bien conocidas y probadas de solucionar problemas de diseño que son recurrentes en el tiempo [9]. Estos patrones son ampliamente utilizados, y gracias a su uso se puede reducir el tiempo empleado en el diseño de sistemas software. Cada patrón de diseño cuenta con cuatro elementos esenciales [10]:

- Aunque no lo parezca, el **nombre** del patrón es algo muy importante, por eso éste no se suele elegir a la ligera cuando se crea uno nuevo. Esto se debe a que gracias a este nombre se ha de poder referir el problema que el patrón trata y la solución que éste aporta con una o dos palabras.
- El **problema de diseño** al que el patrón da solución.
- La **solución** en sí a dicho problema de diseño. En ella se describen los elementos, las relaciones, las responsabilidades y las colaboraciones que constituyen el diseño.
- Una serie de **consecuencias** que enumeran las ventajas y los inconvenientes del uso del patrón de diseño. Incluyen el impacto que éste tiene sobre la flexibilidad, la extensibilidad y la portabilidad del sistema.

Los patrones de diseño se pueden clasificar en función del ámbito de diseño donde tienen aplicación. Las categorías en las que se pueden clasificar son las siguientes [9]:

- **Patrones de creación**: se trata de aquellos que proporcionan una solución relacionada con la construcción de clases, objetos y otras estructuras de datos. Estos pueden ser: *Singleton*, *Abstract Factory*, *Factory Method* y *Prototype*.
- **Patrones estructurales**: este tipo de patrones versan sobre la forma de organizar las jerarquías de clases, las relaciones y las diferentes composiciones entre objetos para obtener un buen diseño en base a unos requisitos de entrada. Estos pueden ser: *Composite*, *Decorator*, *Facade*, *MVC*, *Adapter* y *Proxy*.

- **Patrones de comportamiento**: las soluciones que aportan este tipo de patrones están orientadas al envío de mensajes entre objetos y cómo organizar ejecuciones de diferentes métodos para conseguir realizar algún tipo de tarea de forma más conveniente. Estos pueden ser: *Observer*, *State*, *Iterator*, *Template Method*, *Strategy*, *Reactor* y *Visitor*.

Durante el diseño de *iCROM* se observaron ciertos problemas que eran fáciles de resolver aplicando patrones. Los patrones aplicados han sido los siguientes: *Abstract Factory*, *Facade* y *Observer*. En el **Capítulo 5** se habla más en detalle del uso de estos patrones en esta plataforma.

3.2.4 Programación Modular

Gracias a la programación modular se puede resolver un problema dividiendo éste en problemas más pequeños, resolviendo por partes cada uno de ellos (*divide y vencerás*). Estos subproblemas se conocen como módulos, los cuales deben contener el código fuente y las variables necesarias para llevar a cabo la función que les corresponde. De esta forma se puede llegar a reutilizar ciertos módulos en problemas que requieran de una función similar. Además, al dividir un problema mayor en problemas más pequeños, el mantenimiento y la detección de errores resultan más sencillos de llevar a cabo.

Las ventajas del uso de la programación modular en desarrollo de un sistema software son las siguientes [31]:

- Simplifica el diseño.
- Disminuye la complejidad de los algoritmos.
- Disminuye el tamaño total del programa.
- Ahorra tiempo de programación al promover la reusabilidad del código.
- Favorece el trabajo en equipo.
- Facilita el mantenimiento, la depuración y las pruebas.

Las aplicaciones que componen *iCROM* han requerido en su desarrollo la aplicación de la programación modular. El reproductor de la aplicación debía permitir añadir nueva funcionalidad sin requerir muchos cambios en el código ya creado. Al utilizar este tipo de programación para dicho reproductor se ha conseguido que la extensibilidad del mismo sea mayor, logrando además una fuerte cohesión, facilitando así la depuración y la detección de posibles errores. La aplicación web también ha seguido el mismo modelo de programación, ya que al tratarse de un sistema distribuido, se permitía así la posible reutilización de los componentes ya creados para futuras modificaciones.

3.3 ALMACENAMIENTO DE DATOS

Como plataforma integral para presentaciones, *iCROM* debe permitir la creación de una serie de archivos de secuencia que determinaran el orden de las diapositivas, teclas especiales, una configuración general, etc. Estos archivos normalmente van acompañados de una serie de archivos multimedia, los cuales están asociados a las diapositivas de la presentación. Además, la plataforma también requiere del almacenamiento persistente de los datos asociados a un cuestionario, lo que lleva a la necesidad de conocimiento acerca del uso y la gestión de una base de datos. Por este motivo, se han estudiado en detalle como funcionan el sistema de archivos y las bases de datos.

3.3.1 Sistema de archivos

Un sistema de archivos se representa como una estructura de directorios o carpetas organizadas en la cual se almacenan archivos de diferentes tipos y tamaños. Dicho sistema de archivos se encarga de la creación, modificación y borrado de los archivos y carpetas contenidas en dicha estructura. Otra de sus características permite fijar el formato físico de la información guardada en los medios de almacenamiento, ya sean internos (disco duro) o externos (memorias USB), gestionando así la memoria libre que pueda quedar disponible en estos.

Como se ha dicho anteriormente, el sistema de archivos se representa como una estructura de directorios. Esto es así porque el propio sistema de archivos gestiona los archivos contenidos en el sistema almacenándolos en directorios, los cuales siguen un orden jerárquico que parte del directorio raid, también conocido como "*root*".

Existen multitud de sistemas de archivos: **FAT** (*File Allocation Table*), **NTFS** (*New Technology File System*), **EXT** (*Extended File System*), **XFS**, **JFS** (*Journaling File System*), etc. Cada uno de estos aporta una serie de características que los distinguen del resto, algunos son mejoras de otros sistemas de archivos, otros están obsoletos y ya no se usan, etc.

La mayoría de estos sistemas de archivos suelen ir asociados a un sistema operativo en concreto. Por ejemplo NTFS con Windows, EXT con Linux y HFS+ con Mac OS. A continuación, se muestra una comparativa entre los sistemas de archivos utilizados por *Windows* (NTFS, FAT32, FAT) [21]:

- NTFS tiene la capacidad de recuperarse a partir de algunos errores relacionados con el disco duro automáticamente, mientras que FAT32 esto no puede hacer.
- NTFS tiene mejor compatibilidad para discos duros más grandes. En cambio FAT32 tiene limitaciones de tamaño.

- NTFS aporta mayor seguridad debido a que puede utilizar permisos y cifrado para restringir el acceso a archivos específicos para usuarios aprobados. FAT32 no aporta seguridad alguna.

Los sistemas de archivos pueden gestionar diferentes tipos de ficheros. Un fichero o archivo no deja de ser una colección de bytes que es gestionada y almacenada en bloques en un directorio por el sistema de archivos, determinando que bytes corresponden con el archivo y cuales no. Los archivos que pueden ser almacenados y gestionados por un sistema de archivo se pueden clasificar en:

- **Archivos de texto**: Este tipo de archivos están formados únicamente por caracteres. Estos archivos pueden ser entendibles mediante el uso de editores o visualizadores de texto. Existen multitud de formatos de archivos de texto, los cuales varían en función de la codificación que estos sigan.
- **Archivos multimedia**: Este tipo de archivo almacena información multimedia, como puede ser la información asociada a una imagen, un vídeo o una canción. A diferencia de los archivos de texto, este tipo de archivo no suele ser entendible por humanos haciendo uso de editores de texto, necesitando en este caso el uso de otro tipo de herramientas para que estos datos sean interpretados correctamente y puedan mostrarse o reproducirse. Otra diferencia con los archivos de texto es que este tipo de archivos suelen seguir un algoritmo de compresión para reducir su tamaño en el disco.
- **Otros archivos**: A modo de cajón desastre, el resto de archivos se suelen catalogar de esta forma. Estos archivos suelen ser archivos asociados al sistema operativo, como librerías, ejecutables, etc.

Aunque existen sistemas de archivo específicos para el trabajo con multimedia, *iCROM* funciona en cualquier tipo de partición soportada por el kernel Linux. En el desarrollo se ha empleado EXT4 (última versión del sistema de archivos disponible para *GNU/Linux*).

3.3.2 Bases de datos

Una base de datos consiste en una *colección de datos interrelacionados y un conjunto de programas para acceder a dichos de datos. En otras palabras, una base de datos no es más que un conjunto de información (un conjunto de datos) relacionada que se encuentra agrupada o estructurada* [20].

Una base de datos tiene la capacidad de dar acceso a los datos que contiene de forma simultánea a una serie de instancias. Estos datos son conocidos como *metadatos*, debido a que permiten a las aplicaciones de bases de datos abstraerse al proporcionar una definición

externa de como almacenar los datos internamente (*abstracción de los datos*). Los datos se almacenan en entidades, las cuales constan de una serie de atributos que catalogan los datos que contienen de una determinada forma y permiten las relaciones entre otras entidades de la misma base de datos.

Las bases de datos se gobiernan gracias a un *sistema de administración de bases de datos* (SGBD), los cuales proporcionan una forma de almacenar y recuperar la información de manera práctica y eficiente [20]. Las características que definen un SGDB son las siguientes:

- *Integración de toda la información de la organización*, lo que permite evitar redundancias e inconsistencias en los datos almacenados por la base de datos.
- *Persistencia de datos*.
- *Accesibilidad simultánea para distintos usuarios*.
- *Independencia de los programas respecto a la representación física de los datos*, gracias al empleo de metadatos.
- *Definición de vistas parciales de los datos para distintos usuarios*, capacitando así a generar roles entre los usuarios (administradores, usuarios estándar, etc).
- *Mecanismos para controlar la integridad y la seguridad de los datos*.

De las características anteriores se puede deducir que los SGBD aseguran la independencia, la integridad y la seguridad de los datos.

Para el presente proyecto, se han tenido en cuenta los siguientes gestores de bases de datos:

- **MySQL**[7]: Gestor de bases de datos relacionales, desarrollado por Oracle bajo la licencia dual GPL/Licencia comercial. Entre sus características destacan las siguientes:
  - Multiplataforma (Windows, Linux, Mac Os).
  - Permite la consulta simultánea de diferentes usuarios al mismo tiempo.
  - Soporta gran cantidad de tipos datos para las columnas.
  - Gran portablidad entre sistemas.
  - Permite añadir seguridad a las bases de datos.
  - Gran rapidez, considerada como el gestor de bases más rápido de Internet. Además es fácil de usar.
  - Fácil de instalar y configurar.
- **SQLite**[8]: Gestor de bases de datos relacionales, desarrollado por D. Richard Hipp y

[7] https://www.mysql.com/
[8] https://www.sqlite.org/

distribuido bajo una licencia de dominio público. Entre sus características se pueden encontrar las siguientes:

- Multiplataforma (Windows, Linux, Mac Os).
- La base de datos que gestiona es un fichero, lo que lo caracteriza de ser muy portable al poder mover el fichero con la base de datos a cualquier parte.
- No necesita de configuración extra para acceder a los ficheros con las bases de datos.
- No requiere de una infraestructura pesada de cliente-servidor, a diferencia de MySQL.
- Permite transacciones ACID (Atómicas, Consistentes, Aisladas y Durables).
- Permite tamaños de fichero de hasta 2GB.
- Muy recomendable para la creación de bases de datos temporales o para aplicaciones web.

La aplicación web de preguntas de *iCROM* ha requerido del uso de una base de datos para almacenar los datos asociados a un determinado cuestionario. Aunque MySQL proporciona más funcionalidad que SQLite, se ha elegido SQLite en su última versión (SQLite3) para realizar esta tarea por su facilidad de uso y de despliegue, ya que no requiere de instalación alguna. Además se ha tenido en cuenta que trabajaba con archivos que pueden moverse entre computadores, lo que permite guardar el fichero de datos con la base de datos que contienen los resultados de un determinado cuestionario para futuros usos o revisiones.

3.4 HERRAMIENTAS DE UTILIDAD

El reproductor de diapositivas de la plataforma requiere de una serie de herramientas auxiliares que le permitan evitar tener que implementar cierta funcionalidad asociada con el análisis y la generación de archivos PDF, así como la generación y modificación de imágenes de forma dinámica. Para evitar tener que implementar esta funcionalidad, se ha hecho un estudio sobre las herramientas actualmente disponibles y de uso libre que ya tuvieran implementada dicha funcionalidad y permitieran su uso desde programas ajenos a dichas herramientas. A continuación, se hablará de cada una de estas herramientas, junto a una breve descripción de la funcionalidad aportada al reproductor.

3.4.1 Herramientas de utilidad para el tratado de archivos PDF

El formato de documento portátil (PDF) *es un formato de archivo utilizado para presentar e intercambiar documentos de forma fiable, independiente del software, el hardware o el*

sistema operativo [2]. Inventado por Adobe[9], PDF representa un estándar abierto y oficial reconocido por la Organización Internacional para la Estandarización (ISO[10]).

Los archivos PDF están muy extendidos y, aunque en un principio su objetivo era servir de documentos destinados a la impresión, actualmente se utilizan para casi cualquier tarea, incluso para presentar diapositivas utilizando algún visualizador de PDF. La mayoría de los programas actuales de generación de presentaciones multimedia, generación de documentación e incluso programas de edición y generación de imagen, permiten exportar el resultado generado con dichas herramientas a formato PDF. Esto convierte a este tipo de archivos en un gran candidato para servir de entrada a programas que trabajen con imágenes, permitan la realización de presentaciones multimedia, etc.

Por este motivo se ha estudiado su uso como entrada en la plataforma, lo que ha llevado a una búsqueda de herramientas que añadieran la funcionalidad de creación y modificación de este tipo de documento desde las propias aplicaciones de la plataforma. La búsqueda ha dado como resultado las siguientes herramientas:

- **GNU GhostScript**[11]: Escrito por Peter Deutsch y distribuido bajo una licencia GNU, GhostScript representa un intérprete que permite tratar documentos con formato PS y PDF. Entre sus características se pueden encontrar la impresión de archivos PDF, la obtención de las imágenes que componen un PDF o mostrar este tipo de archivos por pantalla.

- **PDFTK**[12]: Conjunto de herramientas de código libre que permiten dividir, combinar, cifrar/descifrar, comprimir/descompirmir, así como reparar documentos PDF. Aunque principalmente se utiliza mediante comandos en consola, también consta de una serie de interfaces gráficas que permiten su uso de forma más simple e intuitiva.

Aunque ambas permiten tratar este tipo de archivos, no se ha decidido por usar una en exclusividad. Ambas se emplearán en las tareas de creación y generación de este tipo de archivos, utilizando la herramienta más adecuada para cada tarea. En el caso de descomprimir un archivo PDF en las imágenes que lo componen, se deberá usar *GhostScript*, y para la tarea de creación del archivo PDF de la presentación se utilizará PDFTK, ya que permite reunir una serie de imágenes y distribuirlas siguiendo una determinada plantilla generando como resultado un archivo PDF.

[9] http://www.adobe.com/es/
[10] http://www.iso.org/iso/home.html
[11] https://www.gnu.org/software/ghostscript/
[12] https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/

3.4.2 Herramientas de utilidad para la creación y edición de imágenes

Las imágenes representan la base sobre la que se sustenta el reproductor de la plataforma *iCROM*. Por este motivo, era necesaria una herramienta que otorgara cierta funcionalidad para la creación y edición de imágenes temporales durante la ejecución del reproductor. Estas imágenes serían necesarias para la creación de miniaturas, para redimensionar las imágenes obtenidas de los archivos PDF de entrada, ordenar las imágenes de los PDF de salida, etc.

Para esta tarea se ha tenido claro desde el principio qué herramienta auxiliar se va a utilizar debido a que se conocía previamente la funcionalidad que ésta puede aportar al reproductor de la plataforma. Además se sabe que gozaba de no tener un competidor capaz de ofrecer las funcionalidades que ésta ofrece.

Esta herramienta es **Imagemagick**[13], una herramienta que proporciona un conjunto de funcionalidades para la creación, manipulación y conversión de imágenes, capaz de leer hasta 100 formatos distintos, obtener imágenes de vídeos o incluso de archivos PDF. Imagemagick está programada en C, sigue una licencia *Apache 2.0* y es multiplataforma, lo que la hace una herramienta idónea para el reproductor de la plataforma.

3.5 NETWORKING

iCROM debe permitir desplegar una aplicación web para realizar, en tiempo real, cuestionarios al público de una presentación. Para ello es necesario un servidor que permita alojar la aplicación web (de forma local o remotamente), así como una infraestructura de comunicación que permita controlar y sincronizar las partes que la componen.

Para esto ha sido necesario el estudio de los servidores web de uso libre disponibles hasta la fecha, junto con un mecanismo para la comunicación y sincronización entre las partes de la aplicación, estudiando para esto los *sockets* y el *protocolo HTTP*.

3.5.1 Servidores Web

Un servidor web es un software que permite la publicación de contenidos y servicios a través de Internet. Principalmente alojan sitios Web, permaneciendo a la espera de peticiones de los clientes interesados en dichos sitios, los cuales son atendidos mediante un protocolo de comunicación, siendo el más corriente el *Protocolo de Transferencia de Hipertexto* (HTTP).

Se han estudiado una serie de servidores web para el alojamiento de la aplicación web de la plataforma. Estos debían de cumplir una restricción, y era que debían distribuirse bajo licencia de uso libre. Los servidores web que se han encontrado son los siguientes:

[13] http://www.imagemagick.org/script/index.php

- **Apache**[14]: Representa el servidor web más popular de Internet. Es multiplataforma y sigue una licencia Apache 2.0. Al ser un servidor web modular, permite extender el mismo con nuevos módulos que aporten nueva funcionalidad. Además, al ser tan usado, es fácil encontrar ayuda y soporte en caso de posibles problemas en su uso.
- **Lighttpd**[15]: Este servidor web ha sido diseñado para ser rápido, flexible y adaptable a los estándares. Lighttpd también es multiplataforma y se distribuye bajo una licencia de software libre, siendo en este caso la licencia BSD. Permite alojar varios dominios bajo la misma IP, principalmente da soporte para PHP, Ruby y Python, permite redirecciones HTTP y reescrituras de URL, etc. Por último, si hay que tener una desventaja de su uso en cuenta, ésta podría ser su consumo constante de memoria.
- **Nginx**[16]: Este servidor web se caracteriza por ser más ligero que los anteriores. Al igual que el anterior, Nginx es multiplataforma y sigue una licencia BSD, aunque también tiene una versión comercial. Se caracteriza por ser un servidor proxy inverso con opciones de caché, que permite el balanceo de carga y por ser tolerante a fallos.

De las tres opciones anteriores se ha elegido **Apache** como servidor web para la aplicación web de la plataforma, por su estructura modular y gozar de un gran soporte por parte de la comunidad en Internet.

3.5.2 Sockets

Un socket se puede definir como una tupla IP-puerto que permite la comunicación entre dos procesos normalmente alojados en distintas máquinas, permitiendo así que éstas se puedan comunicar en red. Para que la comunicación entre dos sockets funcione, uno debe hacer las veces de servidor, escuchando las posibles peticiones que le puedan llegar de otro proceso local o de otra máquina externa. El otro socket participante en la comunicación envía datos al primer socket con un fin determinado, por lo que este hace las veces de cliente. De esto se puede deducir que cuando se utilizan sockets, indirectamente se está utilizando una arquitectura cliente-servidor.

La conexión que se establece entre dos sockets sigue un determinado protocolo, el cual puede ser *TCP* o *UDP*. Dependiendo del protocolo, se desprenden una serie de características:

- El protocolo **TCP** está orientado a conexión, garantizando así la correcta transmisión de la información que se envíe durante la comunicación entre los sockets. Esto es gracias a que mantiene un orden entre los paquetes de datos que son enviados, consiguiendo así un control total sobre los paquetes que puedan extraviarse durante la comunicación.

[14] https://httpd.apache.org/
[15] https://www.lighttpd.net/
[16] https://www.nginx.com/resources/wiki/

- El protocolo **UDP** permite que los paquetes de datos viajen en cualquier orden. Al contrario que TCP, no está orientado a conexión, por lo no garantiza que lleguen todos los paquetes de datos al socket receptor.

La aplicación de control para la aplicación web de preguntas de la plataforma requiere de un mecanismo de comunicación que le permita recibir los datos emitidos por los mandos auxiliares que se puedan utilizar durante un concurso. Para esto se han empleado sockets y el protocolo TCP, debido a que los mensajes emitidos por los mandos deben llegar a la aplicación de control sin la probabilidad de que éstos se pierdan durante la comunicación.

3.5.3 Protocolo HTTP

La comunicación con un Servidor Web mediante el protocolo HTTP se realiza en dos etapas. La primera de ellas consiste en el envío de una solicitud HTTP al servidor, la cual es procesada y respondida en una segunda etapa de la comunicación por el mismo servidor.

Una *Solicitud HTTP* es un conjunto de líneas que el cliente envía al servidor. Estas líneas incluyen [7]:

- **Una línea de solicitud** que especifica el documento solicitado, junto con el método y la versión del protocolo que se utilizarán.
- **Los campos del encabezado de solicitud**, los cuales vienen a ser un conjunto de líneas opcionales que aportan información adicional a la solicitud.
- **El cuerpo de la solicitud**, el cual se representa como otro conjunto de líneas que deben ser separadas de las líneas que le preceden mediante un espacio en blanco. Esto permite el envío de datos por comando POST.

Una vez el servidor procesa la solicitud y tiene los datos necesarios para responder a la misma, éste prepara una **Respuesta HTTP**. Al igual que la solicitud, una respuesta HTTP se puede definir como un conjunto de líneas que son enviadas desde el servidor al cliente. Estas líneas incluyen [7]:

- **Una línea de estado** que especifica la versión del protocolo utilizada, el código del estado y una explicación del mismo.
- **Los campos del encabezado de respuesta**, los cuales se representan como un conjunto de líneas opcionales que permiten aportar información a la respuesta, de forma similar a *los campos del encabezado de la solicitud*.
- **El cuerpo de la respuesta**, el cual contiene los datos solicitados por el cliente.

La aplicación de control precisa de una forma de comunicarse con la aplicación web de preguntas de la plataforma sin la necesidad de que ésta utilice sockets o algún otro mecanismo similar de comunicación. Esta aplicación web debe de servir de puente entre los usuarios y la aplicación de control, por lo que debe de ser una aplicación sin estado. A modo de marioneta, la aplicación web debe ser gestionada por la aplicación de control con el fin de mostrar los datos actuales del cuestionario y enviar nuevos eventos y respuestas de los usuarios. Debido a que no se pueden utilizar sockets para establecer esta comunicación, se ha estudiado la manera de hacer esto con solicitudes HTTP, las cuales devolvieran unos datos u otros dependiendo del contenido de la solicitud.

3.6 TECNOLOGÍAS AUXILIARES

El desarrollo del reproductor de *iCROM* ha requerido de un conocimiento variado sobre ciertas tecnologías con el fin de proveer de la funcionalidad suficiente para conseguir alcanzar aquellos objetivos propuestos en un principio para dicho reproductor. Durante esta búsqueda de tecnología se han estudiado temas relacionados con la reproducción de contenido multimedia, gráficos 3D y funcionalidad auxiliar que permita extender la funcionalidad básica ofrecida por las herramientas utilizadas hasta el momento. En esta última sección se habla del estudio realizado, de las tecnologías encontradas, así como de las elecciones tomadas de cara al desarrollo del reproductor de la plataforma.

3.6.1 Entorno del reproductor

Para poder mostrar presentaciones por pantalla, el reproductor necesitaba de una ventana de aplicación, de una forma de capturar los eventos locales, así como de otros mecanismos relacionados con el contexto de la aplicación. Para esto fue necesaria la búsqueda de herramientas que permitieran montar la base del reproductor, aquello sobre lo que se pudieran representar objetos a modo de diapositivas. Durante la búsqueda se han encontrado dos herramientas:

- **SDL**[17]: *Simple DirectMedia Layer* es un conjunto de bibliotecas desarrolladas en C que proporciona funcionalidad básica para el dibujado en 2D, gestión de ventanas, detección de eventos locales y hasta reproducción de sonido. Es una biblioteca de código libre y multiplataforma. En su versión 2.0, se añade más funcionalidad y compatibilidad para C++.

[17] https://www.libsdl.org

- **Qt**[18]: Al igual que SDL, Qt es una biblioteca multiplataforma que proporciona una serie de funcionalidades en lo referente a la creación de la ventana de la aplicación, gestión de eventos de entrada, etc. A diferencia de SDL, Qt está programado en C++ y es muy utilizado para la creación de interfaces de usuario.

Debido a que estas tecnologías se han quedado cortas a la hora de cumplir ciertos objetivos del reproductor, ha sido necesaria la búsqueda de otras herramientas que junto a esta tecnología permitieran la gestión integral de los gráficos 3D mostrados por pantalla. Siguiendo esta idea, se han tenido en cuenta *OpenGL*[19] y *DirectX*[20] como opciones, ambas bibliotecas creadas para el despliegue de gráficos 3D por computador.

- **OpenGL**: Biblioteca estándar de uso libre para la representación de gráficos 3D por computador. Ofrece funcionalidad para representar tanto objetos 2D como 3D.

- **DirectX**: Biblioteca privativa creada por *Microsoft* para la representación de gráficos 3D en Windows. Permite aprovechar la aceleración hardware si ésta está disponible y, a diferencia de OpenGL, no es multiplataforma.

Para el reproductor de *iCROM* se ha elegido la combinación de SDL2.0 y OpenGL por dos razones. La primera de estas razones es la alta compatibilidad entre las tecnologías. La segunda, determinada por un objetivo de la plataforma, es por que tanto SDL como OpenGL son distribuidas bajo una licencia de software libre y multiplataforma.

3.6.2 Reproducción de archivos multimedia

Un archivo multimedia, como se ha dicho en la sección de los *Sistemas de Archivos*, es un tipo de archivo que almacena información multimedia, como puede ser información asociada a una imagen, un vídeo o una canción. Además suelen seguir un algoritmo de compresión para reducir su tamaño en el disco.

En una presentación multimedia es corriente el uso de este tipo de archivos a la hora de mostrar un vídeo, audio o efecto de sonido. El reproductor de la plataforma, por tanto, necesitaba de un mecanismo que permitiera descomprimir este tipo de archivos y reproducirlos durante una presentación siguiendo la ejecución normal de la aplicación. Para esto ha hecho falta un estudio en el campo de las librerías actuales para la descompresión y reproducción de archivos multimedia.

[18] https://www.qt.io
[19] https://www.opengl.org
[20] https://www.microsoft.com/es-es/download/search.aspx?q=directx

La descompresión de vídeos y audio para su posterior interpretación y reproducción es una tarea sumamente compleja debido a que requiere de un gran conocimiento sobre la estructura de este tipo de archivo, sus formatos, etc. Además, también requiere de amplios conocimientos de hilos y concurrencia, debido a que esta tarea de descompresión se debe realizar en un hilo separado del hilo principal de ejecución con el fin de no poner trabas a la hora de representar una presentación, todo esto a la vez que se reproduce el audio o vídeo que se esté descomprimiendo.

La búsqueda ha dado como resultados dos herramientas:

- **FFMPEG**[21]: Software libre y multiplataforma, distribuido bajo una licencia *GNU Lesser General Public License 2.1+* o *GNU General Public License 2+* (dependiendo de que bibliotecas se incluyan). FFMPEG permite la comprimir, descomprimir, grabar, transcodificar o hacer streaming de vídeo y audio.
- **GStreamer**[22]: Framework multiplataforma y distribuido bajo una licencia *GNU Lesser General Public License*. Permite crear aplicaciones audiovisuales, tanto de vídeo como de audio haciendo uso de la biblioteca GObject. Está escrito en C y permite abstraer a los desarrolladores de la tarea de descomprimir/comprimir archivos de audio o vídeo mediante el uso de tuberías de reproducción, algunas de las cuales ya creadas por el propio GStreamer. Estas tuberías se construyen mediante la combinación de distintos módulos proporcionados por el framework, cada uno de los cuales realiza una tarea en cuestión ante una entrada produciendo una determinada salida que puede ser final o entrada a otro módulo.

El uso de FFMPEG para esta tarea obliga a crear desde cero los reproductores de audio y vídeo, tarea realmente compleja que de ser llevada a cabo haría que la implementación del reproductor de la plataforma se alargara demasiado y no se pudiera entregar el presente Trabajo Fin de Grado a tiempo. Por este motivo se ha acabado eligiendo Gstreamer como herramienta para esta tarea, ya que permite abstraer al desarrollador de la complejidad subyacente mediante el uso de sus tuberías de reproducción.

3.6.3 Lenguajes de marcado y de serialización

Para el diseño y creación de presentaciones multimedia, *iCROM* requiere de un lenguaje propio basado en algún lenguaje de marcado o de serialización ya existente, que le permita crear y analizar archivos propios de secuencia o de configuración. A partir de la interpretación de estos archivos, *iCROM* debe ser capaz de mostrar una presentación junto con la configuración establecida por el usuario para la misma.

[21] https://ffmpeg.org
[22] https://gstreamer.freedesktop.org

Para esta tarea, se ha realizado un estudio sobre pequeños lenguajes de marcado y serialización que permitieran a *iCROM* evitar tener que desarrollar un lenguaje propio desde cero. El estudio ha dado como resultado los siguientes lenguajes:

- **YAML**[23]: Este es un lenguaje de serialización reciente, diseñado para el intercambio de información. Su abreviación proviene de *Yet Another Markup Language*. Es rápido y fácil de usar, además, facilita tareas de mapeo de estructuras de datos complejas. Estas características lo convierten en un magnífico candidato a la hora de crear archivos de configuración, realizar tareas de depuración o para otros fines similares.

- **JSON**[24]: Este lenguaje permite codificar información en pares *atributo*:*valor*. Su abreviación proviene de *JavaScript Object Notation*. Es ampliamente utilizado en servicios web y APIs REST debido a su simplicidad y facilidad de uso. Uno de los problemas del uso de JSON es que no siempre soporta *Unicode* debido a que dependiendo de la implementación, éste puede sufrir ciertas variaciones. A veces, su uso también puede suponer un riesgo de seguridad si se utiliza para el intercambio de información entre procesos debido a su facilidad de evaluación y obtención de los datos que contiene. Aun así, este lenguaje se utiliza bastante con lenguajes como JavaScript o Python, y al usarse en la mayoría de los navegadores modernos, PHP también permite hacer buen uso de él.

- **XML**[25]: Lenguaje de marcado cuya abreviatura proviene de *Extensible Markup Language*. Este lenguaje fue diseñado por el *World Wide Web Consortium* (W3C), y permite definir un conjunto de reglas para codificar información de manera que sea legible por un ser humano o por un programa de ordenador. XML está muy extendido en servicios web y para archivos de configuración debido a que puede ser utilizado para representar cualquier estructura de datos. Una de las características principales de este lenguaje de marcas es su suporte para *Unicode*, lo que permite escribir los datos en cualquier idioma. Sin embargo, es muy criticado por su verbosidad y complejidad, además, por requerir gran cantidad de memoria y procesador con documentos muy grandes. Aun así, este lenguaje se utiliza mucho hoy en día, siendo un claro ejemplo de ello, PowerPoint. Esta herramienta ofimática utiliza este lenguaje de marcado para la definición de una variante de sus propios archivos de presentación (.pptx).

Para los archivos de secuencia y configuración del reproductor de la plataforma se ha decidido utilizar *YAML* por la potencia y rapidez que este aporta. Por otro lado, para la serialización de los mensajes empleados durante la comunicación entre las partes de la

[23] http://yaml.org
[24] http://www.json.org
[25] http://www.xml.com

aplicación web de preguntas se ha elegido JSON por poseer gran soporte en lo que se refiere a servicios web.

3.6.4 Bibliotecas auxiliares

Para acabar con los antecedentes de *iCROM*, en este apartado se habla de aquellas bibliotecas que se han tenido en consideración para proveer de cierta funcionalidad auxiliar al reproductor en lo referente a mecanismos de búsqueda, creación y navegación de directorios, modificación de mapa de píxeles y generación de imágenes a partir de éstos, obtención de información de las ventanas de los programas ejecutados en segundo plano, etc.

- **Boost**[26]: Conjunto de bibliotecas de código libre y de propósito general que complementan e interaccionan con la librería estándar de C++ [15]. Con un total en torno a las 120 bibliotecas, Boost proporciona funcionalidad para diferentes categorías, pudiendo ser la concurrencia, las estructuras de datos, el procesamiento de texto o la programación de sistemas unos ejemplos de las mismas.

  Para cubrir ciertas necesidades que se desprendían del uso de C++ en el desarrollo de la plataforma, se ha utilizado Boost debido a que éste da soporte para todas necesidades que puedan surgir durante el desarrollo (comparación de cadenas de texto, navegación y creación de directorios, etc).

- **GDK**[27]: Esta biblioteca es una biblioteca multiplataforma y de código libre, distribuida bajo una licencia *GNU LGPL*. Representa un conjunto de funciones para el tratado de gráficos a bajo y alto nivel.

  A la hora de tratar los mapas de píxeles obtenidos de las tuberías de vídeo y de las tuberías de captura, se requería de una funcionalidad que permitiera tratar ese tipo de datos, obtener dimensiones, etc. Para esta tarea se ha elegido GDK por proveer gran funcionalidad en este aspecto y por ser de código libre y por ser multiplataforma.

- **XDO**[28]: Ésta es una biblioteca poco conocida que permite obtener una gran cantidad de datos referentes a la ventana de una aplicación en base a su nombre (total o parcial) o mediante su identificador de ventana. Además, permite simular pulsaciones de ratón y eventos de teclado en aplicaciones ejecutadas en segundo plano.

  A la hora de capturar la ventana de una aplicación era ha sido necesario conocer ciertos parámetros de la misma, por lo que se decidió utilizar esta biblioteca para ello. Además, al permitir simular eventos en dichas aplicaciones, también provee de funcionalidad a la hora de redirigir eventos de teclado a las aplicaciones capturadas.

[26] http://www.boost.org
[27] https://developer.gnome.org/gdk3/stable
[28] http://www.semicomplete.com/files/xdotool/docs/html

CAPÍTULO 4

MÉTODO DE TRABAJO

En este capítulo se indica la metodología empleada durante el desarrollo de la plataforma *iCROM*. Además, se habla de las diferentes herramientas utilizadas junto con su versión de compilación. En relación a la metodología de trabajo, en el *Capítulo 6* se pueden encontrar las iteraciones por la que ha pasado el desarrollo, junto a la complejidad y resultados de cada una de ellas.

4.1 METODOLOGÍA

El presente Trabajo Fin de Grado se ha llevado a cabo siguiendo una **metodología iterativa e incremental**. Ésta en concreto ha sido el **Proceso Unificado de Desarrollo**. Para más información sobre esta metodología y del por qué de su elección para su uso en el presente Trabajo Fin de grado, se recomienda leer la sección de Ingeniería del Software (*Capítulo 3*). Gracias al uso de esta metodología, el desarrollo de la plataforma ha contado con las siguientes características:

- Se ha dirigido por casos de uso, estudiando las funcionalidades requeridas por los diferentes usuarios del sistema.
- Se ha centrado en la arquitectura, dirigiéndose los primeros esfuerzos hacia el desarrollo de la base de la plataforma, añadiendo más y más funcionalidad según avanzaba el desarrollo.
- El desarrollo ha sido iterativo e incremental, implementándose las diferentes partes del sistema en sucesivas iteraciones.
- Se han seguido las cuatro fases del *Proceso Unificado de Desarrollo* y el trabajo ha sido repartido entre los cinco flujos de trabajo fundamentales en los que se basa esta metodología.

4.2 SOFTWARE

En esta sección se enumeran el sistema operativo, los lenguajes de programación, las bibliotecas y las herramientas software utilizadas durante el desarrollo, indicando (entre otras cosas) la versión de compilación utilizada para cada una de ellas.

4.2.1 Sistema Operativo

Los sistemas operativos utilizados para el desarrollo e implementación de este proyecto han sido varios con el fin de probar el funcionamiento de la plataforma en diferentes sistemas y distribuciones. Principalmente se ha utilizado **Linux** para esta tarea, utilizando **Debian** (Jessie), **Ubuntu** (Vivid Vervet), **Xubuntu** (Xenial Xerus) y **ElementaryOS** (Freya) como distribuciones en su versión para computadores de *64 bits*.

4.2.2 Lenguajes de programación

Para la implementación, se han empleado diferentes lenguajes de programación, los cuales varían en función del propósito de su uso. Estos lenguajes son:

- **C++**: Es el lenguaje utilizado por el reproductor de la plataforma para su implementación, así como para la parte lógica y gráfica del mismo. Se ha utilizado el estándar C++11 con el fin de usar las nuevas características que éste ofrece.
- **PHP** (v5.6): Es el lenguaje utilizado para la implementación de la lógica de la aplicación web de preguntas de la plataforma.
- **Python** (v3.4): Se ha utilizado para la implementación de la aplicación de control de la aplicación web de preguntas.
- **JavaScript**: Se ha utilizado para la consulta periódica de la aplicación web de preguntas.
- **AJAX**: Se ha usado, junto a *JavaScript*, para actualizar la interfaz de la aplicación web de preguntas sin tener que recargar la página completa.
- **HTML5**: Este lenguaje de marcas se ha empleado en la aplicación web de preguntas de la plataforma para crear la interfaz de la misma.
- **CSS3**: Es el lenguaje, junto con *HTML5*, con el que se ha especificado la apariencia de la aplicación web de preguntas de la plataforma.

4.2.3 Bibliotecas

Las bibliotecas que se han utilizado para el desarrollo de esta plataforma son muy variadas. Éstas van desde librerías gráficas hasta librerías para decodificación de audio y vídeo. A continuación, se listarán las librerías que han tomado un mayor papel en el desarrollo de la plataforma, así como una breve descripción de su uso en la misma:

- **OpenGL** (v2.0): Es la biblioteca gráfica utilizada por la plataforma para el dibujado de objetos en pantalla.
- **SDL** (v2.0.2): Es la biblioteca utilizada por la plataforma para la carga de imágenes y la creación del contexto del reproductor (ventana, eventos, etc.).
- **YAML-CPP** (v0.5): Es la biblioteca utilizada para el análisis de los archivos propios del reproductor (archivos de secuncia y configuración).
- **Boost** (v1.55): Es una biblioteca que extiende las capacidades del lenguaje *C++*. Se utiliza en la plataforma para varias tareas, como por ejemplo para crear archivos y navegar por directorios en el sistema.
- **STL**: Es una biblioteca de C++ que ofrece multitud de estructuras de datos y algoritmos. En la plataforma se utiliza en casi cualquier clase. Un ejemplo de su uso podría ser la creación de vectores con los datos referentes a una presentación (Diapositivas, Teclas especiales, etc.).
- **GStreamer** (v1.6.3): Es un *framework* que se ha utilizado para la decodificación y reproducción de contenidos multimedia (vídeo y audio).
- **JSON**: Biblioteca utilizada para la codificación y decodificación de los datos enviados entre las partes que componen la aplicación web de la plataforma.
- **Request**: Biblioteca utilizada en el backend de la aplicación web de preguntas para el envío de solicitudes HTTP.
- **SQLite** (v3.0): Lenguaje utilizado para crear la base de datos de la aplicación web de preguntas de la plataforma, así como para realizar consultas sobre la misma.

4.2.4 Herramientas de desarrollo

La totalidad del código de la plataforma se ha escrito utilizando un editor de software libre. Además, ha sido compilado y depurado usando herramientas de GNU. A continuación, se listan cada una de las herramientas de desarrollo utilizadas:

- **Atom** (v1.8): Éste es un editor de código libre que permite la edición de código y texto plano. Tiene soporte para OS X, Linux y Windows. Utiliza un sistema de plugins escritos en *Node.js* que permiten añadir funcionalidad a la herramienta. Además, tiene embebido *git* para el control de versiones.

- **GNU Make** (v3.81): Ésta es una herramienta que facilita la compilación de archivos fuente para la obtención de ejecutables. Puede lanzar varios hilos simultáneamente, acelerando la compilación al aprovechar los núcleos disponibles en el equipo. Se ha utilizado para compilar los archivos fuente del reproductor de la aplicación.

- **GDB** (v7.7.1): Es el depurador de los sistemas GNU por defecto. Se ha empleado en ocasiones para depurar el código C++ asociado al reproductor de la plataforma.

- **Git** (v1.9.1): El control de versiones es una herramienta muy útil para la creación de proyectos con cierto nivel de complejidad. Por esta razón se utilizado Git para llevar el control de versiones del proyecto.

- **GCC** (v4.8.2): Como compilador en GNU/Linux se ha utilizado GCC por ser libre y específicamente desarrollado en sus orígenes para los sistemas GNU. También se tiene en cuenta que da un gran soporte a lenguajes como C, C++, etc.

- **ShareLaTeX**: Herramienta web utilizada para la creación y compilación del código LaTeX asociado a la documentación del actual Trabajo Fin de Grado.

4.2.5 Herramientas de edición gráfica

Se han utilizado diferentes herramientas de edición gráfica para la creación de las imágenes, diagramas de clase, diagramas UML, etc. Éstas han sido las siguientes:

- **Blender**[1] (v2.77): Programa de software libre para la creación y modelado de objetos tridimensionales.

- **Gimp**[2] (v2.8.16): Herramienta de edición gráfica de código abierto utilizada para la creación y edición de imágenes.

- **Draw.io**[3]: Herramienta web utilizada para la creación imágenes 2D, diagramas UML, digramas de clases, etc.

[1] www.blender.org
[2] www.gimp.org
[3] www.draw.io

4.2.6 Herramientas auxiliares

Para poder ofrecer ciertas funcionalidades en el proyecto, se han utilizado un conjunto de herramientas auxiliares, libres y multiplataforma, ya implementadas por terceros. Éstas han sido las siguientes:

- **Imagemagick** (v7.0.2): Representa un conjunto de herramientas para la creación, edición, composición y conversión de imágenes. Se ha utilizado en la plataforma, entre otras cosas, para redimensionar diapositivas y generar miniaturas.
- **PDFTK** (v2.1): Conjunto de herramientas para el tratado de archivos PDF. Se utiliza en la plataforma para la generación del archivo PDF de una presentación siguiendo una determinada plantilla.
- **GNU Ghostscript** (v9.1): Al igual que PDFTK, *Ghostscript* permite tratar con archivos PDF y realizar ciertas acciones con los mismos. Se utiliza en la plataforma para dividir los PDF de entrada en imágenes.

4.3 HARDWARE

Los medios hardware que se han utilizado para el desarrollo del proyectos han sido varios, utilizando en algunos casos medios hardware facilitados por terceros debido a su coste económico:

- **Computador**: Para el desarrollo del proyecto se ha utilizado el computador del alumno. La especificación del mismo es la siguiente:
  - **Procesador**: Intel Core i7-2670QM
  - **Memoria RAM**: 4 GB
  - **Disco Duro**: 500 GB
  - **Tarjeta Gráfica**: Nvidia GeForce GT540M 2 GB
  - **Sistema Operativo**: Windows 10 (x64) // Debian Jessie (x64)
- **Proyector**: Se ha utilizado un proyector para las pruebas del reproductor de la plataforma en diferentes resoluciones. La marca del proyector utilizado ha sido *Optoma*. Prestado por el Centro de Tecnologías y Contenidos Digitales de la Universidad de Castilla-La Mancha.

- **Dispositivos móviles**: Se han utilizado varios y diversos tipos de dispositivos móviles para hacer las pruebas de la iteración entre el público simulado de una presentación y la aplicación web de preguntas de la plataforma. Estos han sido: *Bq Aquaris E5*, *Sony Xperia S* y *LG Spirit*.

- **Mando de control de la presentación para el ponente**: Este mando se ha utilizado para hacer pruebas con los mandos utilizados por los ponentes para gestionar las presentaciones y probar además las teclas auxiliares. El modelo del mando de control ha sido *Logitech Professional Presenter r700*. Prestado por el Centro de Tecnologías y Contenidos Digitales de la Universidad de Castilla-La Mancha.

- **Mandos auxiliares para el público**: Este tipo de mandos se han utilizado con el mismo fin que los dispositivos móviles. La marca de los mandos auxiliares ha sido *Turning Technologies*. Prestados por el Centro de Tecnologías y Contenidos Digitales de la Universidad de Castilla-La Mancha.

- **USB receptor**: USB necesario para la recepción de las respuestas de los mandos auxiliares del público. Al igual que estos mandos, el USB receptor es de la marca *Turning Technologies* y ha sido prestado por el Centro de Tecnologías y Contenidos Digitales de la Universidad de Castilla-La Mancha.

CAPÍTULO 5

ARQUITECTURA

En este capítulo se describe la estructura modular de *iCROM* siguiendo un enfoque *top-down*, es decir, primero se hará una breve descripción general de los módulos que componen las aplicaciones de la plataforma, para luego pasar a describir más en detalle las decisiones de diseño e implementación de cada uno de ellos.

La plataforma *iCROM* está compuesta por dos aplicaciones:

- **pCROM**: Es la aplicación principal. Permite el despliegue efectivo de presentaciones creadas por el usuario, bien a partir de archivos *PDF* o a partir de los propios archivos de la aplicación, los llamados archivos *CROM* (.crom). Su núcleo es un reproductor que es capaz de reproducir desde imágenes hasta capturas de otros programas en tiempo real.

  Esta aplicación también consta de una serie de herramientas auxiliares que facilitan al usuario la realización de ciertas tareas en lo referente a la creación, despliegue y distribución de su presentación.

- **qCROM**: Es una aplicación auxiliar, diseñada como un sistema distribuido con una arquitectura cliente-servidor, que permite al usuario de la plataforma *iCROM* realizar breves cuestionarios al público de la presentación en tiempo real. El público podrá participar bien mediante mandos auxiliares que el ponente facilite, o bien desde su móvil/tablet/ordenador personal, utilizando la interfaz para el público que la aplicación proporciona.

  qCROM puede ejecutarse en un navegador en segundo plano y ser capturada por la aplicación principal *pCROM*, lo que permitiría tener dichos cuestionarios como una diapositiva más de la presentación.

  Esta aplicación proporciona al ponente una interacción bidireccional con el público asistente. La interacción en una presentación puede llegar a ser una gran aliada a la hora de potenciar las capacidades de enseñanza-aprendizaje en docencia presencial.

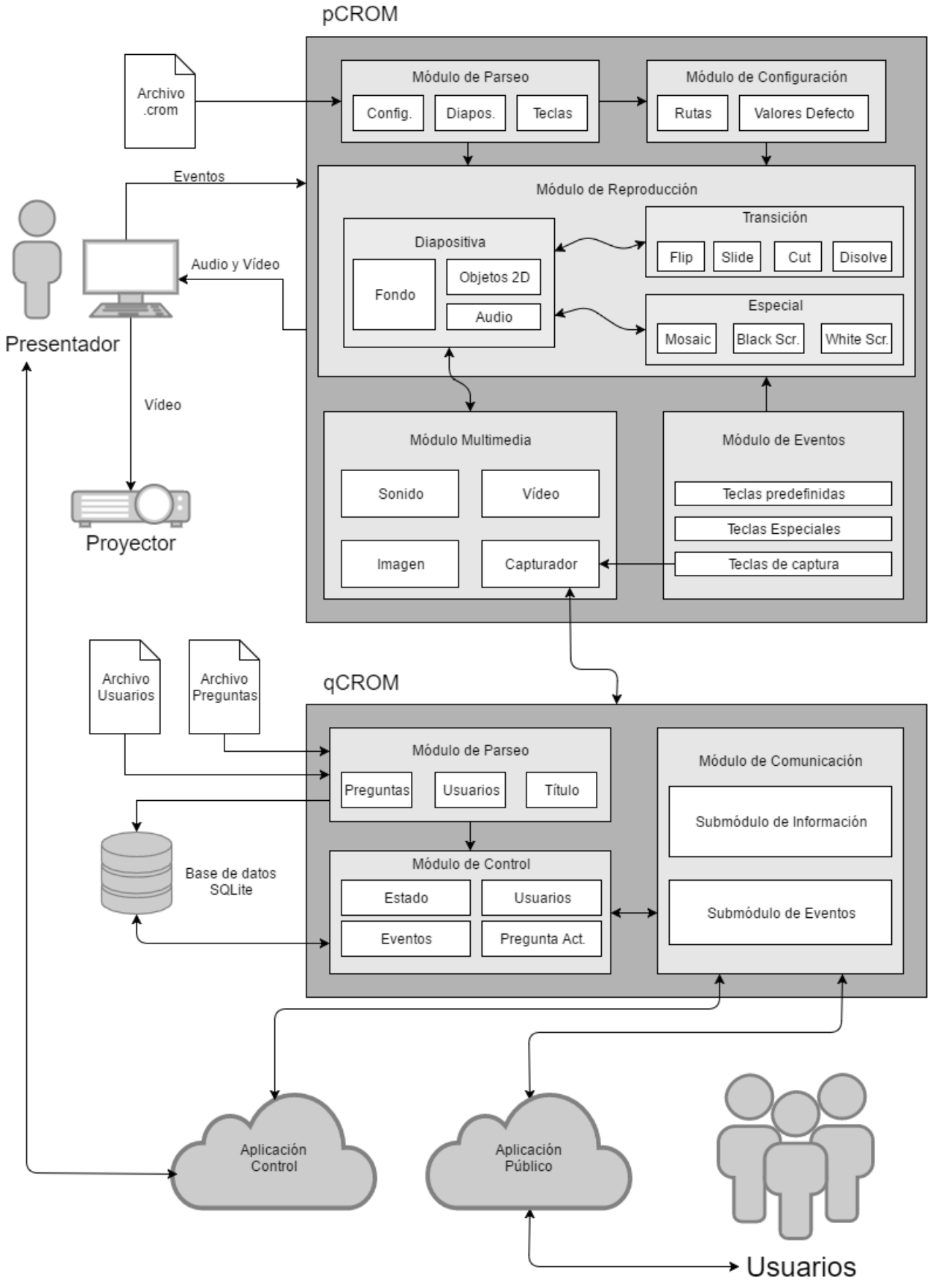


Figura 5.1: Esquema de la plataforma *iCROM*

La plataforma está formada por ocho módulos en total, cinco pertenecientes a la aplicación *pCROM* y tres pertenecientes a *qCROM*.

Breve introducción a los módulos de *pCROM*:

- *Módulo de parseo*: Este módulo se encarga de analizar los archivos *.crom* para obtener la configuración general, las diapositivas y las teclas especiales para la presentación.
- *Módulo de configuración*: Este módulo gestiona todo lo referente a la configuración de la presentación. Además, almacena rutas a archivo y valores por defecto para posibles usos durante la presentación.
- *Módulo de reproducción*: Este módulo es el encargado de mostrar en pantalla las diapositivas (fondo, imágenes, vídeos, etc.), transiciones y estados especiales de la presentación.
- *Módulo de multimedia*: Este módulo provee funcionalidad a las diapositivas a la hora de trabajar con el multimedia que las compone.
- *Módulo de eventos*: Este módulo gestiona todos los eventos de entrada que se puedan producir durante la presentación para que la aplicación responda de forma correcta a los mismos.

Breve introducción a los módulos de *qCROM*:

- *Módulo de parseo*: Este módulo se encarga de analizar el archivo con las preguntas a mostrar en la aplicación, así como el archivo con los posibles usuarios para su precarga en la base de datos.
- *Módulo de comunicación*: Este módulo gestiona la comunicación de la aplicación con el exterior. Permite responder a distintas peticiones de forma concurrente, dando servicio tanto a la parte local como a la parte remota.
- *Módulo de control*: Este módulo gestiona el estado de la aplicación, las preguntas y los participantes, así como la comunicación con la base de datos local.

5.1 DESCRIPCIÓN GENERAL

La estructura de la plataforma *iCROM* ha sido diseñada de forma modular. Desde un principio, la plataforma estaba pensada para que escalara con las necesidades de los usuarios, por lo que al comienzo de su desarrollo sólo se creó una pequeña base con una funcionalidad básica y según fueron surgiendo dichas necesidades se le fue añadiendo más y más

funcionalidad. Esta característica confiere a la plataforma un gran potencial, ya que si con el tiempo surgen nuevas necesidades por parte de los usuarios, extender la funcionalidad de las aplicaciones de la plataforma *iCROM* (o de la misma plataforma en sí) no requeriría de muchas modificaciones en código.

Otras características destacables de esta plataforma son:

- **Multimedia**: *pCROM* permite la carga de cualquier tipo de imagen, vídeo o sonido, independientemente de su formato. Esto libera al usuario de la preocupación de si los archivos multimedia utilizados por el mismo son o no compatibles con la aplicación, ya que estén en el formato que estén, la plataforma los va poder cargar sin problema.
- **Portabilidad**: Dado que uno de los requisitos de la plataforma era que ésta estuviera basada en librerías de código libre y multiplataforma, *iCROM* es capaz de compilar y ejecutarse allá donde estas librerías tengan soporte.
- **Flexibilidad**: Las diapositivas ya no se limitan a imágenes, vídeos y efectos entre diapositivas. *pCROM* da soporte para tener en una diapositiva todo lo que el usuario pueda imaginar, ya que gracias a su capturador se puede tener cualquier programa como diapositiva.
- **Personalización**: La aplicación auxiliar *qCROM* permite modificar su archivo con los estilos, tanto de la aplicación local como de la aplicación remota, con el fin de que el usuario pueda modificar la interfaz que muestra al público a la hora de hacer un cuestionario.
- **Configuración**: La aplicación principal *pCROM* consta de una configuración general totalmente editable, en la cual se podrá modificar desde los frame por segundo (FPS) a los que se quiere que se ejecute la aplicación, hasta rutas por defecto a archivos multimedia.
- **Mantenibilidad**: Al estar claramente definidas y acopladas las partes de la plataforma, es fácil editar el código fuente de la aplicación para solventar posibles problemas que pudieran surgir con el tiempo.

Otro aspecto importante de la plataforma *iCROM* es el uso de *patrones de diseño* [11] como **Abstract Factory**, **Facade** u **Observer**, y algunos patrones avanzados como puede ser el uso de punteros inteligentes (**SmartPointers**).

- El patrón *Abstract Factory* proporciona una interfaz para crear de forma sencilla objetos que pueden ser de distintos tipos, aunque pertenecen a una misma jerarquía de clases. Se utiliza para crear objetos multimedia, en código conocidos como *SlideObjects*.

- El patrón *Facade* permite gestionar una parte de la aplicación que sea de más bajo nivel o que requiera de un control especial desde otro módulo. Se utiliza para los managers y controllers de las aplicaciones de la plataforma.
- El patrón *Observer* se utiliza para que un determinado tipo de objeto pueda notificar y/o actualizar el estado de otros automáticamente. Este patrón lo sigue la clase *Slide* de la aplicación *pCROM* para notificar y actualizar todos los elementos que la componen.
- También se utilizan punteros inteligentes a lo largo y ancho de la aplicación *pCROM*, ya que facilitan la liberación de memoria utilizada por el programa durante su ejecución y en caso de interrupción inesperada de la misma.

En los siguientes puntos, se realizará una descripción más en detalle de cada uno de los aspectos de los módulos de las aplicaciones que componen la plataforma, haciendo incapié en aquellas dependencias y requisitos con otros submódulos y detalles de implementación relevantes.

5.2 APLICACIÓN pCROM

pCROM está implementado en C++ y utiliza (entre otras) las librerías de **OpenGL**, **SDL2**, **GStreamer 1.0** y **Boost**. Tiene por objetivo proporcionar al usuario una serie de herramientas auxiliares que le permitan crear, desplegar y distribuir su presentación de forma fácil y sencilla, así como para aprovechar el canal de retorno del público de la presentación.

Estas **herramientas auxiliares** (*Anexo D*) van completamente aparte del reproductor de la aplicación. Este reproductor representa el núcleo de la aplicación y necesita ser arrancado (ejecutando todos los procesos que eso conlleva) para poder funcionar. Las herramientas auxiliares se han implementado aparte del núcleo de la aplicación con el fin de que el usuario pueda utilizarlas sin tener que arrancar dicho reproductor. Aprovechando que estas herramientas funcionan sin el reproductor, éste utiliza algunas de estas herramientas para conseguir funcionar correctamente ante las entradas del programa, sin tener que implementar funcionalidad adicional. Los módulos que componen la aplicación son los siguientes:

- Módulo de parseo
- Módulo de configuración
- Módulo de reproducción
- Módulo multimedia
- Módulo de eventos

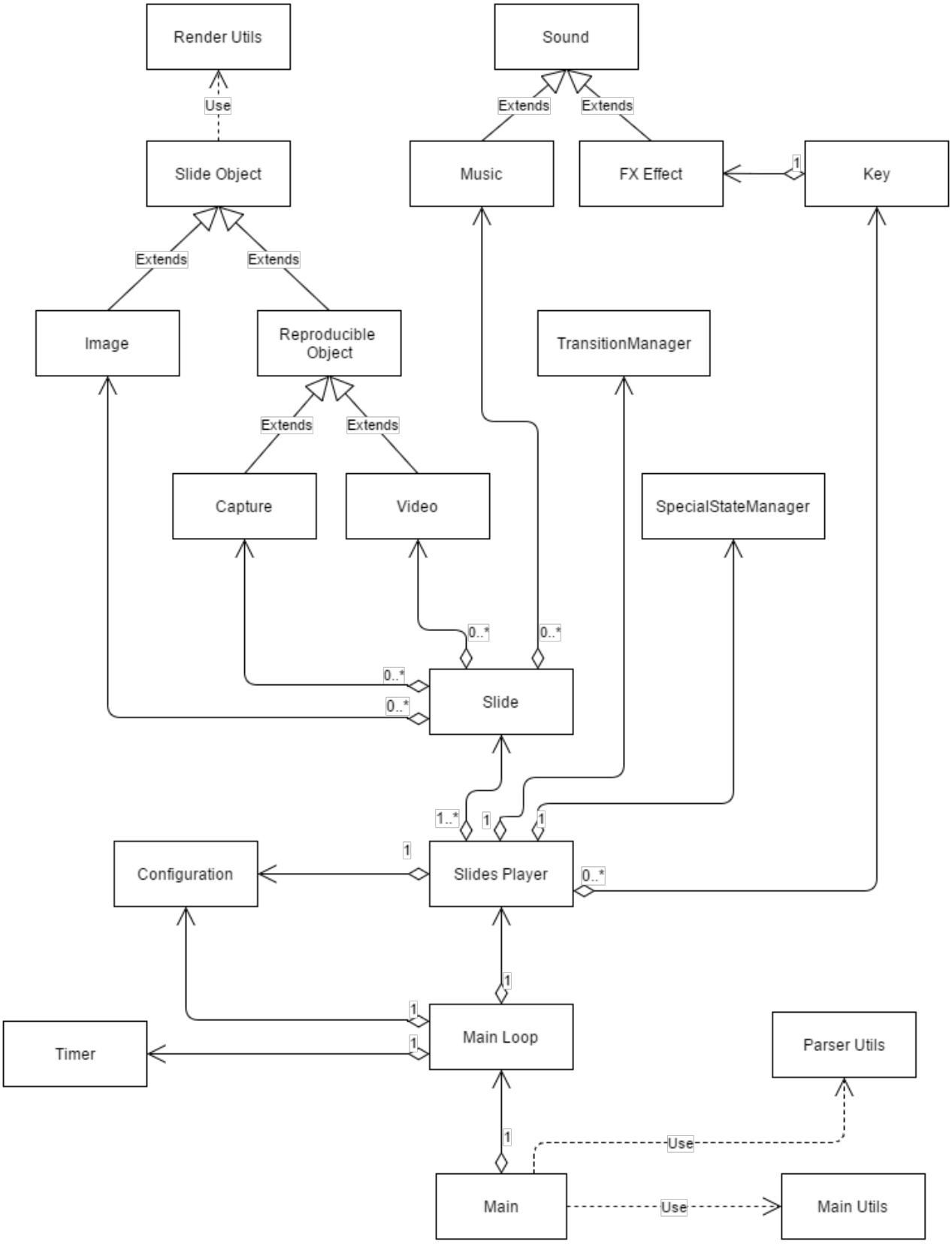


Figura 5.2: Diagrama de clases de la aplicación *pCROM*

Los módulos principales de la aplicación son el *módulo de parseo* y el *módulo de reproducción*, quedando el resto de módulos como módulos auxiliares. A continuación, se describirán más en detalle estos módulos y el papel que toman en la aplicación, así como algunos detalles de su diseño e implementación.

5.2.1 Módulo de parseo

Este módulo es el encargado de parsear los archivos de entrada de la aplicación. Estos archivos de entrada pueden ser archivos **PDF** o los propios de la aplicación, los archivos **CROM** (*Anexo B*). A partir de estos archivos, el módulo de parseo tiene por objetivo generar estructuras contenedoras con los datos obtenidos de dichos archivos para que el módulo de reproducción pueda empezar su ejecución.

El módulo de parseo hace uso de la librería de *YAML* para C++. Se ha elegido este lenguaje y no otro para la implementación de los archivos CROM porque su sintaxis relativamente sencilla y su diseño legible pero a la vez fácilmente mapeable, hacen de este lenguaje un lenguaje más que deseable para facilitar las tareas de diseño e implementación de un parser, a la vez que es vistoso de cara al usuario.

El módulo de parseo representa la única entrada del programa y sólo parsea archivos con extensión ".crom". Aun así, el módulo también admite archivos *PDF*. Para entender mejor esto, se hablará de como se trata cada tipo de archivo por separado.

Archivos PDF

En el diseño de la aplicación se pensó que los archivos PDF deberían ser una posible entrada de la misma ya que supondrían una ventaja para la aplicación más que una desventaja. Al admitir este tipo de archivo, la aplicación admite cualquier tipo de diapositivas generadas con otras herramientas ofimáticas. Hoy en día, cualquiera de estas herramientas permiten pasar un determinado contenido generado con los mecanismos que éstas ofrecen a un archivo PDF.

El módulo de parseo sólo trata archivos CROM. Cuando el módulo detecta que la entrada es un archivo PDF, lleva a cabo una serie de procesos que pasan ese archivo PDF a un archivo CROM. Para conseguir esto, el módulo requiere de una de las *herramientas auxiliares* de la aplicación, aquella que divide un PDF en las imágenes que lo componen. Esta herramienta auxiliar recibe el nombre de **split** y no sólo permite obtener las imágenes de un PDF, si no que también las permite redimensionar a la vez que las obtiene. Se decidió pasar el archivo PDF a archivo CROM porque tratar esos archivos por separado suponía añadir una parte al parser que en realidad no necesitaba, ya que el submódulo de parseo de archivos CROM tenía

que ser creado a la fuerza. De esta forma, el parser sólo requiere de un submódulo para tratar dos tipos de archivo diferentes de la misma manera.

El proceso de división de estas imágenes es sencillo. La herramienta auxiliar **split** hace uso de la parte de *filesystem* de *Boost* para crear un directorio temporal en el que alojar las imágenes que se obtengan del archivo PDF. Este directorio recibe el mismo nombre que tenga el archivo PDF en cuestion.

Listado 5.1: Comando de *Ghostscript* para la obtención de las imágenes de un PDF

```
$ gs -dNOPAUSE -sDEVICE=jpeg -r200 -dTextAlphaBits=4
   -dGraphicsAlphaBits=4 -dMaxBitmap=500000000 -dBATCH
   -dAlignToPixels=0 -dGridFitTT=2 -dJPEGQ=98
   -sOutputFile=directorio_temporal/%03d.jpg archivo_de_entrada.pdf
```

Una vez creado el directorio, se hace uso de *GNU Ghostscript* para dividir el PDF de entrada en las imágenes que lo componen (List. 5.1). Estas imágenes son almacenadas en la carpeta creada anteriormente. Para esta tarea se pudo haber utilizado *Imagemagick*, ya que ésta proporciona un mecanismo similar para la división de PDFs en imágenes. Tras el estudio de como funcionaban los mecanismos que Imagemagick ofrecía para esto, se observó que se apoyaba también Ghostscript, por lo que se decidió evitar utilizar Imagemagick para esta tarea y utilizar Ghostscript directamente con el fin de evitar procesos intermediarios que ralentizaran la obtención de las imágenes.

Ya con las imágenes almacenadas en el directorio temporal, se vuelve a llamar a la librería de *filesytem* para listar, **en orden**, las imágenes que contiene. Con esta lista de imágenes y la ruta al directorio temporal, el módulo de parseo genera un archivo CROM con dos secciones. La primera de ellas, la sección de configuración, contendrá dos parámetros: uno con la transición por defecto entre las dipositivas y otro con la ruta al directorio temporal con las imágenes. La segunda de estas secciones, la sección de las diapositivas, contendrá ordenadas tantas diapositivas como imágenes se encontraran en dicho directorio.

Listado 5.2: Ejemplo del resultado obtenido de pasar un archivo PDF a un archivo CROM

```
1  configuration:
2  {
3      transition_default_value: "cut",
4      images_directory: "archivo_de_ejemplo/"
5  }
6
7  slides:
8  {
9      {page: 1, background_image: 001.jpg},
10     {page: 2, background_image: 002.jpg},
11     {page: 3, background_image: 003.jpg}
12 }
```

Terminados los pasos intermedios para la obtención del archivo CROM, éste se analiza y los datos obtenidos son enviados al módulo de reproducción para que pueda empezar su ejecución. Salvo por estos pasos, el resto del proceso de análisis es como el que se describe en el siguiente apartado. Esta funcionalidad también se puede obtener por separado, ya que *pCROM* permite esta conversión de PDF a archivo CROM sin tener que arrancar la aplicación. Para más información sobre esta utilidad, se recomienda leer el anexo de *herramientas auxiliares*.

Archivos CROM

Si el módulo de parseo detecta que el archivo de entrada es un archivo con extensión "*.crom*", se pasa directamente a la parte de análisis. Cada archivo *CROM* está compuesto de tres secciones, cada una de las cuales está compuesta a su vez por una serie de parámetros cuyos efectos pueden influir tanto a la ejecución de la aplicación en sí, como a la totalidad de las diapositivas. El orden en el que se definan en el archivo CROM es irrelevante para el parser. Ésta es una de las ventajas que se obtiene al usar *YAML*. Estas secciones son *Configuración general*, *Diapositivas* y *Teclas especiales*.

El primer paso que lleva a cabo el módulo de parseo es cargar el contenido del archivo con YAML. De esta tarea se encarga la función **prepare_crom_file**, la cual toma de entrada el nombre del archivo que se pretende parsear. Si la carga del archivo falla, el módulo de parseo se intentará recuperar del error reformateando el archivo de entrada e intentando volver a cargarlo con YAML. Si el error vuelve a aparecer, el módulo de parseo lanzará una excepción. El parser se intentará recuperar del error reformateando el archivo de entrada debido a que muchos de los errores encontrados empleando YAML son por el mal formateado de sus archivos. De esta manera, se intenta evitar que el usuario tenga que reformatear sus archivos para que la aplicación los admita. Si este reformateo no tiene éxito en su búsqueda de solventar el error, éste se deberá a algún error por parte del usuario a la hora de crear un objeto o sección del archivo. Si esto pasa, como se ha dicho anteriormente, el programa emitirá una excepción reportando el problema, la cual será recogida por la clase principal, y ésta a su vez mostrará el error por pantalla, junto a una posible solución. Al reportar el error, la clase principal liberará la memoria utilizada por la aplicación, así como posibles elementos temporales utilizados por el parser. Este proceso se llevará a cabo cada vez que el módulo encuentre un error en el archivo CROM de entrada.

Si la carga del archivo con YAML ha sido satisfactoria, el módulo tendrá un nodo YAML con tres posibles subnodos. Un nodo en YAML se puede definir como un conjunto de datos estructurados de una determinada forma. Estos nodos pueden pertenecer directamente al primer nivel de la jerarquía de nodos o formar parte de otro nodo de dicha jerarquía. De esta forma, dentro de un nodo de YAML se pueden encontrar más nodos, y dentro de éstos aún

más nodos, etc.

Al obtener en el primer paso el nodo raíz del archivo, dentro de éste podrán aparecer subnodos, los cuales en este caso representarán las secciones de los archivos CROM anteriormente comentadas. De entre esos subnodos, se buscará en primer lugar el que representa la configuración de la presentación. De esta tarea se encarga la función **parse_configuration_node**, la cual toma de entrada el nombre del archivo fuente y el nodo a parsear.

Una vez en esta función, se crea un objeto de la clase **Configuration**. El constructor de esta clase genera una configuración general con la cual el módulo de parseo pueda trabajar. Una vez creado este objeto, se buscan todos aquellos parámetros válidos de este nodo. Esta sección contiene parámetros que pueden afectar a la ejecución de la propia aplicación, así como a valores por defecto de la sección de las diapositivas. Cada parámetro que encuentre el parser, será analizado, tratado y almacenado en la configuración general que se generó anteriormente. El módulo de parseo empieza buscando este subnodo y no otro porque el resto de secciones dependen de lo que éste pueda contener, por eso en caso de no existir, se crea una configuración genérica que permita a la aplicación arrancar y ejecutarse sin problemas.

Terminado el análisis del nodo con la configuración de la presentación, el módulo de parseo tratará de obtener los datos contenidos en el nodo con las *diapositivas*. De esta tarea se encarga la función **parse_sequence_slides_node**, la cual toma por parámetros de entrada el nodo con las diapositivas y la configuración obtenida en el paso anterior.

Este nodo contiene tantos subnodos como diapositivas haya definido el usuario en el archivo CROM. La función encargada de analizar este nodo crea un vector de *STL* donde irá almacenando cada una de las diapositivas que vaya obteniendo durante el parseo. Para cada uno de los subnodos, se pregunta por su número de diapositiva y, si éste se obtiene, se pasa dicho subnodo junto al número obtenido y la configuración a la función **parse_slide_content**, la cual se encargará de desmenuzar dicho subnodo y obtener toda la información que éste contenga, devolviendo un objeto de la clase *Slide*, formado por la combinación de objetos del tipo *SlideObjet* y *Music*.

Los nodos de las dispositivas contienen una secuencia de parámetros de los cuales una gran mayoría son comunes para todas las diapositivas. Estos pueden ser: el *tipo de transición*, el *fondo*, el *flag de cabecera*, etc. Por tanto, la función encargada de desmenuzar estos nodos divide la tarea en cinco partes:

- Obtención de los parámetros generales. Esta tarea la consigue delegando en la función **parse_general_content**.
- Obtención de las posibles imágenes de la diapositiva. La función crea un vector *STL* que almacena objetos de la clase **Image**. Este vector se rellena llamando a la función **parse_image_content** por cada nodo encontrado en el subnodo con las imágenes.

- Obtención de los posibles vídeos de la diapositiva. La función crea un vector *STL* que almacena objetos de la clase **Video**. Este vector se rellena llamando a la función **parse_video_content** por cada nodo encontrado en el subnodo con los vídeos.

- Obtención de las posibles capturas de la diapositiva. La función crea un vector *STL* que almacena objetos de la clase **Capture**. Este vector se rellena llamando a la función **parse_capture_content** por cada nodo encontrado en el subnodo con las capturas.

- Obtención de los posibles sonidos de la diapositiva. La función crea un vector *STL* que almacena objetos de la clase **Music**. Este vector se rellena llamando a la función **parse_music_content** por cada nodo encontrado en el subnodo con los sonidos.

Listado 5.3: Obtención de las diapositivas y reordenamiento

```
1  /* Ordena el vector de diapositivas */
2  bool number_sort (const std::shared_ptr<Slide> &s, const
       std::shared_ptr<Slide> &ss)
3  {
4      return s->get_number() < ss->get_number();
5  }
6
7  /* Parsea el nodo obtenido del archivo CROM */
8  void parse_crom_file (...)
9  {
10     ...
11
12     if (seq_node[SLIDES_TAG])
13     {
14         slides = parse_sequence_slides_node(seq_node[SLIDES_TAG],
               config);
15         std::sort(slides.begin(), slides.end(), number_sort);
16     }
17
18     ...
19 }
```

Una vez parseadas todas las diapositivas, el vector debe ser reordenado en función del número obtenido al parsear cada una de las diapositivas. Esto se realiza de esta forma porque YAML permite acceder a los nodos de un archivo mediante un *iterator*, es decir, permite acceder a los nodos de forma aleatoria. STL ofrece una función que permite ordenar un vector en función de un criterio. En este caso, el vector se reordena en función de este número que se comentaba anteriormente. En principio lo que parecía una desventaja, ahora permite al usuario definir diapositivas en un archivo CROM en el orden que quiera, siempre y cuando sigan un orden lógico.

En el último paso, el módulo de parseo busca el nodo con las *teclas especiales*. Esta tarea la realiza la función **parse_sequence_keys_node**, la cual devuelve un vector *STL* con todas

aquellas teclas que encuentre durante el parseo de dicho nodo. Esta función busca cada uno de los subnodos con las teclas y obtiene toda la información que estos contienen. Cada uno de estos subnodos almacena pares **tecla-acción**, donde la *acción* puede ser reproducir un efecto de sonido o renderizar un estado especial. Cada una de estas teclas se almacenan en un objeto de la clase **Key**, donde una variable discriminante (*KeyAction*) indica la acción que realizará dicha tecla al ser pulsada por el usuario.

Una vez el módulo de parseo ha obtenido la *configuración global*, la lista con todas las *diapositivas* y la lista con todas las *teclas especiales*, éste delega dichos objetos en el módulo de reproducción, el cual empieza su ejecución. Como se ha podido observar, el módulo de parseo lo componen un conjunto de funciones auxiliares, las cuales permiten parsear archivos CROM por partes, de mayor a menor importancia. Se ha decidido crear un parser por funciones y no mediante una clase gestora ya que no se consideraba necesaria una instancia de una clase que gestionara el análisis de este tipo de archivos. Mediante el empleo de estas funciones desde la clase principal se podía conseguir el objetivo buscado sin mucha complicación.

5.2.2 Módulo de configuración

Este módulo es el encargado de proveer de una configuración a toda la aplicación. Esta configuración, como ya se ha comentado en el apartado anterior, se crea en primera instancia como una configuración genérica, la cual contiene las variables y los valores necesarios para que la aplicación funcione sin problemas.

El usuario debe decidir, al crear un archivo *CROM*, si es necesario cambiar alguno de estos valores por defecto. Esto manifiesta que la sección de configuración en un archivo *CROM* es totalmente opcional. La configuración genérica está pensada para que la aplicación ofrezca un soporte básico para la reproducción de una presentación, no requiriendo (en la mayoría de los casos) una modificación de esta sección por parte del usuario. Esto está más bien pensado para los usuarios que no dominen mucho la aplicación, quedando la parte de modificación de esta sección para aquellos usuarios más avanzados que deseen sacar más partido a *pCROM*.

La configuración es accesible desde tres clases distintas. Estas son *MainLoop*, *SlidesPlayer* y *ParserUtils*. Aún siendo accesible desde tres clases distintas, sólo existe una única instancia de esta configuración en la aplicación. Esto puede recordar al patrón de diseño **Singleton**, aunque el módulo de configuración no implementa dicho patrón. A la hora de diseñar esta clase se decidió que este patrón no se usaría para su implementación y que en cambio se haría un mayor control sobre la instancia a la hora de compartirla con otras clases, ya que el patrón singleton no siempre se ve como una buena decisión de diseño y es conocido por muchos como el *antipatrón*. Las variables y los valores por defecto de este módulo son los siguientes:

- La tasa de **imágenes por segundo** (**FPS**) de la aplicación. Esta variable toma un valor por defecto de *60*. Esta tasa indica la velocidad a la cual la aplicación muestra imágenes por pantalla, es decir, las veces que el módulo de reproducción renderiza una diapositiva en un segundo. En presentaciones en las cuales no existan vídeos/capturas (ni muchos efectos entre diapositivas) esta tasa se puede bajar a valores entre *15-25*, lo que haría que el programa consumiera menos ciclos de CPU y fuera más ligero para el sistema. Es peligroso bajar dicha tasa a valores inferiores de 15, ya que puede que la aplicación no responda debidamente ante eventos como puedan ser las pulsaciones de teclas. Tampoco es recomendable aumentar la tasa a valores superiores de 60, ya que el ojo humano no percibirá ninguna diferencia respecto a lo que percibe con 60 FPS, y si hará que el programa consuma más tiempo de CPU y sea más pesado para el sistema.

  En caso de tener presentaciones con vídeos/capturas, el valor que debe tomar esta variable no debe ser inferior al máximo de todos los FPS de los vídeos/capturas de la presentación. Si por ejemplo en una presentación existieran tres vídeos con una tasa de 25, 25 y 50 respectivamente, la tasa de imágenes por segundo de la aplicación no debe ser inferior a la del tercer vídeo, es decir, no debería ser inferior a 50 FPS. Esto debe ser así ya que al renderizar la diapositiva donde se encuentre dicho vídeo se perderían muestras y se obtendría un vídeo posiblemente entrecortado, con saltos entre una pasada del módulo de reproducción a otra.

- El valor por defecto de **transición entre diapositivas**. En la aplicación, las transiciones entre diapositivas se producen a *corte* por defecto. Este valor afecta a todas las diapositivas, ya que si en estas diapositivas no se indica que tipo de transición se desea reproducir, esa transición tomará el valor que esta variable tenga. Los valores que puede tomar esta variable son:

  – *Corte*: la transición entre diapositivas se realiza de forma instantánea, cambiando de diapositiva de una pasada del módulo de reproducción a otra. Es el modo por defecto para las transiciones en el reproductor debido a su sencillez de ejecución.

Figura 5.3: Transición Corte

  – *Deslizar*: la transición entre diapositivas se realiza como si dichas diapositivas se "deslizaran" una a continuación de otra en la dirección indicada por el usuario. Si el usuario quiere volver a una diapositiva anterior, dichas diapositivas se deslizarán hacia la derecha, y si quiere ir a la siguiente diapositiva disponible, lo harán hacia la izquierda.

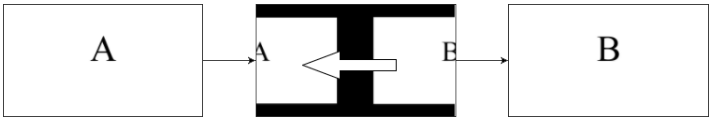


Figura 5.4: Transición Deslizar

– *Disolver*: la transición se realiza en profundidad, es decir, la nueva diapositiva se renderizará debajo de la actual, mientras que esta en un primer plano irá perdiendo valor en su canal *alpha* (se irá haciendo transparente). En el momento que la diapositiva en primer plano sea completamente transparente, la diapositiva que se renderizaba en un segundo plano pasará al primero y la transición habrá acabado.

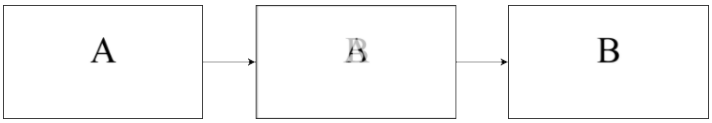


Figura 5.5: Transición Disolver

– *Voltear*: la transición se realiza girando las diapositivas en el **eje Y**. La diapositiva actual se renderiza hasta que su ángulo alcanza un cierto valor, posteriormente se renderiza la siguiente diapositiva en sentido contrario hasta que el ángulo alcanza el valor cero. Las diapositivas no llegan en ningún momento a girar completamente, si no que se juega con la percepción del usuario. Esto se realiza así porque si se colocará la siguiente diapositiva justo detrás y se girara a la vez que la actual, el resultado final después del que el giro terminara sería una diapositiva espejada de la original.

Figura 5.6: Transición Voltear

– *Rotar*: al igual que la transición anterior, ésta se renderiza girando las diapositivas. En esta ocasión, el giro se produce en el **eje Z**. Ya no sólo aplica la rotación en dicho eje, si no que reduce y amplia las diapositivas durante la transición, jugando a la vez con su canal *alpha*. El primer paso es reducir y girar la diapositiva actual en el sentido contrario a la siguiente diapositiva. Mientras ésta se reduce y gira, su canal alpha va disminuyendo hasta que éste llega a 0. En el momento que esta situación se da, la siguiente diapositiva entra girando y ampliándose por el lado contrario al lado por el cual desapareció la diapositiva anterior, partiendo de un alpha a 0. En el momento que la diapositiva llega al centro de la pantalla, alcanzando el tamaño máximo y un alpha a 1, la transición habrá acabado.

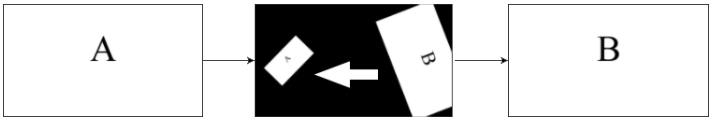


Figura 5.7: Transición Rotar

– *Alejar-Acercar*: ésta es una transición muy simple. Consiste en renderizar la diapositiva jugando con su tamaño en pantalla, reduciendo la diapositiva actual hasta que la misma desaparece. En ese momento se sustituye la diapositiva actual por la siguiente (con el mismo tamaño) y se empieza a ampliar hasta que alcanza el tamaño máximo.

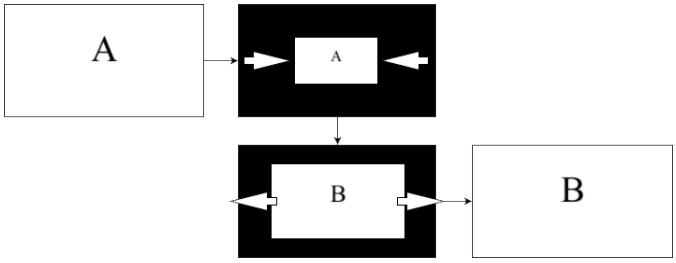


Figura 5.8: Transición Alejar-Acercar

La clase encargada de reproducir estas transiciones es la clase **TransitionManager**, la cual necesita de tres parámetros: la diapositiva actual, la diapositiva siguiente y el tipo de transición. El cómo y el cuándo se lleva a cabo el renderizado se verá más en detalle en el ***módulo de reproducción***.

- El **archivo de cabecera** y su valor por defecto. El archivo de cabecera es una imagen a pantalla completa que se renderiza siempre en último lugar sobre la diapositiva actual, lo que permite poner imágenes de cabecera, marcas de agua, etc. Esta variable toma un valor por defecto de *off*, es decir, aunque se indique un archivo de cabecera, si el valor por defecto no toma un valor de *on*, ninguna de las diapositivas que indiquen explícitamente que quieren que se muestre dicha imagen lo harán.

- Los **volúmenes por defecto** de vídeos, música y efectos de sonido. Esta variable permite indicar a la aplicación qué volumen dar a aquellos recursos multimedia con sonido en los cuales el usuario no ha establecido ningún volumen explícito. Toma un valor por defecto de *1.0*, es decir, un 100 % del volumen del computador donde se esté llevando a cabo la presentación.

- La **reproducción en bucle** de vídeos y música. Esta variable permite establecer si se desea que los vídeos y la música de la presentación se reproduzcan en bucle si el usuario no ha indicado nada al respecto en el objeto multimedia. Por defecto toma un valor negativo, *off*, lo que quiere decir que si no se indica lo contrario, los vídeos y música de

la presentación se reproducirán una única vez *cada vez que la diapositiva en la que se encuentren entre en escena*.

- Los **directorios multimedia**. Estos directorios permiten indicar a la aplicación donde buscar el media de las diapositivas (imágenes, vídeos y sonidos). Por defecto, la aplicación busca en el directorio actual de ejecución.

La modificación de algunos de los parámetros de este módulo no tiene otro objetivo que no sea el ahorrar trabajo a la hora de definir diapositivas por parte del usuario. La configuración general no sólo es útil a la hora de definir variables que afectaran a la aplicación, si no que también es útil a la hora de definir variables globales que afectaran a todas las diapositivas en general.

5.2.3 Módulo de reproducción

El módulo de reproducción es el encargado de renderizar cada una de las diapositivas de la presentación. Este renderizado se realiza un número determinado de veces por segundo, dependiendo de la tasa de imágenes que el programa tenga definida. Por defecto, como se mencionó en el ***módulo de configuración***, la tasa por defecto que se establece para la aplicación es de *60* imágenes por segundo. Con el fin de ampliar el rango de computadores donde *pCROM* pueda ejecutarse, esta tasa puede ser modificada, pudiendo bajarse y así suponer menos carga para el sistema. Esto supondría una ventaja para aquellos computadores con menos capacidad de procesamiento.

La clase encargada de llamar a renderizar es la clase *MainLoop*, la cual representa el bucle principal de la aplicación. Además, esta clase también tiene por objetivo controlar la tasa de imágenes por segundo, así como capturar los eventos externos de la aplicación con el fin de pasárselos a la clase encargada de interpretarlos.

La función principal de esta clase es la función **run**. Esta función contiene el bucle de ejecución principal de la aplicación, el cual podemos ver en el Listado 5.4. El reproductor de la plataforma implementa su propio bucle de ejecución con el objetivo de tener un mayor control sobre lo que pasa en la aplicación iteración a iteración. Se podría haber utilizado otros bucles ya implementados por terceros, pero se hubiese perdido este control extra sobre la ejecución de la aplicación.

Como se puede observar, el bucle principal tiene tres partes. Cada una de estas partes realiza una tarea muy importante en la ejecución de la aplicación. Estas tareas son la captura de eventos locales (de teclado o de la ventana de la aplicación), dibujado de las diapositivas/transiciones/estados especiales, y control de la tasa de imágenes por segundo:

Listado 5.4: Bucle principal de la aplicación *pCROM*

```
while (!_quit)
{
    this_time  = _cap_timer.start();
    delta_time = (float)(this_time - last_time) / 1000;
    last_time  = this_time;

    /*********** Eventos ***********/

    handle_local_events();

    if (!_quit)
    {
      /*********** Render ***********/

      _player->display(delta_time);

      /******** Control FPS ********/

      fps_ticks = _cap_timer.get_ticks();

      if (fps_ticks < ticks_x_frame)
      {
        SDL_Delay(ticks_x_frame - fps_ticks);
      }
    }
}
```

- La parte de **control de eventos locales**. Estos eventos hacen referencia a pulsaciones de teclas por parte del usuario así como eventos que se puedan producir en la ventana de la aplicación (minimizado, maximizado, cambio de tamaño, pérdida del foco, etc.)

- La parte de **renderizado**. En esta parte se llama a la función **display** de la clase *SlidesPlayer*, la cual representa el núcleo del módulo de reproducción. Esta función requiere de un tiempo *delta*, el cual representa el tiempo transcurrido entre una pasada del bucle y otra. Este tiempo es esencial a la hora de renderizar transiciones entre diapositivas, ya que permite que éstas se reproduzcan de la misma forma en computadores que permitan ejecutar la aplicación sin problemas, y en computadores a los cuales les cueste más renderizar en un momento dado de la presentación. Para calcularlo, es necesario usar la clase *Timer*, la cual cuenta *Ticks* (instantes de tiempo) entre una pasada del bucle y otra. Cada vez que el bucle empieza una nueva iteración, el temporizador se inicia, y al iniciarse devuelve los *ticks* de ese momento. Para calcular *delta*, sólo hace falta restar los ticks de la iteración anterior con los de la actual y pasar el resultado a segundos.

- La parte de **control de la tasa de imágenes por segundo**. Este control se realiza apoyándose de nuevo en los *ticks* obtenidos gracias al objeto de la clase *Timer*. En este caso, para controlar la tasa de imágenes por segundo, es necesario saber cuantos ticks

han transcurrido desde que se inició la iteración hasta que ésta ha terminado. La clase *Timer* provee un método para obtener dichos ticks de forma rápida y sencilla, **get_ticks**. Una vez obtenidos dichos ticks, si esa cantidad es inferior a la tasa de imágenes por segundo establecida para la aplicación en ticks, el bucle entra en suspensión un tiempo determinado, el cual será la diferencia que hay entre dichos ticks y los ticks obtenidos del temporizador. Para obtener la representación de la tasa de imágenes por segundo en ticks, es tan sencillo como realizar la siguiente operación:

```
1     float ticks_x_frame = 1000 / _config->get_fps_limit();
```

Reproductor

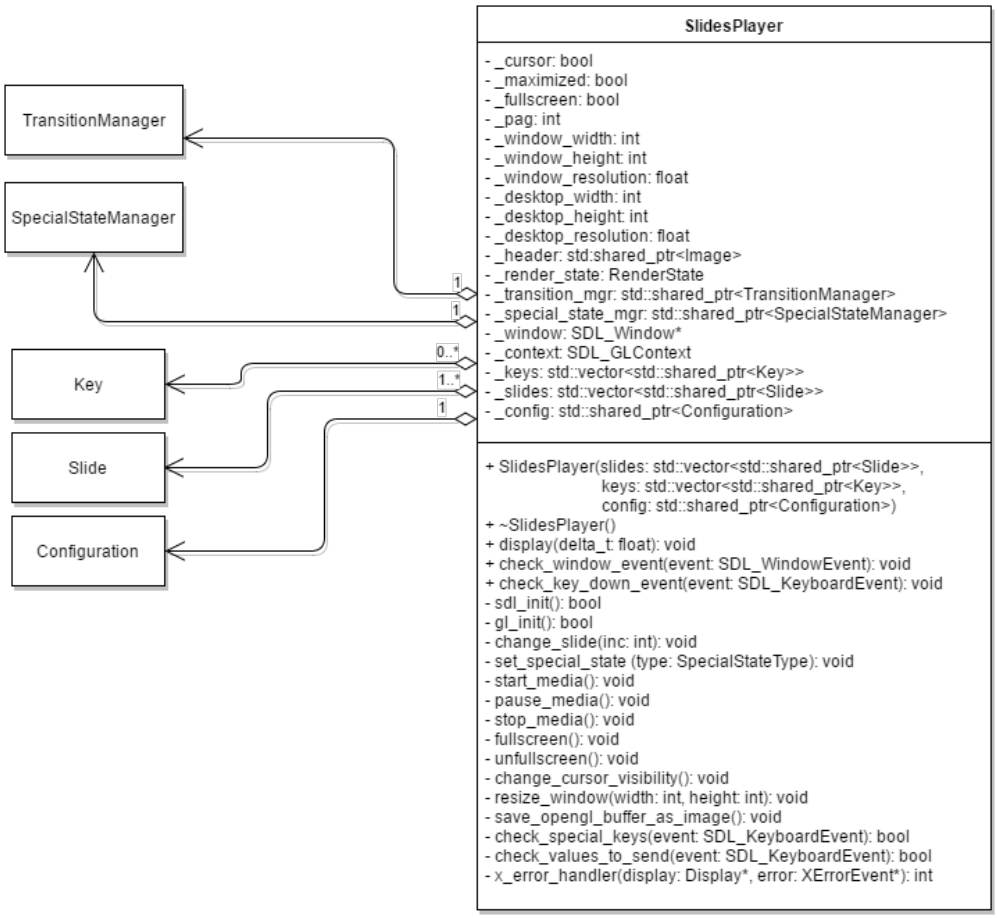


Figura 5.9: Diagrama de clases *SlidesPlayer*

La clase **SlidesPlayer**, anteriormente mencionada, es la encargada de renderizar todas las imágenes, vídeos y capturas de una diapositiva. Además, también se encarga de reproducir música y los efectos de sonido que puedan ser producidos por la pulsación de teclas especiales.

Esta clase hace las veces de gestor de la presentación así como de reproductor, ya que sobre ella recaen responsabilidades como interpretar las pulsaciones de teclas del usuario, controlar

el estado de la presentación, gestionar las diapositivas y las teclas especiales, inicializar los recursos multimedia, etc.

El renderizado que lleva a cabo esta clase se basa en estados. Estos estados pueden ser: *nada* (SNULL), *diapositiva* (SSLIDE), *transición* (STRANSITION) y *especial* (SSPECIAL). En código, esto se representa mediante un *enum* (el cual recibe el nombre de **RenderState**) con cuatro valores:

```
1 enum RenderState { SNULL, SSLIDE, STRANSITION, SSPECIAL };
```

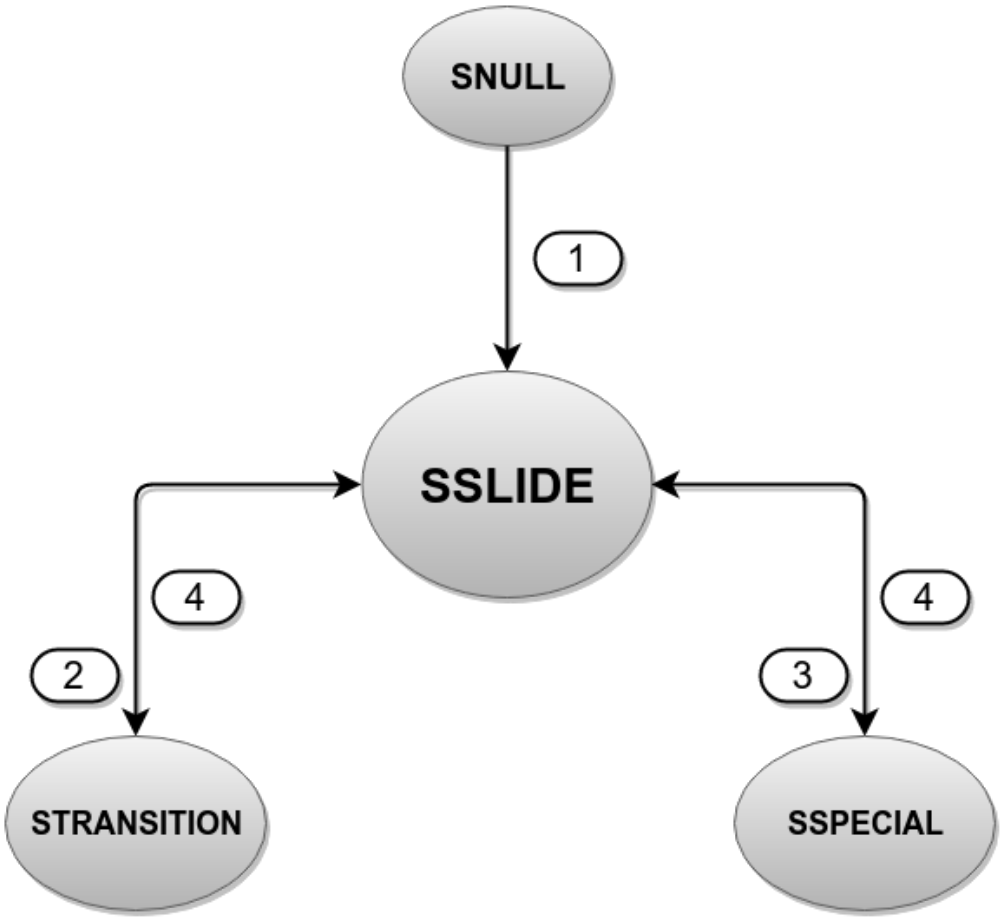


Figura 5.10: Estados del reproductor de *pCROM*

Teniendo en cuenta la figura 5.10 se podrá observar que cuando se inicializa el reproductor, el estado del mismo pasa de *SNULL* a *SSLIDE* (1). Si se produce un cambio de diapositiva (una transición), el estado pasa de *SSLIDE* a *STRANSITION* (2), y si se produce un estado especial, se pasa de *SSLIDE* a *SSPECIAL* (3). Cuando estas transiciones o estados especiales terminan, se vuelve al estado *SSLIDE* (4). De esta forma se consigue que la clase únicamente tenga una función para el renderizado, la cual renderiza una cosa u otra dependiendo del estado actual del reproductor. Cabe destacar que esta función delega el dibujado de los objetos en clases especiales que gestionan cada uno de los posibles estados del reproductor. En caso

de renderizar una transición, es su gestor el que realiza la tarea de dibujado, siendo un caso similar el del gestor de estados especiales cuando el estado del reproductor pasa de SSLIDE a SSPECIAL.

SlidesPlayer utiliza las librerías de **OpenGL**, **SDL** y **GStreamer**. En el diseño de la aplicación se decidió utilizar estas librerías porque SDL ofrece funcionalidad para la captura de eventos de teclado, captura de eventos de la ventana de la aplicación y creación de un contexto para el marco de OpenGL. GStreamer ofrece funcionalidad para la reproducción de sonidos y vídeo, así como para capturar programas en segundo plano. Se pudo haber utilizado SDL para la parte de la reprodución de sonido, pero se decidió integrar esta parte también con GStreamer para tener toda la parte de reproducción del media bajo la misma tecnología. Además, GStreamer da soporte para cualquier formato de vídeo y audio, mientras que SDL no. Por último, OpenGL es utilizado para dibujar las imágenes y los buffers de los vídeos/capturas en pantalla.

Las *diapositivas*, *configuración* y *teclas especiales* son recibidas desde el ***módulo de parseo*** por esta clase. Por tanto, en ella recae la responsabilidad de inicializar cada una de las diapositivas (obtener sus texturas, redimensionar, etc) y cada una de las teclas especiales basándose en dicha configuración. De la tarea de inicialización se encarga la función **init** de la clase **SlidePlayer**.

Esta función inicializa varios aspectos del reproductor. Lo primero que se inicializa es SDL, OpenGL y GStreamer, por ese orden. GStreamer podría inicializarse antes que SDL y OpenGL, el único que depende del orden para poder inicializarse es OpenGL, ya que requiere del contexto que crea SDL para poder dibujar en pantalla. La realización de la tarea de inicializar esta parte son las funciones *sdl_init()*, *gl_init()* y *gst_init()*. Una vez inicializados SDL, OpenGL y GStreamer, el siguiente paso es inicializar cada una de las diapositivas y teclas especiales obtenidas en el módulo de parseo.

Listado 5.5: Inicialización de SDL, GStreamer y OpenGL

```
1  bool SlidesPlayer::init ()
2  {
3    /* Inicializacion de SDL y OpenGL*/
4    if (!sdl_init() || !gl_init())
5    {
6      return false;
7    }
8
9    /* Inicializacion de GStreamer */
10   gst_init(0, 0);
11
12   ...
13 }
```

La inicialización de las teclas especiales se ve más en detalle en el ***módulo de eventos***. En

cuanto a la inicialización de las diapositivas, cada una de éstas está representada en código mediante una clase del tipo **Slide**, la cual tiene un método de inicilización que requiere de tres parámetros: el ancho y el alto de la ventana, así como de la resolución de la misma. Por último, se pasa del estado *SNULL* al estado *SSLIDE*, ya que existe una diapositiva en memoria y el reproductor la puede empezar a dibujar los objetos en pantalla.

Listado 5.6: Inicialización de diapositivas y carga de la primera en memoria

```
1  bool SlidesPlayer::init ()
2  {
3    ...
4
5    /* Inicializacion de las diapositivas */
6    for (auto slide : _slides)
7    {
8      slide->init(_window_width, _window_height, _window_resolution);
9    }
10
11   /* Cargamos la primera diapositiva en memoria */
12   _slides[_pag]->load(_window_width, _window_height,
         _window_resolution);
13
14   ...
15
16   _render_state = SSLIDE;
17 }
```

Los valores que se pasan a la función **init** de la clase *Slide* se obtienen al inicializar SDL, ya que es en este paso donde se crea la ventana de la aplicación. Por defecto, estas dimensiones son las que el escritorio tenga, ya que la ventana se crea siempre maximizada. Se decidió que la ventana se mostrara maximizada y no a pantalla completa porque de esta manera el usuario podría hacer otras cosas mientras *pCROM* arranca y no perder el foco de lo que esté haciendo cuando la aplicación se muestrara por pantalla. De esta forma es el usuario el que elige cuando establecer la aplicación a pantalla completa e inicializar la presentación y no al revés.

El método *init* de la clase *Slide* no carga la diapositiva en memoria. Su único objetivo es inicializar las tuberías de reproducción de los posibles vídeos, música y capturas que pueda tener la diapositiva en cuestión.

Para poder cargar una diapositiva en memoria es necesario recurrir a la función **load** de la clase *Slide*. Esta función carga las texturas de las imágenes de la diapositiva, inicia la reproducción de vídeos y música, así como las capturas que ésta pueda tener. Por último, redimensiona todo en función a las dimensiones de la ventana que se le pasan a la función. La diapositiva se libera de memoria utilizando la función **unload** de la clase *Slide*. Esta función detiene la ejecución de posibles vídeos, música y capturas que pueda tener la diapositiva, para después eliminar de memoria las texturas utilizadas por la misma.

La carga de una diapositiva en memoria y su posterior eliminación se realiza de forma dinámica con el fin de no ocupar demasiado espacio en memoria, ya que ante presentaciones muy extensas, la capacidad de memoria que éstas requerían podía llegar a ralentizar el computador al ocupar la mayoría de la memoria disponible.

La tarea de carga y eliminación de diapositivas en memoria recae en la clase **TransitionManager**. Esta clase tiene por objetivo gestionar las transiciones entre diapositivas, así como gestionar esta carga y eliminación de las mismas de memoria. Para realizar esta tarea, este gestor requiere de la diapositiva actual y de la siguiente. Estas diapositivas son pasadas al manager mediante la función **set_slides** de dicha clase. En el momento que el manager tiene las diapositivas necesarias para renderizar una transición, el estado del reproductor pasa de *SSLIDE* a *STRANSITION*, delegando el renderizado actual en dicho gestor. La función *set_slides* carga la siguiente diapositiva en memoria junto con la actual, la cual ya se encuentra cargada. A su vez, inicializa ciertos valores dependiendo del tipo de transición que se vaya a renderizar. Cabe destacar que la transición que se renderiza es aquella que se le ha asignado a la diapositiva **siguiente** y no a la actual. Una vez la transición haya acabado, el gestor elimina de memoria la diapositiva anterior y cede el control al reproductor, cambiando el estado de *STRANSITION* a *SSLIDE*.

Las transiciones del reproductor están limitadas a una **proyección Ortogonal**, ya que el reproductor utiliza este tipo de proyección. Dicha proyección se crea trazando la totalidad de las rectas de un objeto para después proyectarlas perpendicularmente a un cierto plano. En otras palabras, esto sería como coger un objeto 3D y aplastarlo contra una pared. El resultado que se obtendría serían objetos 3D convertidos a 2D, similar a lo que se muestra en la figura 5.11.

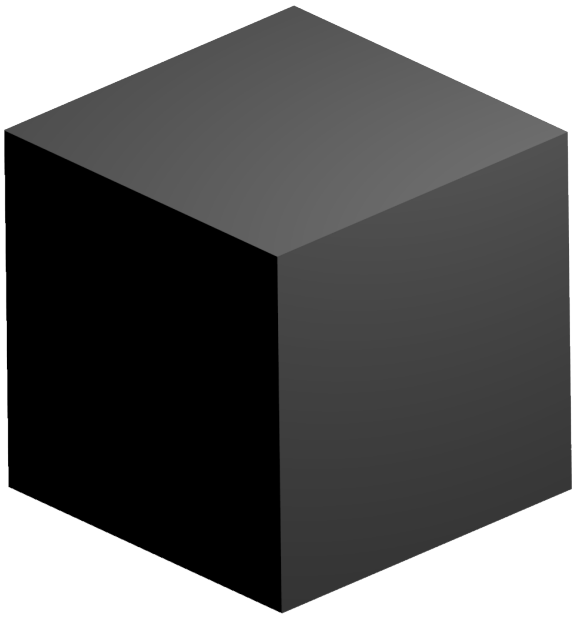


Figura 5.11: Proyección en Ortogonal

En el diseño del reproductor se pensó que éste debería utilizar este tipo de proyección porque facilitaría proyectar objetos 2D, como son las imágenes, vídeos y capturas. Esto limita el reproductor en cierto sentido, ya que al utilizar este tipo de proyección no se pueden representar objetos 3D como tales (se pierde la sensación de profundidad). Como se puede ver la figura 5.12, el cubo que se mostraba a modo de figura en dos dimensiones en la figura anterior, en verdad es un objeto 3D. Para poder ver el objeto como en verdad es, sería necesario una *proyección en perspectiva* la cual da esa sensación de profundidad que no se obtiene del uso de una proyección en ortogonal.

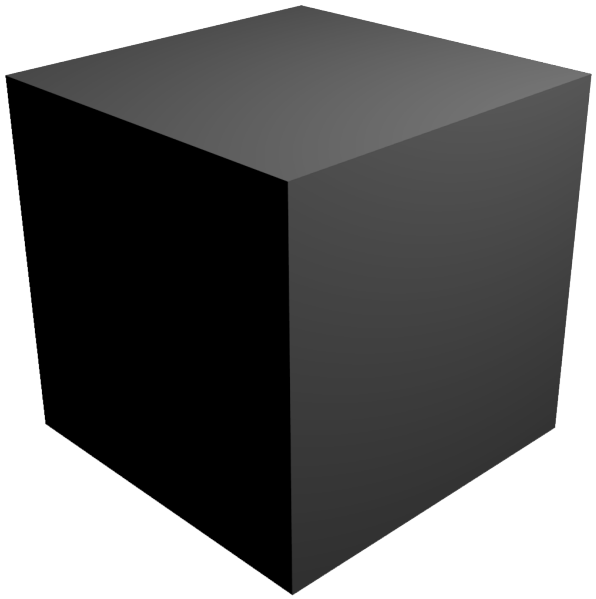


Figura 5.12: Proyección en Perspectiva

Cuando una diapositiva es renderizada en el reproductor, ésta se renderiza por capas. Estas capas son aplastadas por orden sobre el plano de proyección. Las más cercanas a la cámara se superpondrán sobre las que estén más alejadas de la misma, etc.

El orden en el que el proceso de renderizado por capas se lleva a cabo es el siguiente:

- En primer lugar se renderiza el **fondo**. El fondo, como ya se comentó en secciones anteriores, puede ser una imagen, un vídeo o una captura. Éste se renderiza a pantalla completa.

- En segundo lugar se renderizan el **resto de objetos de una diapositiva**. Estos objetos también pueden ser imágenes, vídeos o capturas, siendo éstos de menor tamaño y ocupando una posición determinada en pantalla.

- En último lugar se renderiza la **imagen de cabecera** (si así el usuario lo viera oportuno). Este objeto también se renderiza a pantalla completa y se superpone al resto de la presentación, por lo que si el objeto de cabecera no consta de transparencia, todo lo que

está por debajo no se verá. Se decidió que esto fuera así porque no sólo permite mostrar objetos de cabecera, si no que (por ejemplo) también permite poner marcas de agua.

Figura 5.13: Ejemplo de renderizado por capas

Para esta aplicación, todas y cada una de las diapositivas se renderizan de la forma mencionada anteriormente, por lo que si se cambiara el tipo de proyección a una proyección perpendicular con un cierto ángulo para la cámara, las diapositivas se verían de forma similar a como se puede ver en la figura 5.13

El reproductor también permite **renderizar estados especiales**. Estos estados pueden ser: **pantalla en blanco**, **pantalla en negro** y **modo mosaico**. Los estados de *pantalla en blanco* y *pantalla en negro* son muy sencillos de renderizar, ya que lo que se hace en este caso es cambiar el color de fondo a blanco o negro, según corresponda, cada vez que se pinta en pantalla. Por defecto, el fondo siempre se renderiza de color **negro**. Este fondo no se debe confundir con el fondo de una diapositiva, ya que los objetos de la diapositiva se renderizan sobre el segundo fondo y éste sobre el primero.

El *modo mosaico* es algo más complejo que los otros dos, ya que permite poner en pantalla un número determinado de diapositivas, las cuales hay que redistribuir de una forma determinada. En el diseño del reproductor se pensó que debería existir una forma de acceder a la última dispositiva sin tener que pasar por todas las diapositivas que separan la diapositiva actual con la mencionada. Por este motivo se creó este modo, el cual permite al usuario acceder a cualquier diapositiva de forma rápida y sencilla.

Los estados especiales son gestionados por una clase llamada **SpecialStateManager**. Dicha clase hace las veces de gestor y actua de forma similar al gestor de transiciones.

Cuando se produce un evento de teclado y éste corresponde con una tecla asignada a un estado especial, el reproductor cambia su estado de *SSLIDE* a *SSPECIAL* cediendo el control a *SpecialStateManager*. Una vez este gestor tiene el control, carga el estado especial y lo renderiza iteración tras iterarión hasta que se vuelve a producir un evento de teclado que corresponda con otro estado especial. Si dicho evento corresponde con el estado especial actual, el gestor devuelve el control al reproductor y cambia el estado de *SSPECIAL* a *SSLIDE*.

El *modo mosaico* puede mostrar una matriz de miniaturas con un tamaño variable, el cual dependerá del tamaño de la presentación. Cada una de estas miniaturas corresponde con el fondo de una de las diapositivas. Si el fondo es un vídeo o una captura, la obtención de las miniaturas se realiza de manera distinta a como se generan las miniaturas de diapositivas cuyo fondo es una imagen. Todas las miniaturas se generan una única vez y se almacenan en una subcarpeta de la carpeta que contiene las imágenes de la presentación. Dichas diapositivas se generan cuando la presentación arranca y esto sólo ocurre una única vez a no ser que se borre la carpeta de las miniaturas para futuras presentaciones. Si alguna de las diapositivas cambia o se añade una nueva diapositiva a la presentación, la función encargada de generar las diapositivas actualizará la carpeta con las miniaturas añadiendo la nueva o cambiando la modificada, pero no generando de nuevo las que ya existen. Hay que tener en cuenta que la generación de miniaturas ralentiza el inicio de la presentación, cosa que por lo general sólo ocurre una vez.

La función encargada de generar las miniaturas de las diapositivas es la función **generate_miniatures**, una función de utilidad de la clase principal que se apoya en **Imagemagick** para conseguir este objetivo. Esta función requiere de la ruta a la carpeta con las imágenes de la presentación, el alto y el ancho del escritorio y el vector con las diapositivas. Una vez la función tiene lo que necesita, lo primero es comprobar si existe la carpeta de miniaturas como subcarpeta de la carpeta de imágenes, creándose dicha carpeta en caso de no existir. El segundo paso es comprobar si las miniaturas correspondientes a cada una de las diapositivas existen en dicha carpeta. Si no existen, éstas se deben crear basándose en la imagen de fondo de la diapositiva. El problema está en que no todas las diapositivas de una presentación tienen porque tener por fondo una imagen, algunas de ellas podrían tener un vídeo o incluso una captura. Por tanto, a la hora de generar la miniatura, se comprueba que tipo de fondo tiene y, en función al resultado, se genera un tipo de miniatura u otra:

- En caso de **imágenes** de fondo, la generación de la miniatura es sencilla. La función llama a un comando de *Imagemagick* el cual escala la imagen a un tamaño menor, almacenando el resultado en la carpeta de miniaturas anteriormente creada.
- En caso de **vídeos** de fondo, la generación de la miniatura requiere que *Imagemagick* obtenga el primer frame del vídeo y éste lo pase a una imagen, la cual es almacenada en la carpeta de miniaturas. Para que esto ocurra sin problemas, hace falta tener instalado

FFMPEG, ya que Imagemagick hace uso de esta herramienta para obtener el primer frame del vídeo.

- En caso de **capturas** de fondo, la generación de una miniatura es más compleja ya que la tubería de reproducción de la captura aún no está creada y no se puede generar una representación del fondo. Por este motivo, se vuelve a utilizar *Imagemagick* para generar una imagen donde aparece un texto representativo indicando que dicha miniatura representa una captura.

Una vez obtenidas todas las miniaturas, éstas son asignadas a su correspondiente diapositiva. Las miniaturas son objetos del tipo **Image**, el cual hereda de **SlideObject**. Este tipo de objeto se utiliza para representar objetos de una diapositiva. Aun no siendo las miniaturas objetos para mostrar en una diapositiva, se decidió en el diseño que no sería correcto implementar dos clases casi idénticas cuando ambas se iban a utilizar para lo mismo (representar imágenes por pantalla). Por tanto, al utilizar objetos de la clase *Image* para representar dichas miniaturas, éstos deben ser inicializados como cualquier objeto que hereda de *SlideObject*, recayendo dicha tarea en el gestor de estados especiales. Cuando éstos objetos terminan de ser inicializados, el gestor los distribuye por la pantalla dependiendo del tamaño de la presentación.

- Si el tamaño de la presentación es inferior o igual a **30**, la matriz de miniaturas se pagina, mostrando matrices de **3x3**.

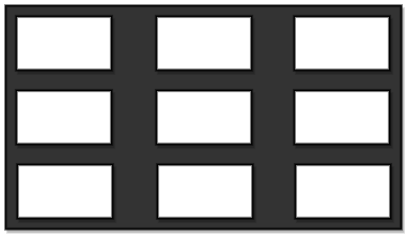


Figura 5.14: Distribución de miniaturas en una matriz de 3x3

- Si el tamaño de la presentación es mayor a **30** y menor o igual a **60**, la matriz de miniaturas se vuelve a paginar, mostrando una matriz de **5x5**.

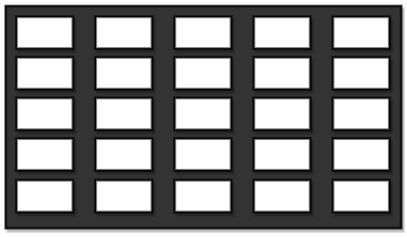


Figura 5.15: Distribución de miniaturas en una matriz de 5x5

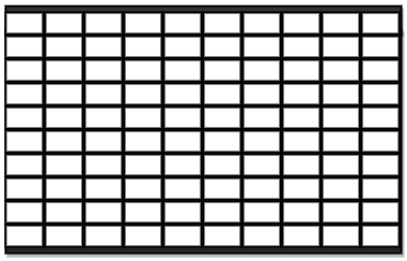


Figura 5.16: Distribución de miniaturas en una matriz de 10x10

- Si el tamaño de la presentación es superior a **60**, la matriz aumenta a **10x10**.

El objetivo de que el tamaño de la matriz aumente no es otro que reducir el tamaño de las miniaturas, con el fin de poder mostrar más miniaturas utilizando el mismo espacio. De esta forma, en presentaciones grandes, el usuario no le resulta tan difícil buscar una diapositiva entre las posibles paginaciones que pueda hacer este modo.

5.2.4 Módulo de eventos

Este módulo es el encargado de recoger todos aquellos eventos que puedan ser producidos por el usuario, bien sean pulsaciones de teclas o eventos de ventana, con el fin de tratarlos e interpretarlos y así realizar una u otra operación sobre la aplicación o las diapositivas.

Listado 5.7: Captura de eventos locales

```
1  void MainLoop::handle_local_events()
2  {
3    SDL_Event e;
4
5    while (SDL_PollEvent(&e) != 0)
6    {
7      switch (e.type)
8      {
9        case SDL_QUIT:
10         _quit = true;
11         break;
12       case SDL_KEYDOWN:
13         _player->check_key_down_event(e.key);
14         break;
15       case SDL_WINDOWEVENT:
16         _player->check_window_event(e.window);
17         break;
18       default:
19         break;
20     }
21   }
22 }
```

Los eventos son recibidos en primera instancia por el bucle principal de ejecución (**MainLoop**), el cual busca aquellos eventos que hayan podido suceder entre iteraciones. Esto lo consigue apoyándose en una de sus funciones, **handle_local_events**, la cual es llamada al principio de cada iteración del bucle. Esta función divide los eventos en *eventos producidos por pulsación de teclas* y *eventos producidos por interacción con la ventana*. Dependiendo del tipo de evento, se pasa a una función u otra del reproductor, con el fin de interpretar correctamente el evento en cuestion. También tiene en cuenta el evento de salida, **SDL_QUIT**, el cual es producido al cerrar la ventana de la aplicación. Si este evento ocurre, el flag que determina que el bucle principal siga iterando pasa a ser *True*, lo que hace que el bucle pare de iterar y la ejecución de la aplicación termine. Este evento se trata en esta clase porque al reproductor no le es útil, pero en cambio a esta clase le permite saber si continuar o no la ejecución del bucle de la aplicación.

Eventos de teclado

Los eventos de teclado son interpretados por la función **check_key_down_event** de la clase **SlidesPlayer**. Por defecto, la aplicación consta de una serie de atajos de teclado que permiten al usuario realizar diferentes acciones en la aplicación. Estos atajos se pueden encontrar en ***Atajos de Teclado*** (*Anexo C*), junto a una definición de la acción que realizan sobre la aplicación.

Los eventos de teclado pueden ser interpretados de distinta manera dependiendo de si la diapositiva actual tiene una o varias capturas, o de si el usuario se encuentra en un estado especial. Esto se debe a que cuando el usuario se encuentra ante una diapositiva con un objeto del tipo **Capture**, éste puede tener asignadas algunas teclas para ser redirigidas al programa que se captura, por lo que antes de interpretarlas, el reproductor debe preguntar a la diapositiva actual si tiene objetos del tipo *Capture* y si, en caso afirmativo, alguna de ellas redirige la tecla obtenida del evento producido por el usuario. En caso de que el usuario se encuentre en un *estado especial*, dependiendo del tipo de modo que el estado especial actual represente, se interpretarán de una manera u otra los eventos de teclado:

- Si el estado especial actual representa el *Modo pantalla en blanco* o en el *Modo pantalla en negro*, las únicas teclas que son interpretadas son **W** y **B**, ya que estas teclas permiten alternar de un modo a otro, o volver a la vista de diapositiva.

- En cambio, si el estado especial actual representa el *Modo mosaico*, las teclas que pueden ser interpretadas son **Enter** y **las teclas de dirección**. Éstas permitirán al usuario moverse entre las miniaturas del modo mosaico y seleccionar la diapositiva buscada pulsando *Enter*.

Estos modos se diseñaron de esta manera con el fin de que cualquier evento de teclado que no tuviera nada que ver con el funcionamiento del modo fuera descartado. Si no se da ninguno de estos dos casos (diapositiva con capturas o estado especial), entonces el evento de teclado será interpretado en base a las teclas predefinidas por la aplicación. Si aún así la tecla no se puede asociar a ninguna acción, ésta será descartada.

Eventos de ventana

Listado 5.8: Función para la interpretación de eventos de ventana

```
1  void SlidesPlayer::check_window_event (SDL_WindowEvent e)
2  {
3    switch (e.event)
4    {
5      case SDL_WINDOWEVENT_RESIZED:
6        resize_window(e.data1, e.data2);
7        break;
8      case SDL_WINDOWEVENT_FOCUS_LOST:
9        pause_media();
10       unfullscreen();
11       break;
12     case SDL_WINDOWEVENT_FOCUS_GAINED:
13       start_media();
14       break;
15     case SDL_WINDOWEVENT_MAXIMIZED:
16       _maximized = true;
17       break;
18     default:
19       break;
20   }
21 }
```

Los eventos de ventana permiten a la aplicación identificar pérdidas de foco, cambios en las dimensiones de la ventana, paso de pantalla completa a ventana maximizada o viceversa, etc. Gracias a que la aplicación puede interpretar estos eventos, ésta puede optimizarse en ciertos sentidos. Por ejemplo, cuando la aplicación pierde el foco, ésta puede estar minimizada o en segundo plano, por lo que el media que la diapositiva pueda estar reproduciendo es innecesario. Por ello, este media es pausado y puesto en marcha de nuevo cuando la aplicación vuelve a tener el foco del sistema. También permite solucionar bugs que presenta la versión utilizada de la librería de **SDL2**.

Al utilizar Alt+Tab teniendo la aplicación en modo pantalla completa, en algunos gestores de ventanas esto implicaba que la aplicación se congelara y no hubiera forma de parar la aplicación, aun matando el proceso. Por ello, al capturar el evento de pérdida de foco, la ventana de la aplicación se puede minimizar y así evitar que este conocido bug ocurra. Este bug es un error que sólo ocurre al usar determinadas distribuciones linux que no acaban de

ser compatibles con la nueva funcionalidad ofrecida por SDL.

Otros eventos de ventana permiten saber si las dimensiones de la misma cambian. Esto es muy útil a la hora de redimensionar la diapositiva que se está mostrando en un momento dado o la distribución de las miniaturas del *modo mosaico*. Gracias a esto, *pCROM* puede presumir de no dar problemas al utilizar diferentes tipos de resolución en una presentación, ya que al pasar a pantalla completa se llevan a cabo una serie de reajustes y redimensiones que colocan de forma correcta en pantalla el contenido actual que se esté mostrando en la presentación.

5.2.5 Módulo multimedia

Este módulo lo forman todas aquellas clases que representan el media en la aplicación. Como ya se dijo en módulos anteriores, en una diapositiva se pueden tener **imágenes**, **vídeos**, **capturas**, **música** o **efectos de sonido**. Cada uno de estos tipos es representado en código mediante una clase, la cual hereda de **SlideObject** si es un objeto que puede ser renderizado en pantalla, o de **Sound** si es un objeto que representa algún tipo de sonido. A continuación, se hará una introducción a cada una de estas clases, cómo se renderizan o reproducen en la aplicación y qué métodos utilizan para que todo esto ocurra sin problema.

Objetos renderizables

Los objetos renderizables en la aplicación son aquellos objetos que heredan de la clase **SlideObject**. Esta clase es una *clase abstracta*, la cual posee una serie de variables y funciones que deben ser comunes para todo objeto susceptible de ser dibujado en pantalla. Al utilizar funciones virtuales, se consigue que la clase que herede de *SlideObject* tenga que implementar dichas funciones si no se quiere que la clase en cuestión pase a ser una clase abstracta. Con esto se busca utilizar uno de los conceptos más importantes de la programación orientada a objetos, el **polimorfismo**.

Gracias a la potencia que el *polimorfismo* aporta, por ejemplo, la clase **Slide** posee un único vector de *SlideObject* en vez de poseer un vector por cada uno de los tipos de clases que heredan de la misma (imágenes, vídeos y capturas).

Las funciones que sobrescriben las clases que heredan de *SlideObject* son: **render** y **resize**. Estas dos funciones permiten al reproductor poder dibujar el objeto en cuestión en pantalla y reajustarlo dependiendo de las dimensiones de la ventana de la aplicación.

Las **Imágenes** son representadas en código mediante la clase **Image**, la cual hereda de SlideObject. La textura de la imagen se obtiene gracias a la función **load_texture**. Esto lo consigue utilizando la librería **SDL_Image**[1], la cual carga dicha imagen como un objeto

[1] https://www.libsdl.org/projects/SDL_image

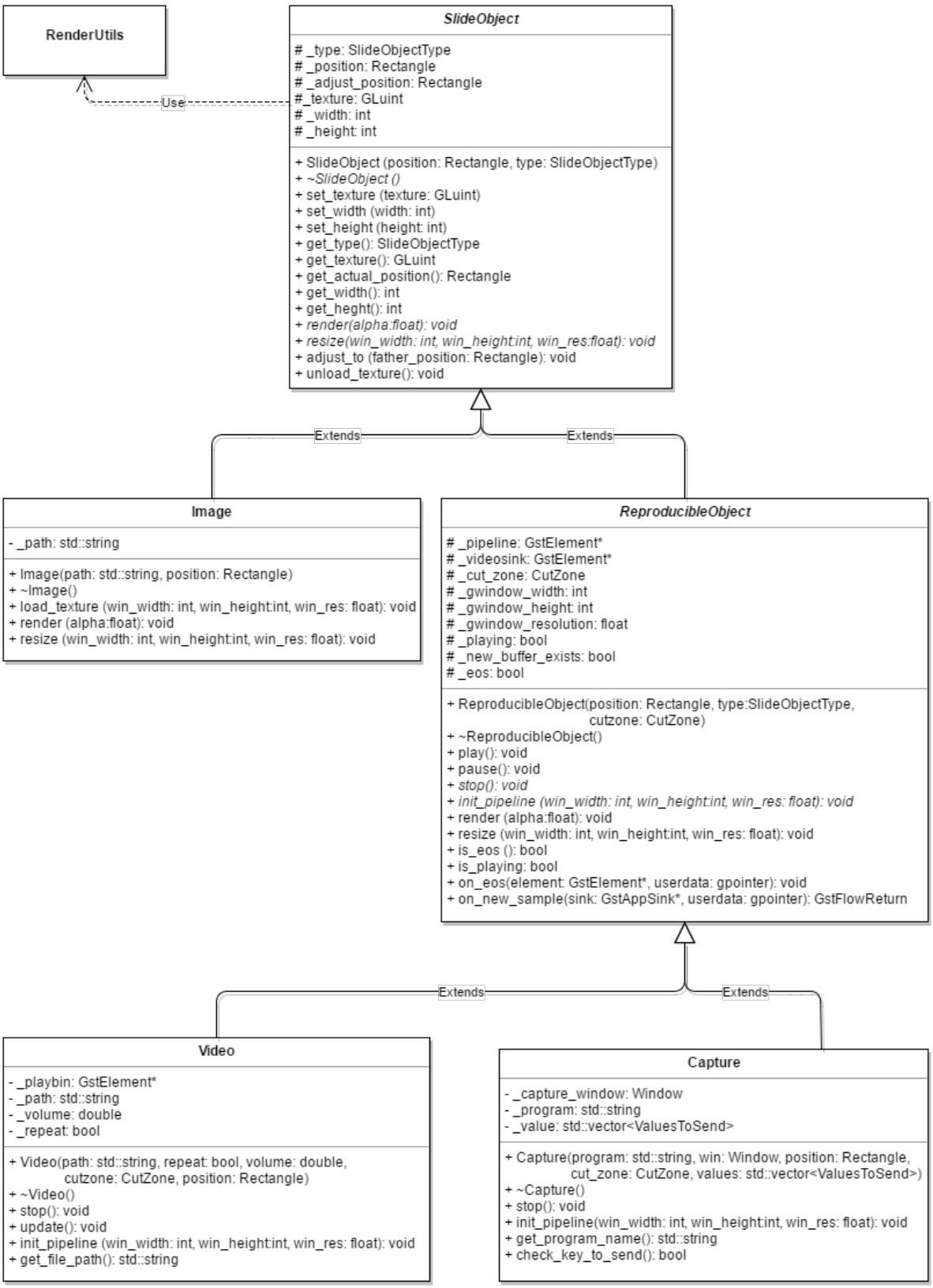


Figura 5.17: Diagrama de clases *SlideObjects*.

Surface de **SDL**, para luego obtener gracias a dicho objeto el ancho, el alto y los píxeles asociados a la imagen. Al utilizar esta librería para la carga de imágenes, la aplicación se ve beneficiada en el sentido de que permite aceptar un amplio conjunto de formatos de imagen. En concreto, los formatos de imagen que la aplicación acepta son: **BMP**, **GIF**, **JPEG**, **LBM**, **PCX**, **PNG**, **PNM**, **TGA**, **TIFF**, **WEBP**, **XCF**, **XPM** y **XV**.

Listado 5.9: Obtención de la textura de una imagen.

```
1  void Image::load_texture (int window_width, int window_height,
2    float window_resolution)
3  {
4    ...
5
6    if (surface)
7    {
8      /* Obtencion de la textura */
9      glGenTextures(1, &_texture);
10     glBindTexture(GL_TEXTURE_2D, _texture);
11
12     if (surface->format->BytesPerPixel == 4)
13     {
14       mode = GL_RGBA;
15     }
16
17     _width = surface->w;
18     _height = surface->h;
19
20     glTexImage2D(GL_TEXTURE_2D, 0, mode, _width, _height,
21       0, mode, GL_UNSIGNED_BYTE, surface->pixels);
22
23     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
24     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
25     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
26     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
27
28     SDL_FreeSurface(surface);
29   }
30
31   ...
32 }
```

Una vez la función está en posesión de los píxeles asociados a la imagen, éstos se cargan como una textura de OpenGL, la cual será utilizada para mostrar la imagen en pantalla. Para poder mostrar una imagen por pantalla, es necesario llamar a su función **render**. Ésta a su vez hará una llamada a la función **render_texture_quad**, la cual es una de las funciones de utilidad implementadas para facilitar la tarea de renderizado en la aplicación. Este conjunto de funciones auxiliares son utilizadas en toda la aplicación a la hora de mostrar texturas por pantalla o reajustar las posiciones en las que éstas se muestran. Este reajuste viene de la mano de dos funciones, **add_letterbox** y **add_pillarbox**, las cuales adaptan la textura de un objeto renderizable a la zona donde el usuario decidió mostrar dicho objeto. Cada vez que se genera

una textura nueva o las dimensiones de la ventana de la aplicación cambian, la aplicación reajusta el contenido de la presentación. Esto lo consigue llamando a la función **resize** de los objetos que heredan de SlideObject. Este reajuste comprueba si la resolución es mayor o menor a la resolución actual a la cual se está mostrando la presentación. Dependiendo del resultado, se realiza un tipo de reajuste u otro:

- *Resolución de la textura **mayor** a la resolución de la aplicación*. En este caso se llama a la función **add_letterbox**. Esta función reajusta añadiendo filas transparentes arriba y abajo de la textura, con el fin de conseguir que ésta se pueda mostrar en una resolución inferior a la suya propia. Estas filas se generan al redimendionar la textura en altura, por lo que no la recortan en ningún sentido. El resultado obtenido es similar al mostrado en la figura 5.18, pero en este caso las filas añadidas son de color negro porque detrás no hay nada más, por lo que se muestra el fondo negro de la aplicación.

Figura 5.18: Ejemplo de uso de **letterbox**.

  El resultado del reajuste no se guarda en la variable asignada por el usuario para la posición del objeto, sino que se guarda en una variable diferente, la cual será utilizada para mostrarlo. El utilizar una variable diferente tiene por objetivo mantener en memoria la posición inicial que el usuario asignó al objeto en pantalla para futuros reajustes.

- *Resolución de la textura **menor** a la resolución de la aplicación*. En este caso se llama a función **add_pillarbox**. Esta función reajusta añadiendo columnas transparentes a la derecha e izquierda de la textura, con el fin de conseguir que ésta se pueda mostrar en una resolución superior a la suya propia. Estas columnas se generan al redimensionar la textura en anchura.

Figura 5.19: Ejemplo de uso de **pillarbox**.

Al igual que con el ejemplo de *letterbox*, las columnas añadidas a la imagen son de color negro porque detrás de la imagen no hay nada más, sólo el fondo de la aplicación.

- *Resolución de la textura **igual** a la resolución de la aplicación*. En este caso no se produce reajuste alguno, simplemente se utiliza tal cual la posición asignada por el usuario para mostrar el objeto.

El uso de estas funciones facilita mucho el despliegue de presentaciones en diferentes tipos de resolución, ya que gracias a que ajustan el contenido de la presentación a la resolución que la pantalla o el proyector le permite, el contenido siempre se muestra como el usuario lo definió en el archivo CROM de la presentación.

En el caso de **Vídeos** y **Capturas**, renderizar este tipo de objetos tiene algo más de complejidad, ya que para poder obtener los píxeles asociados a una muestra del vídeo o captura, es necesario crear una tubería de **Gstreamer**. Las tuberías en Gstreamer permiten unir módulos de decodificación y transformación de píxeles con el fin de hacer trivial algo como puede ser la decodificación de vídeo, cosa que de forma manual no lo es en absoluto.

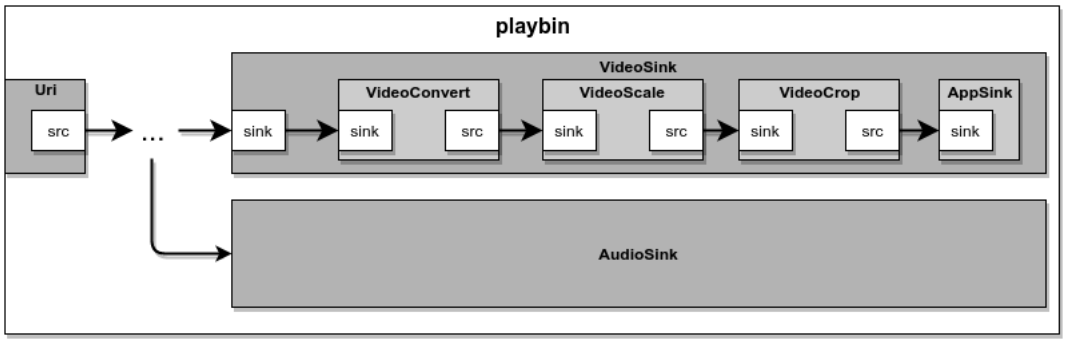


Figura 5.20: Estructura de la tubería de *vídeo*.

La tubería empleada para la decodificación de vídeo sigue la estructura de **playbin**, una tubería ya implementada por GStreamer para este objetivo. A esta tubería se le cambia el *sink* que tiene para vídeo, utilizando uno propio creado con el fin de tener acceso y poder trabajar con los *samples* (**muestras** en español) resultantes de la reproducción del vídeo. La inicialización de la tubería y el cambio del *sink* se lleva a cabo en la función **init_pipeline** de la clase Video. Al utilizar GStreamer para la decodificación de vídeo, la aplicación puede mostrar casi cualquier vídeo por pantalla, sea cual sea su formato.

En este caso, el *sink* del vídeo de esta tubería está formado por los siguientes módulos:

- **VideoConvert**: Este módulo permite convertir los fotogramas de vídeo de entre una gran variedad de formatos de vídeo.
- **VideoScale**: Este módulo reajusta el tamaño de los fotogramas de vídeo que le llegan, comprobando en una primera instancia si ese reajuste es necesario o no.
- **VideoCrop**: Este módulo permite recortar los fotogramas de vídeo que recibe, permitiendo eliminar parte de los mismos de la parte superior, inferior, derecha o izquierda. En caso de no poder recortar, el fotograma de salida es el mismo que el de entrada.
- **AppSink**: Este módulo permite obtener los fotogramas que viajan por la tubería al final de la misma, con el fin de que la aplicación pueda trabajar con ellos.

En el caso de la tubería utilizada para la captura de programas, ésta se ha creado siguiendo una estructura totalmente distinta. Para empezar, ya no se parte de *playbin* para crear dicha tubería, si no que se crea desde cero. Se decidió que esto fuera así debido a que *playbin* utiliza un sink para decodificar audio, y en este caso no era necesario.

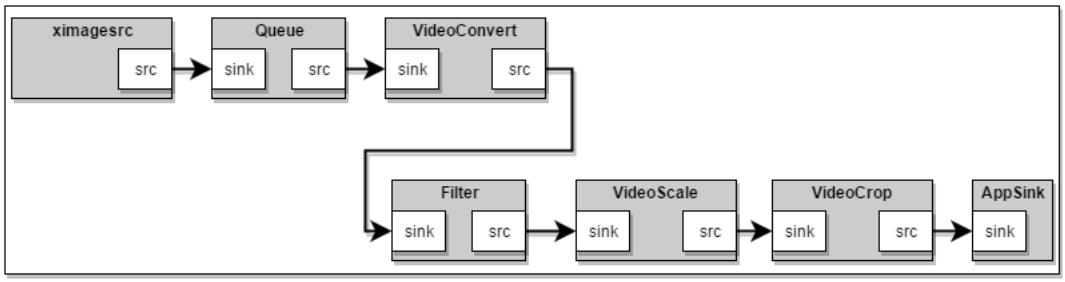


Figura 5.21: Estructura de la tubería de *captura*.

En este caso, la tubería en su totalidad está formada por los siguientes módulos. Algunos de ellos son los mismos que se han utilizado por la tubería para los vídeos, por lo que no se describirán:

- **XImageSRC**: Este módulo permite capturar los píxeles asociados a la ventana de un determinado programa.

- **Queue**: Este módulo almacena fotogramas con el objetivo de no sobrecargar la tubería.

- **Filter**: Este módulo aplica un filtro de color a los fotogramas que circulan por la tubería. En este caso, el filtro pasa el formato de color de los fotogramas a RGB.

El problema de capturar un programa en ordenadores que no son lo sucifientemente potentes, es que se pierden muestras/fotogramas. Para solventar este problema, se ha añadido una cola entre el origen de la captura y el módulo de *VideoConvert*, consiguiendo de esta manera almacenar aquellas muestras que serían descartadas si el ordenador en cuestión no tuviera capacidad para capturar correctamente un programa en tiempo real. Al igual que la tubería de vídeo, ésta se crea e inicializa en la función **init_pipeline**.

En cuanto a su dibujado en pantalla, ambos objetos se renderizan gracias a la misma función con lo que lo hacen los objetos del tipo **Image**, utilizando además el mismo tipo de reajuste. La diferencia en este caso es que el contenido de la textura, una vez creada, se actualiza por cada nueva muestra, consiguiendo de esta forma liberar a la aplicación de generar y borrar texturas de OpenGL constantemente.

Objetos de audio

Los objetos de audio deben tratarse de forma distinta a los objetos renderizables por el mero hecho de que estos no se pueden mostrar por pantalla, lo que no quita que no puedan formar parte de una diapositiva.

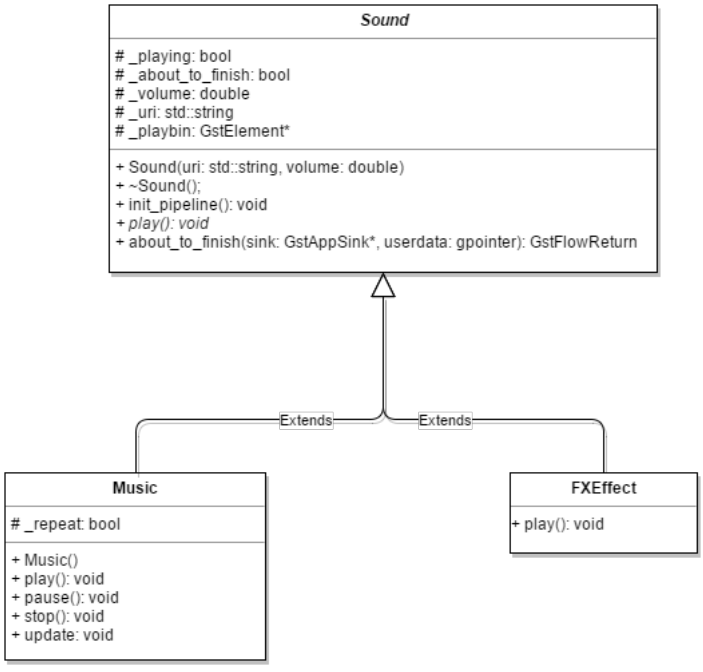


Figura 5.22: Diagrama de clases *Sound*.

En este caso, los objetos de audio son aquellos que heredan de la clase **Sound**. Esta clase, al igual que **SlideObjet**, es una clase abstracta que hace uso de una tubería ya creada por Gstreamer para la reproducción de audio. La función virtual que sobrescriben **Music** y **FXEffect** es **play**, ya que la estructura de la tubería es la misma para las dos clases. La diferencia radica en que los efectos de sonido, *FXEffect*, se deben reproducir una única vez, mientras que la música se debe poder pausar, detener o volver a reproducir. Se podría haber utilizado *Music* para reproducir tanto música como efectos de sonido, pero se decidió que de esta manera se distinguiría mejor el uso de cada tipo de sonido y no se añadiría más complejidad a la clase *Music* de la que debe tener. Al igual que pasa con los vídeos, el uso de una tubería de GStreamer para la decodificación y reproducción de audio permite a la aplicación reproducir casi cualquier formato de audio.

5.3 APLICACIÓN qCROM

qCROM es una aplicación auxiliar creada para que los usuarios de la plataforma puedan aprovechar el canal de retorno del público asistente realizando pequeños cuestionarios. Éstos formarían parte de la presentación como una diapositiva más y permitirán que el público pueda participar activamente en la misma. Esto no sólo permite aprovechar ese canal de retorno con el público, si no que también permite ofrecer una interacción bidireccional con el mismo gamificando parte de las presentaciones.

Esta aplicación funciona en base a *sesiones*. Cada una de estas sesiones se corresponde con un cuestionario en concreto, almacenando su estado, la cantidad de usuarios y respuestas, las puntuaciones en un momento dado, etc.

La aplicación se puede dividir en:

- *Frontend*: Es la parte de la aplicación encargada de registrar los usuarios de entre el público que quieran participar en el cuestionario, recibir los eventos de teclado del ponente y mostrar las preguntas del cuestionario, así como las puntuaciones y el número de respuestas en tiempo real.
- *Backend*: Es la parte encargada de generar una nueva sesión, de llevar el control del cuestionario asociado a dicha sesión, de actualizar la puntuación de los usuarios, de interpretar los eventos recibidos, etc.

El *frontend* se puede alojar en servidores públicos o privados, así como localmente. En cambio, la parte que solamente se puede ejecutar localmente es el *backend*, y éste necesitará saber dónde se encuentra alojada la parte de *frontend*. De esta forma, el *frontend* se puede alojar en un servidor público y dar funcionalidad a varios *backend*. La limitación actual de la

aplicación está en que el *frontend* sólo atiende a la vez peticiones referentes a una sesión en concreto. Hasta que esta sesión no termine, el *frontend* descartará las peticiones que no sean de la sesión actual. El *frontend* podría modificarse para que atendiera varias peticiones a la vez de diferentes *backends*, pero ese no era un objetivo a cumplir por esta plataforma.

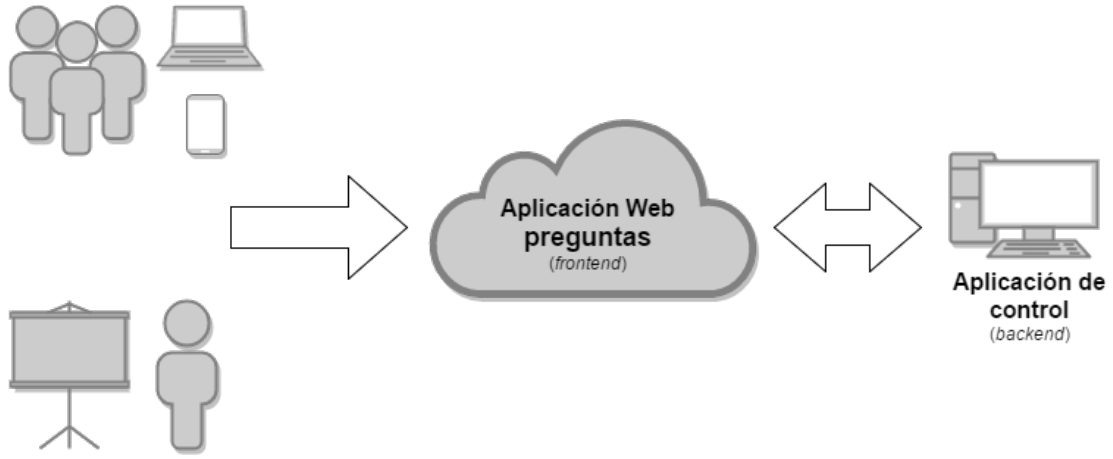


Figura 5.23: Esquema de flujo de comunicación en la aplicación *qCROM*.

5.3.1 Aplicación web para las preguntas

Esta aplicación representa el *frontend* de *qCROM*. Para su implementación se han utilizado **PHP**, **HTML5** y **CSS3**. Toda la lógica de la aplicación está implementada con PHP, quedando HTML5 y CSS3 para la parte de la interfaz de usuario.

Esta aplicación, como ya se dijo anteriormente, atiende peticiones del *backend* asociado a una determinada sesión. Cuando una nueva sesión se inicia, se guarda en caché el identificador de la sesión junto al título del cuestionario. Esta caché sigue un diseño propio, creándose desde cero para la aplicación de preguntas. Uno de los requisitos de la aplicación *qCROM* era utilizar el menor número de dependencias posible para que éste pudiera alojarse en servidores públicos de bajo coste, por lo que si se hubiera utilizado una caché implementada por terceros, este objetivo no se hubiese cumplido del todo al requerir de una dependencia más. Al implementar una caché desde cero, sin utilizar ninguna dependencia adicional, si que se consigue cumplir este objetivo.

La aplicación distingue entre usuarios corrientes y usuarios con el rol de **administrador**. Dependiendo de esto, la aplicación carga un contenido u otro, por lo que la vista de los usuarios del tipo administrador es diferente a la vista que se muestra a los usuarios normales, ya que la de los usuarios está diseñada para mostrar botones y los campos necesarios para que el usuario pueda responder a las preguntas, mientras que la de administrador está diseñada para poder controlar la presentación y para mostrar a la vez las preguntas, los resultados, etc. Estas vistas se recargan una vez por segundo realizando peticiones *AJAX* a un determinado

archivo PHP. Dependiendo del tipo de usuario y del estado actual del cuestionario, este archivo PHP genera una vista u otra. Si el resultado de la petición difiere del resultado de la petición anterior, el contenido de la última respuesta será cargado en el *div* general de la vista.

La aplicación web hace uso de una *caché* y de *cookies* para poder funcionar sin problemas. A continuación, se describirán como funciona cada una de estas dos cosas y su utilidad en la aplicación.

Gestor de la caché

Figura 5.24: Estructura de la caché de la aplicación web de *qCROM*.

Esta caché está representada como una carpeta que se crea automáticamente en el servidor y que se limpia una vez se inicia o termina una sesión.

La carpeta de la caché se gestiona gracias a una clase creada en PHP, la cual recibe el nombre de **CacheMgr**. Este gestor permite añadir, eliminar o actualizar elementos de la caché, los cuales no dejan de ser archivos de texto con un cierto contenido alojados en alguna de las subcarpetas de la carpeta que representa la caché. Estas secciones/subcarpetas son:

- **Global**: En esta subcarpeta se guardan elementos genéricos que afectan a todo el cuestionario (identificador de sesión, identificador de la pregunta actual, etc).
- **Usuarios**: En esta subcarpeta se añaden tantos elementos como usuarios participen en un cuestionario. Cuando se muestra la primera pregunta, el contenido de esta carpeta es

solicitado por el *backend*. Una vez éste tiene dicho contenido, la carpeta que lo contiene es vaciada con el fin de liberar memoria del servidor.

- **Respuestas de usuario**: Esta subcarpeta almacena las respuestas de los usuarios para una determinada pregunta. Estas respuestas son solicitadas por el *backend* cada cierto tiempo, limpiando la carpeta cada vez que esto sucede.
- **Eventos**: Esta subcarpeta almacena los eventos producidos por la pulsación de determinadas teclas por el usuario que lleva el control de la presentación. Al igual que pasa con las respuestas de los usuarios, el contenido de esta carpeta es solicitado cada cierto tiempo para después eliminarlo de dicha carpeta.
- **Preguntas**: Esta subcarpeta almacena las preguntas del cuestionario, así como el número de respuestas y los resultados de cada una de ellas. El contenido de esta carpeta se va actualizando gracias al *backend*, el cual envía la nueva pregunta una vez ésta debe ser mostrada por pantalla. Al no eliminarlas, cada vez que se vuelve a una pregunta anterior, el contenido de la pregunta junto a la totalidad de las respuestas y sus resultados siguen existiendo, por lo que el *backend* no tiene que volver a enviar nada a la aplicación, ahorrando así en consumo de red por parte de la aplicación.

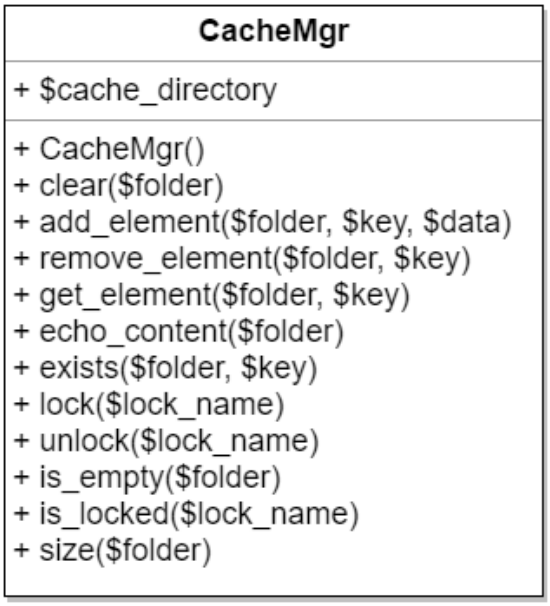


Figura 5.25: Diagrama UML de la clase *CacheMgr*.

Con el fin de no enviar al *backend* respuestas, usuarios o eventos repetidos, cada vez que se envían datos se limpian las carpetas que contienen estos archivos. Al limpiarse algunas de las carpetas de la caché, debe existir un control de acceso a dichas carpetas con el fin de que no se eliminen datos que aún no se han enviado al *backend* por ser añadidos a la caché al mismo tiempo que ésta limpia las carpetas (*condiciones de carrera*). Esto se consiguió solucionar añadiendo un *semáforo mutex* por cada una de las carpetas en las que esta situación

se daba, consiguiendo de esta forma que mientras ese semáforo mutex estuviera bloqueado se entendería como que se estaban solicitando datos a la aplicación y no se deberían añadir nuevos datos hasta que este semáforo no se liberara. La aplicación implementa una totalidad de **tres** cerrojos, uno para la carpeta de usuarios, otro para la carpeta de respuestas y otro para la carpeta de eventos. La funcionalidad para la creación y eliminación de estos cerrojos la ofrece el *gestor de la caché* mediante las funciones **lock** y **unlock**. Estos cerrojos se crean por la caché agregando ciertos elementos en la carpeta *global*. Si dicho elemento existe cuando se va a añadir algo a alguna de estas carpetas, se interpretará como que el cerrojo está bloqueado. Por el contrario, si este elemento no existe en la carpeta global, eso hará entender que el cerrojo está abierto y se pueden añadir nuevos elementos a las carpetas en cuestión.

Listado 5.10: Agregación de una nueva respuesta de usuario.

```
1 while($_cache->is_locked(ANSWERS_LOCK));
2 $_cache->add_element(CACHE_ANSWERS, $session->user_id . '_answer',
      json_encode($answer));
```

Uso de cookies

La aplicación utiliza *cookies* para el almacenamiento temporal de ciertos datos referentes a un participante del cuestionario. Estos datos dependen del tipo de usuario en cuestión, aunque comparten ciertos campos. Éstos son:

- **Tipo de usuario**, variables que puede tomar los valores de *administrador* o de *usuario corriente*.
- **Nombre de usuario**, variable cuyo valor es meramente informativo, útil a la hora de mostrar los resultados del cuestionario por pantalla.
- **Identificador único de usuario**, variable cuyo valor identifica y diferencia un usuario de otro a la hora de guardar y leer respuestas.
- **Identificador único de sesión**, variables cuyo valor identifica una determinada sesión. Utilizado por la aplicación a la hora de seleccionar respuestas recibidas de los usuarios, descartando aquellas cuyo identificador de sesión no corresponda con el del cuestionario actual.

Si el usuario no es del tipo administrador, se añade un campo **Respuestas**, el cual almacena la totalidad de las respuestas del usuario para cada una de las preguntas contestadas por el mismo. Esto permite a la aplicación generar una vista para los usuarios que muestre la respuesta seleccionada y la respuesta correcta de las preguntas que el usuario responda.

Estas cookies son **cookies de sesión**, es decir, se crean al iniciar sesión en la aplicación para un determinado cuestionario y se eliminan al terminar dicho cuestionario. Los datos referentes a la cookie no son almacenados ni cedidos para ningún propósito que no sea ayudar a la aplicación en ciertas funcionalidades. Debido a que las cookies en esta aplicación no se utilizan para los fines citados anteriormente, la ley no exige que se informe al usuario de la aplicación sobre éstas, pero se ha preferido dar una pequeña información de cómo funcionan las cookies y por qué se usan en esta aplicación con el fin de tener mejor informado al usuario final y que de esta forma conozca un poco mejor como funciona la aplicación. Esta información se puede encontrar mediante un enlace en la vista de inicio de sesión de la aplicación.

5.3.2 Aplicación de control

Esta aplicación de control representa el *backend* de *qCROM*. La totalidad de la aplicación está escrita en **Python**, ya que agilizaba tareas de implementación, como por ejemplo la parte de comunicación con los mandos auxiliares usando **sockets**. Su objetivo es iniciar una nueva sesión, llevar el control del cuestionario y parsear los archivos de preguntas y de usuarios entre otras cosas. Está formada por una totalidad de tres módulos, los cuales se describirán a continuación uno a uno, identificando las partes más importantes.

Módulo de parseo

Este módulo tiene por objetivo analizar y obtener los datos de los archivos de la aplicación. La estructura de cada uno de estos archivos se puede encontrar en el *Anexo E*. Para cada uno de estos archivos, se analiza y obtiene cada una de las lineas que contienen, descartando comentarios y lineas en blanco.

- **Archivo de configuración**: Por cada linea válida de este archivo, se comprueba si empieza por alguna de las palabras claves asociadas a este tipo de archivo. Estas son: **site**, **pass**, **quiz_file**, **users** y **refresh**. Si estas lineas no comienzan por alguna de estas palabras clave, la linea será descartada.

- **Archivo de preguntas**: Estos archivos están formados por secciones. Cada una de estas secciones se delimita con una determinada palabra clave. El archivo se analiza y se divide según estas secciones, descartando aquellas partes mal formadas. Cada una de éstas se divide a su vez en función de otra palabra clave, con el fin de catalogar cada una de las partes en función de si son respuestas, media o texto de la pregunta. Una vez obtenidos los datos, estas preguntas son almacenadas en la base de datos *sqlite* de la aplicación.

- **Archivo de usuarios**: Estos archivos, al igual que el de las preguntas, se dividen en secciones. Cada una de estas secciones representa a un usuario. Una vez en posesión de una sección, esta se divide en función de otra palabra clave con el fin de obtener el usuario y el identificador del mando auxiliar. Terminado la obtención de datos de este tipo de archivos, los usuarios son cargados en la base de datos *sqlite* de la aplicación.

Módulo de control

El módulo de control es el encargado de gestionar un determinado cuestionario. Para ello se establece un estado y en función de lo que éste represente para la aplicación, se realizará una u otra acción.

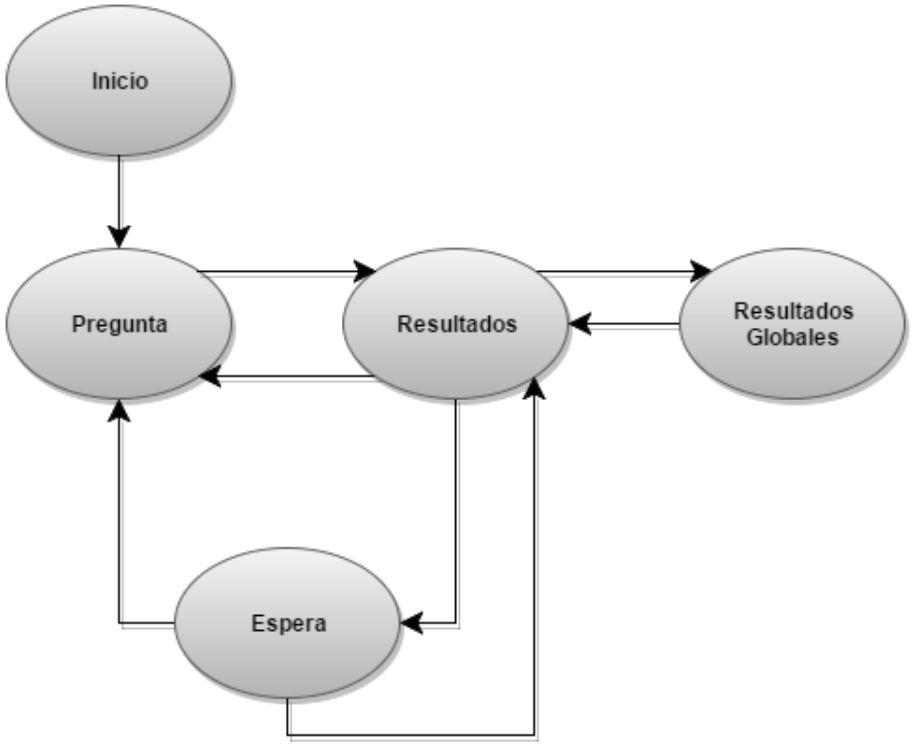


Figura 5.26: Diagrama de estados de la aplicación de control de *qCROM*.

Los posibles estados de las aplicación son los siguientes:

- **Estado de inicio**: Este es el estado inicial del cuestionario. En este estado, los usuarios del público asistente que quieran participar en el cuestionario se pueden registrar mediante el uso de la interfaz de usuario que ofrece el *frontend* para ello.

- **Estado de pregunta**: En este estado, la aplicación se encuentra mostrando una pregunta. En este modo, el *backend* solicita las respuestas de los usuarios al *frontend*, enviando en cada petición el total de respuestas para la pregunta actual en un momento dado.

- **Estado de resultados**: En este estado el *frontend* muestra la respuesta correcta de una pregunta en caso de la vista de administrador, o la respuesta correcta y la del usuario en el caso de la vista para resto de usuarios.
- **Estado de espera**: Este estado permite establecer una pausa en el cuestionario, bloqueando las vistas de administrador y del resto de usuarios con una pantalla de espera.
- **Estado de resultados globales**: A este estado sólo se puede acceder desde el *estado de resultados*. Permite mostrar las puntuaciones globales de los usuarios en la vista de administrador, no teniendo efecto en la vista para el resto de usuarios.

Estos estados cambian en función de los eventos de teclado generados por el administrador del cuestionario. Éstos son recogidos por una función en JavaScript que escucha a la espera de posibles eventos producidos por el usuario con rol de administrador. Si alguno de esos eventos no coinciden con los establecidos para la aplicación, serán descartados. Las teclas válidas y sus acciones se pueden encontrar en el *Anexo C*.

Módulo de eventos

Este módulo tiene por objetivo recibir todos aquellos eventos provenientes del *frontend* como de los mandos auxiliares que el ponente pueda ofrecer al público de una presentación.

- **Eventos del *frontend***: Estos eventos se pueden dividir en función del tipo de usuario. Si el usuario es del tipo administrador, los eventos que serán recibidos por la aplicación podrán modificar la pregunta actual o el estado del cuestionario. Estos eventos son pedidos al *frontend* cada cierto tiempo, el cual dependerá del tiempo de refresco que el usuario haya establecido para la aplicación de control en el archivo de configuración. Las peticiones que hace el *backend* al *frontend* se llevan a cabo mediante **request**. Estas peticiones permiten enviar o solicitar información a ciertos archivos PHP alojados en el servidor. Si la contraseña establecida por el usuario en el archivo de configuración no coincide con la contraseña establecida para el *frontend*, estas peticiones siempre devolveran una cadena vacía. Esto se hace así con el fin de que ninguna persona física ajena al concurso pueda interferir en el intercambio de información entre el *backend* y el *frontend*.

  Si por el contrario los eventos son producidos por el resto de usuarios de la aplicación, éstos representarán posibles respuestas a la pregunta actual, por lo que se almacenarían en la base de datos añadiendo un nuevo registro en la *tabla de respuestas* o modificando uno ya existente. En la figura 5.27 se puede observar un diagrama de secuencia de este tipo de peticiones. En este caso se representa la petición de información periódica al *frontend*, la cual dependerá del estado actual del cuestionario. Para esta petición, siempre

se devuelven los eventos de teclado producidos por el administrador del cuestionario como información, aunque dependiendo del estado actual del cuestionario además se podrá devolver otro tipo de información, como por ejemplo los nuevos usuarios o las nuevas respuestas a una pregunta. A la vez que se pide información, se aprovecha para actualizar la que el *frontend* guarda para mostrarla a los usuarios. En este caso, lo que se envía en la petición es la totalidad de respuestas actuales a una pregunta. Al igual que con la respuesta a la petición, que esto suceda dependerá de si el cuestionario se encuentra en el estado de pregunta.

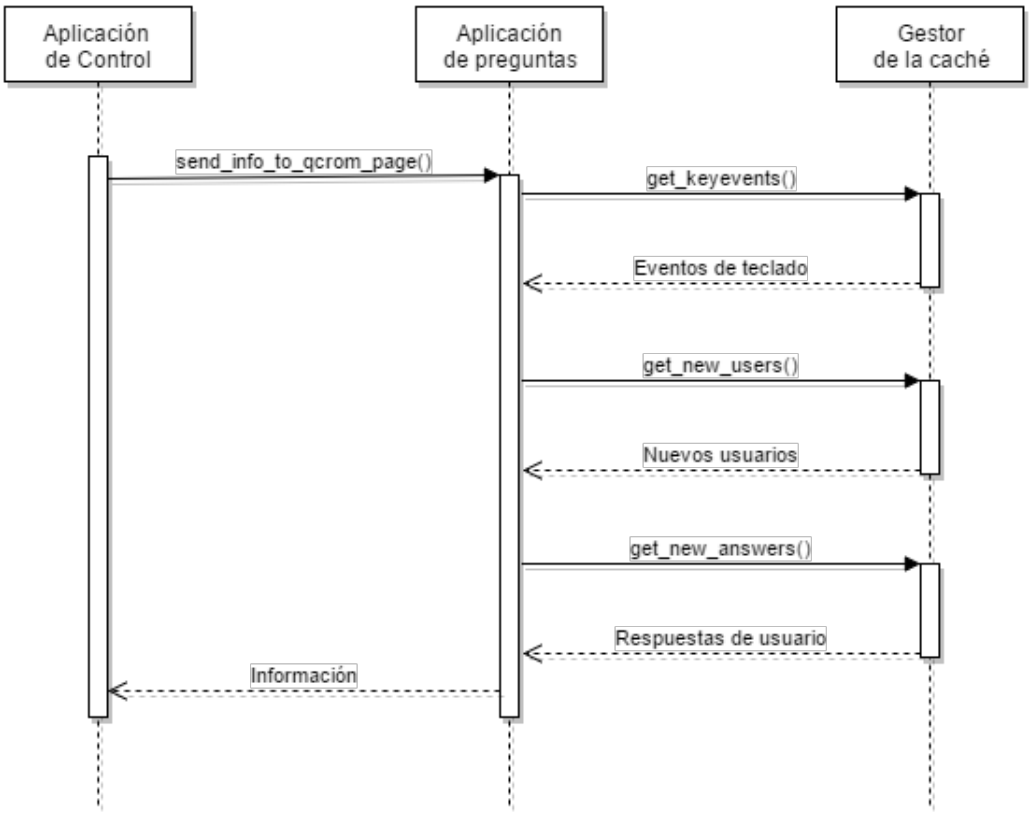


Figura 5.27: Diagrama de secuencia de una petición *request* en la aplicación.

- **Eventos de mandos auxiliares**: Cuando la aplicación de control se inicia, ésta lanza dos hilos. En el primero se lleva el control de la aplicación, y en el segundo se lanza un **SocketServer** escuchando peticiones en el puerto **10003**. Cuando un nuevo paquete de datos llega a dicho puerto, éste se decodifica y se transforma en un diccionario con **JSON**. Una vez en posesión de los datos en forma de diccionario, se obtiene el identificador y la respuesta del mando que ha enviado el paquete de datos. Si la respuesta no entra dentro del rango de las posibles respuestas, ésta será descartada. Por el contrario, si es una respuesta válida, ésta será almacenada en la tabla de respuestas junto al identificador del usuario asociado al mando auxiliar en cuestión. Para ello se realiza una consulta previa a la base de datos en busca del usuario que tiene asociado dicho identificador de mando. Si esta consulta no devuelve nada, querrá decir que el

usuario no ha sido cargado en la base de datos préviamente y que, por tanto, se debe descartar la respuesta al no participar dicho usuario en el cuestionario actual.

CAPÍTULO 6

EVOLUCIÓN Y COSTES

En este capítulo se describe la evolución que ha sufrido el presente Trabajo Fin de Grado durante su desarrollo, así como los costes asociados al mismo. Como se mencionó en el capítulo de *Método de trabajo*, este proyecto ha seguido la metodología del *Proceso Unificado de Desarrollo*, por lo que al tratarse de una metodología iterativa, la evolución que ha sufrido este proyecto se describirá por partes, definiendo y detallando cada una de las iteraciones que se han llevado a cabo durante el desarrollo. Por último, se darán algunos datos sobre el coste temporal que ha requerido el desarrollo de esta plataforma y del posible coste económico asociado al mismo.

6.1 EVOLUCIÓN

En esta sección se habla de cada una de las iteraciones por las que ha pasado el desarrollo de la plataforma, analizando en detalle cada una de las fases que han constituido dichas iteraciones.

Tabla 6.1: Fechas de entrega según iteración.

Iteración	Fecha de entrega
1	20 de Octubre de 2015
2	18 de Diciembre de 2015
3	15 de Enero de 2016
4	28 de Enero de 2016
5	5 de Febrero de 2016
6	15 de Febrero de 2016
7	10 de Marzo de 2016
8	31 de Marzo de 2016
9	20 de Abril de 2016
10	5 de Mayo de 2016
11	20 de Mayo de 2016
12	1 de Junio de 2016

6.1.1 Fase de Inicio

En esta fase se puso en marcha el desarrollo del proyecto, estableciendo los objetivos mínimos, los primeros requisitos funcionales y las fechas de entrega (Tabla 6.1). Gracias a esta fase se obtuvo la base del reproductor, la cual incluía el bucle principal de ejecución y una aproximación al analizador de archivos de entrada de la plataforma. Esta fase constó con un total de cuatro iteraciones.

Iteración 1

En la primera iteración los esfuerzos se dirigieron a la captura de requisitos y a la elaboración de esquemas generales del sistema.

- **Requisitos**: Mediante una serie de reuniones con el personal interesado en el desarrollo de la plataforma y del estudio de las herramientas actuales para la creación de la misma, se determinaron los principales requisitos funcionales que el presente proyecto debía cumplir de forma general.

- **Análisis**: En función de los requisitos obtenidos y las herramientas encontradas para la implementación de la plataforma, se crearon una serie de objetivos mínimos que la plataforma debía alcanzar, así como un boceto de lo que sería el reproductor de dicha plataforma.

El resultado de esta iteración fue la obtención de los requisitos funcionales mínimos, así como una aproximación a la base de la plataforma y los objetivos que debían alcanzarse para que ésta cumpliera los requisitos obtenidos en esta iteración.

Iteración 2

En la segunda iteración se buscaba crear la base del reproductor de la plataforma. Ésta debía basarse en un bucle principal de ejecución que empezara a utilizar las herramientas elegidas para la implementación de la plataforma, así como poder controlar sus iteraciones mediante el uso de algún tipo de temporizador.

- **Requisitos**: Se siguieron aquellos requisitos para la definición de la base del reproductor. A estos requisitos se les añadió uno nuevo en el que se establecía que las iteraciones del bucle del reproductor debían poder controlarse con el fin de disminuir la carga que éste representaba para el sistema.

- **Análisis**: A partir de estos requisitos se determinaron las funcionalidades básicas del bucle principal del reproductor de la plataforma, las cuales eran poder controlar las iteraciones del bucle y la captura de eventos de teclado por iteración.

- **Diseño**: Siguiendo estos requisitos, se diseñó dicho bucle principal junto a una clase auxiliar mediante diagramas UML y diagramas de clases.

- **Implementación**: Se generó el código asociado a la base del bucle principal del reproductor, creando una clase auxiliar (*Timer*) que permitía el control de las iteraciones del bucle contando instantes de tiempo entre las mismas. Esta clase auxiliar hace uso de *SDL* para contar los instantes de tiempo, siendo éste una de las herramientas elegidas para la implementación de la plataforma. Esta herramienta es usada también para la captura de eventos de teclado por iteración.

- **Pruebas**: Se realizaron una serie de pruebas unitarias en las que se establecía un límite de iteraciones por segundo del bucle y se observaba como éste respondía ante una serie de eventos de teclado. Se pudo observar que el bucle respondía perfectamente a dichos eventos aunque se bajaran radicalmente las iteraciones del bucle.

El resultado de esta iteración fue la obtención del bucle principal del reproductor de la plataforma, el cual ya permitía controlar su carga en el sistema regulando las iteraciones por segundo, así como la captura de eventos de teclado.

Iteración 3

En esta iteración se buscaba crear una analizador básico de los archivos de secuencia propios de la plataforma haciendo uso de alguna de las herramientas elegidas para dicha tarea.

- **Requisitos**: Se siguieron aquellos requisitos en los cuales se definía la estructura de los archivos de secuencia de la plataforma.

- **Análisis**: A partir de estos requisitos se determinaron las funcionalidades básicas que el analizador debía implementar, así como ciertas palabras clave y estructuras que este tipo de archivos utilizaría para la definición de las diapositivas.

- **Diseño**: Siguiendo el resultado del análisis, se diseñó un analizador que debería ser capaz de obtener los datos contenidos en este tipo de archivo basándose en *YAML* y las palabras reservadas designadas para la definición de las diapositivas. Utilizando diagramas UML y diagramas de clases, se diseñaron una serie de funciones que pudieran ser utilizables por la clase contenedora del bucle principal del reproductor, así como una serie de clases contenedoras de los datos obtenidos de los archivos en cuestión.

- **Implementación**: Se generó el código asociado al analizador, se extendió la funcionalidad del bucle del reproductor y se crearon las clases contenedoras de los datos asociados a las diapositivas contenidas en los archivos de secuencia.

- **Pruebas**: Se realizaron una serie de pruebas unitarias en las que se analizaban distintos archivos de secuencia con distintas distribuciones de las diapositivas, las cuales por el momento sólo admitían objetos del tipo *imagen*.

Con esta iteración se consiguió obtener una aproximación de lo que sería el analizador de archivos de entrada del reproductor, así como una pequeña herramienta para las pruebas de las siguientes iteraciones.

Iteración 4

En esta iteración se empezó con el diseño de la parte gráfica del reproductor, centrándose más en la parte de mostrar objetos por pantalla utilizando el bucle de ejecución creado en la iteración dos y usando el analizador de archivos de secuencia creado en la iteración tres. Dicha parte del reproductor debía poder mostrar además objetos de diapositiva del tipo imagen, así como transiciones a corte entre diapositivas y efectos especiales de pantalla en negro y pantalla en blanco.

- **Requisitos**: Se siguieron aquellos requisitos en los cuales se definía como debía comportarse la parte gráfica del reproductor.

- **Análisis**: A partir de estos requisitos se determinaron las funcionalidades básicas que el reproductor debía cumplir a la hora de mostrar las diapositivas, las transiciones y los efectos especiales.

- **Diseño**: Siguiendo estos requisitos, se diseñó dicha base del reproductor creando nuevos diagramas UML y completando el diagrama de clases de la iteración anterior, así como extendiendo la funcionalidad de las clases contenedoras en lo referente a renderizar el contenido de las mismas en pantalla. También se diseñó una clase gestora de transiciones y una clase gestora de estados especiales.

- **Implementación**: Se generó el código asociado al reproductor, el cual permitía mostrar imágenes básicas por pantalla a modo de diapositivas. Para ello se utilizó *SDL* y *OpenGL*. También se añadieron funciones auxiliares de renderizado y funciones específicas para las clases contenedoras. Además, se implementaron las clases gestoras de transiciones y de estados especiales.

- **Pruebas**: Se realizaron una serie de pruebas unitarias en las que se mostraban imágenes de diferentes resoluciones a diferentes iteraciones por segundo del bucle principal del reproductor, obteniendo buenos resultados en el visionado de dichas imágenes en pantalla, fuera cual fuera la resolución. Se iteró entre diapositivas comprobando que las transiciones a corte funcionaban correctamente, así como que los estados especiales de pantalla en negro y pantalla en blanco funcionaban al pulsar ciertas teclas.

En esta iteración se consiguió tener la base completa del reproductor, haciendo uso del bucle principal del reproductor y del analizador de archivos de secuencia. Gracias a esta base ya se podían mostrar imágenes por pantalla, transiciones y estados especiales utilizando la combinación de SDL y OpenGL.

6.1.2 Fase de Elaboración

Con los requisitos funcionales básicos, los objetivos mínimos y la base del reproductor implementada, se concertó una nueva reunión con el personal interesado en el desarrollo de la plataforma con el fin de obtener nuevos requisitos funcionales y correcciones sobre la base.

El resultado de la reunión fue la división de la plataforma en dos aplicaciones: un reproductor y una aplicación web de preguntas que permitiera hacer cuestionarios como una diapositiva más de una presentación. Esto conllevó nuevos requisitos funcionales, nuevos objetivos y modificaciones en los requisitos referentes al tipo de objetos que podían mostrar como diapositiva o como objeto de la misma. También se obtuvieron bocetos de las transiciones que debería poder soportar el reproductor, así como un nuevo estado especial que permitiera mostrar las diapositivas de la presentación en modo mosaico.

En esta fase se centró en terminar el diseño de la arquitectura de la plataforma y en identificar y diseñar los módulos a implementar.

Iteración 5

En esta iteración se diseñaron las nuevas transiciones, los nuevos estados especiales y los nuevos tipos de objeto de una diapositiva.

- **Requisitos**: Se analizaron los nuevos requisitos funcionales en lo referente al reproductor.

- **Análisis**: Con el análisis de estos requisitos se obtuvieron los nuevos objetivos a alcanzar, entre los cuales se encontraban la implementación de nuevas transiciones y estados especiales, nuevos tipos de objeto y nuevos tipos de archivo de entrada a la aplicación.

- **Diseño**: En función de estos objetivos se diseñaron nuevos diagramas de clases y diagramas UML para la futura implementación de las transiciones, objetos y estados especiales del reproductor que aún estaban sin implementar. También se ampliaron las funciones de clases ya implementadas, como por ejemplo las funciones del analizador.

Con esta iteración se obtuvo la base para la implementación de las últimas partes del reproductor.

Iteración 6

En esta iteración se diseñó la base de la aplicación web de preguntas, eligiendo las herramientas y los lenguajes necesarios para su implementación. También se diseñó la capa de comunicación entre el *frontend* y el *backend* de la misma.

- **Requisitos**: Se analizaron los requisitos funcionales en lo referente a la aplicación web de la plataforma.
- **Análisis**: Del análisis de estos objetivos se obtuvieron los objetivos a alcanzar para la implementación de las partes de la aplicación de preguntas.
- **Diseño**: Se diseñó la base del analizador de archivos de entrada para esta aplicación, así como los diagramas de clase del módulo de comunicación del backend con el frontend.
- **Implementación**: Se creó el código asociado a la base del analizador del backend y la base del frontend, añadiendo además el código para la comunicación del backend con el frontend usando peticiones *request* y con mandos auxiliares mediante el uso de *sockets*.

Gracias a esta iteración se creó la base de la aplicación web de preguntas, consiguiendo que las partes de la aplicación se comunicaran entre sí sin problemas.

6.1.3 Fase de Construcción

En esta fase se obtuvo la versión funcional de la plataforma: un reproductor de diapositivas totalmente funcional y una aplicación web para las realización de pequeños cuestionarios al público de una presentación.

Iteración 7

En esta iteración se continuó con el desarrollo del reproductor. En este caso, se crearon los objetos de diapositiva de audio y vídeo utilizando *GStreamer* y siguiendo el diseño creado en la iteración cinco. También se añadió el código referente a las teclas especiales.

- **Requisitos**: Se siguieron aquellos requisitos en los cuales se definían los tipos de objeto de diapositiva que hacían referencia a vídeos y audio. También se tomaron en cuenta los requisitos que establecían como se debían definir las teclas especiales en los archivos de secuencia y como debían ser interpretadas en el reproductor una vez obtenidas de este tipo de archivo.

- **Análisis**: A partir de estos requisitos se determinaron las funcionalidades básicas que el reproductor debía cumplir a la hora de reproducir vídeos y audio. También se analizaron los requisitos referentes a las teclas especiales, obteniendo nuevas palabras clave y estructuras para la definición de este tipo de objetos dentro de los archivos de secuencia.

- **Diseño**: Siguiendo estos requisitos, se continuó el diseño de las clases que extendían los tipos de objetos de diapositiva. También se diseñaron las tuberías de reproducción, tanto de vídeo como de audio, y las clases contenedoras de teclas especiales.

- **Implementación**: Se implementaron los nuevos tipos de objeto de diapositiva haciendo uso de *GStreamer*, así como las clases contenedoras de las teclas especiales. También se ampliaron las funciones del analizador para la parte de la obtención de los datos de las teclas especiales de los archivos de secuencia.

- **Pruebas**: Se realizaron una serie de pruebas unitarias en las que se mostraban diapositivas de diferente tipo, alternando el fondo con vídeos o imágenes. También se utilizaron este tipo de objetos como parte de una diapositiva, utilizando en ocasiones efectos de sonido y audio. El objetivo era comprobar que estos objetos se mostraban como debían, reproduciéndose correctamente. También se comprobó que los eventos de teclado generados lanzaban los efectos de las teclas especiales definidas en los archivos de secuencia.

Iteración 8

En esta iteración se terminó de construir el reproductor, generando así la primera versión funcional del mismo. Se creó el ultimo tipo de objeto de diapositiva, las transiciones deseadas y el último tipo de estado especial. También se añadieron ciertas teclas por defecto para la realización de ciertas acciones sobre el reproductor, y se amplió el analizador para que éste permitiera como entrada archivos con formato PDF.

- **Requisitos**: Se analizaron los últimos requisitos funcionales para el reproductor.

- **Análisis**: Del análisis de estos objetivos se obtuvieron los últimos objetivos que se debían alcanzar para obtener la primera versión funcional de la plataforma. Estos fueron crear el último tipo de objeto de transición (los objetos de captura), el resto de transiciones y estados especiales, así como el otro tipo de entrada al reproductor.

- **Diseño**: Siguiendo estos objetivos, se terminó el diagrama de clases para los objetos de diapositiva. También se añadieron nuevas funcionalidades para los tipos de transiciones y estados especiales restantes, y las funciones para el análisis del nuevo tipo de entrada al reproductor.
- **Implementación**: Se añadió el nuevo tipo de objeto de diapositiva, modificando el reproductor para que ciertas teclas especiales pudieran ser redirigidas a los programas de captura. También se crearon las funciones para el renderizado de las nuevas transiciones y el modo mosaico de la clase gestora de estados especiales. Por último se añadieron excepciones en aquellos eventos de teclas que no correspondían con ninguna tecla especial o tecla de captura, con el fin de que estas pudieran ser interpretadas en función de una serie de teclas por defecto para el reproductor.
- **Pruebas**: Se realizaron pruebas unitarias en las que se comprobó que los objetos de captura funcionaban correctamente. También se realizaron pruebas para comprobar que las transiciones iban a la velocidad que deberían y para comprobar si el estado mosaico seleccionaba bien las diapositivas al moverse entre las que mostraba. Por último, se realizaron pruebas para comprobar que las teclas por defecto funcionaban siempre que los eventos de teclas no se correspondieran con las teclas especiales definidas.

Iteración 9

Con el reproductor terminado, esta iteración buscaba la creación de las herramientas auxiliares complementarias a dicho reproductor. Algunas de ellas se crearon junto algunos módulos del reproductor, pero éstas no fueron accesibles desde fuera hasta la finalización de esta iteración.

- **Requisitos**: Se analizaron los requisitos funcionales referentes a las utilidades auxiliares que la plataforma debía ofrecer sin tener que arrancar el reproductor.
- **Análisis**: Gracias al análisis de estos requisitos se obtuvieron las distintas herramientas auxiliares que compondrían el conjunto de utilidades de la plataforma. También se obtuvieron las palabras clave de entrada a la aplicación que permitirían identificar el tipo de herramienta auxiliar a utilizar.
- **Diseño**: A partir del análisis de los requisitos se diseñaron un conjunto de funciones de utilidad para que la clase principal permitiera la ejecución de este tipo de herramientas.
- **Implementación**: Se creó el código asociado a cada una de las herramientas auxiliares. Una de estas herramientas necesitaba la modificación de una parte del reproductor para la generación de capturas de pantalla con el fin de poder generar el archivo PDF de la presentación asociado al archivo de entrada.

- **Pruebas**: Se realizaron pruebas unitarias para cada una de las herramientas auxiliares, consiguiendo los resultados buscados para cada una de dichas herramientas.

Gracias a esta iteración se obtuvieron las herramientas auxiliares que la plataforma debía ofrecer a los usuarios de la misma junto al reproductor.

Iteración 10

En esta iteración se continuó con el desarrollo de la aplicación web de preguntas, ampliando las funcionalidades de cada una de sus partes y optimizando la comunicación entre las mismas.

- **Requisitos**: Para esta iteración se siguieron los requisitos relacionados con los tipos de archivo de entrada de la aplicación, así como los requisitos de comunicación entre sus partes.

- **Análisis**: Gracias al análisis de estos requisitos, se obtuvieron los objetivos a alcanzar en esta iteración, entre los cuales se encontraba ampliar las funcionalidades del backend, la optimización del módulo de comunicación creado en la iteración siete y la creación de la clase gestora de la caché en el frontend.

- **Diseño**: Partiendo del análisis de estos requisitos, se crearon nuevos diagramas de secuencia y diagramas de clases. Haciendo uso de estos diagramas se diseñaron nuevas funcionalidades del backend de la aplicación y la clase gestora de la caché.

- **Implementación**: Se creó el código asociado a las nuevas funciones del backend, optimizando aquellas encargadas de la comunicación con el frontend y los mandos auxiliares. También se creó el código asociado a la clase gestora de la caché y su uso en en frontend.

- **Pruebas**: Se hicieron pruebas unitarias para comprobar el rendimiento de la aplicación ante diferentes archivos de entrada con diferentes tipos de preguntas. Se comprobó también como reaccionaba la cache ante las peticiones del backend y las peticiones del frontend.

Gracias a esta iteración se avanzó en el desarrollo de la aplicación web de preguntas, mejorando aquellos módulos ya implementados en iteraciones anteriores.

Iteración 11

En esta iteración se terminó de desarrollar la aplicación web de preguntas, terminado de cumplir los objetivos restantes para cada una de sus partes y obteniendo así la primera versión funcional de la misma.

- **Requisitos**: Se utilizaron los requisitos funcionales restantes para esta aplicación, los cuales se centraban en cómo debía ser la interfaz de usuario, la transición entre diapositivas, etc.
- **Análisis**: Gracias al análisis de estos requisitos se obtuvieron los estados del backend, el diseño de la interfaz de usuario y los modos del frontend.
- **Diseño**: Se completaron los diagramas creados en iteraciones anteriores añadiendo las nuevas funcionalidades asociadas a los nuevos requisitos.
- **Implementación**: Se implementaron los estados en el backend junto a las transiciones entre los mismos. También se creó el código asociado a la interfaz de usuario del frontend y los modos del mismo.
- **Pruebas**: Se hicieron pruebas unitarias comprobando que los estados del backend se correspondían con los modos del frontend, iterando entre preguntas y usando móviles y mandos auxiliares para asegurar que la comunicación entre las partes de la plataforma seguía funcionando correctamente.

Con esta iteración se terminaba de desarrollar la aplicación web de preguntas, obteniendo así la primera versión funcional de la plataforma junto al reproductor y las herramientas auxiliares.

6.1.4 Fase de Transición

Con el reproductor, las herramientas auxiliares y la aplicación web de preguntas implementados, la plataforma se encontraba funcionando en su versión beta. Por tanto se pasó a esta fase, en la cual se realizaron las pruebas funcionales que permitieron detectar y solventar aquellos errores que pudieran surgir durante la realización de la pruebas. Por último se planificó y realizó el despliegue de la plataforma.

Iteración 12

En esta última iteración se realizaron las pruebas funcionales, ofreciendo al personal interesado en el desarrollo de la aplicación participar en las mismas. De esta forma se

consiguieron corregir algunos errores y añadir algunas nuevas funcionalidades que surgieron durante las pruebas.

- **Implementación**: Gracias a las pruebas, se encontraron ciertos errores que fueron corregidos en el acto, ya que gracias a la modularización de las aplicaciones de la plataforma refactorizar el código de las mismas era una tarea sencilla. También se añadieron nuevas funcionalidades para la aplicación web de preguntas y para la interpretación de las teclas especiales en el reproductor.

- **Pruebas**: Tras las modificaciones anteriores se volvió a realizar una serie de pruebas que determinaron que la plataforma estaba lista para ser utilizada sin riesgos de errores que impidieran la correcta ejecución de la misma.

Como resultado de esta iteración se obtuvo la primera versión funcional de la plataforma, la cual podía ponerse a disposición de los usuarios junto a unas instrucciones de instalación.

6.2 COSTES

En esta sección se hablará de los costes asociados al desarrollo de la plataforma, teniendo en cuenta los recursos de los que ya de disponía antes de empezar el desarrollo y de los recursos que hicieron falta durante el mismo. También se darán estadísticas asociadas al repositorio de trabajo en el cual se fue alojando el código de la plataforma durante del desarrollo.

6.2.1 Costes de desarrollo

El desarrollo de la plataforma *iCROM* ha abarcado desde el 01 de Octubre de 2015 hasta el 01 de Junio de 2016. Hasta Enero de 2016 se trabajó en el desarrollo de la plataforma empleando media jornada cinco días a la semana. A partir de Enero se aumento a jornada completa debido a que la carga de desarrollo aumentó para esa fecha. Teniendo en cuenta que el salario mínimo por hora de un programador se estima entre 25-35€[1] brutos por hora, eso hace un total aproximado de 34.000€. A esto se ha de sumar el coste de los recursos nuevos y existentes utilizados en el desarrollo, lo que hace un total de 36.200€.

[1] www.infojobs.net

Tabla 6.2: Coste asociado al desarrollo de la aplicación.

Recurso	Cantidad	Coste
Salario desarrollador software	1	34.500€
Asus A53S	1	600€
Proyector	1	200€
Mandos auxiliares	20	700€
Receptor mandos auxiliares	1	100€
Total		**36.200€**

6.2.2 Estadísticas del desarrollo

Durante el desarrollo, el código que se iba generando fue almacenado en un repositorio privado de *Github*[2]. Una vez terminado el desarrollo, este repositorio pasó a ser público y gracias a él se han podido obtener las siguientes estadísticas del mismo.

Tabla 6.3: Lineas de código de *iCROM* contabilizadas gracias a la aplicación *cloc*.

Lenguaje	Archivos	En blanco	Comentarios	Código
C++	38	864	222	3847
PHP	14	180	32	864
Python	1	182	27	438
CSS	2	48	2	357
HTML	1	0	0	76
JavaScript	4	10	0	69
Make	1	11	6	20
Bourne Shell	2	2	0	8
Total	**63**	**1297**	**289**	**5679**

[2]www.github.com

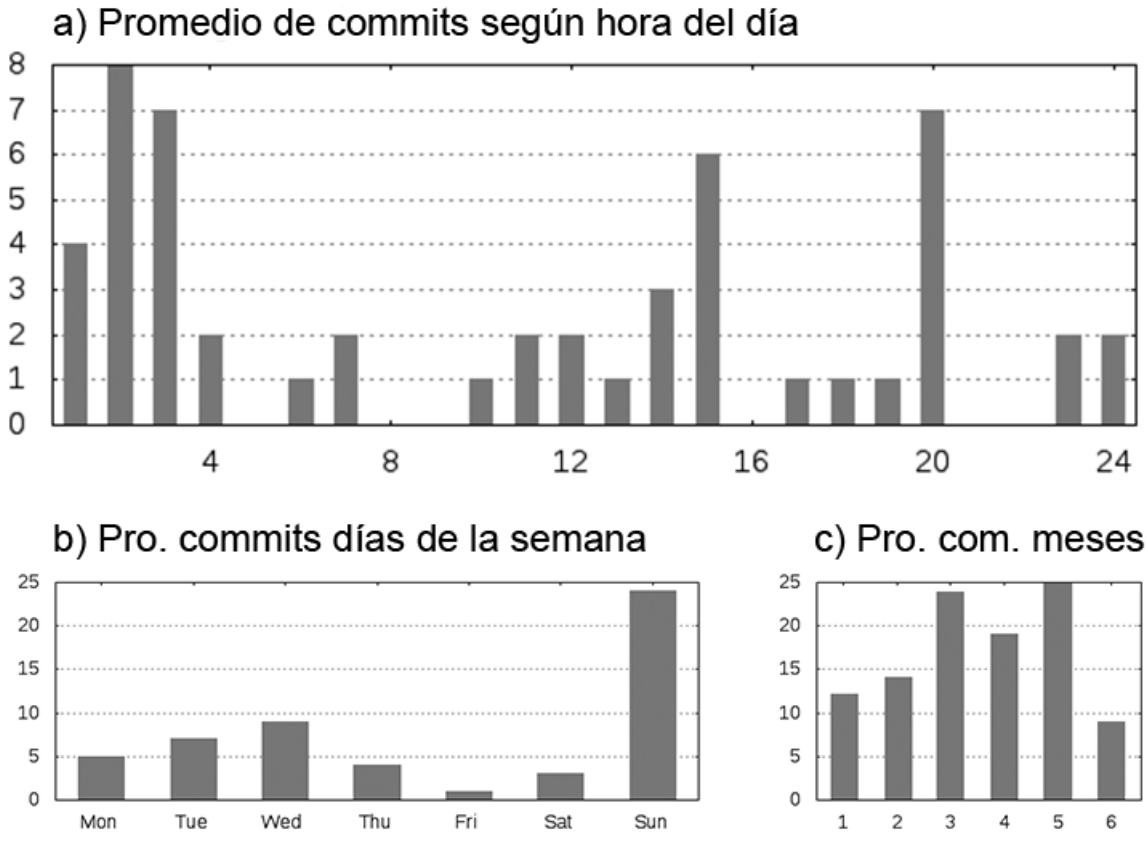


Figura 6.1: Estadísticas del desarrollo obtenidas gracias al repositorio de trabajo.

CAPÍTULO 7

RESULTADOS Y CONCLUSIONES

En este capítulo se muestran una serie de resultados obtenidos tras someter la plataforma a ciertas pruebas, se describen los objetivos alcanzados durante el desarrollo y se proponen una serie de ideas para la mejora y para un posible trabajo futuro que permita extender las funcionalidades de la plataforma *iCROM*.

7.1 RESULTADOS

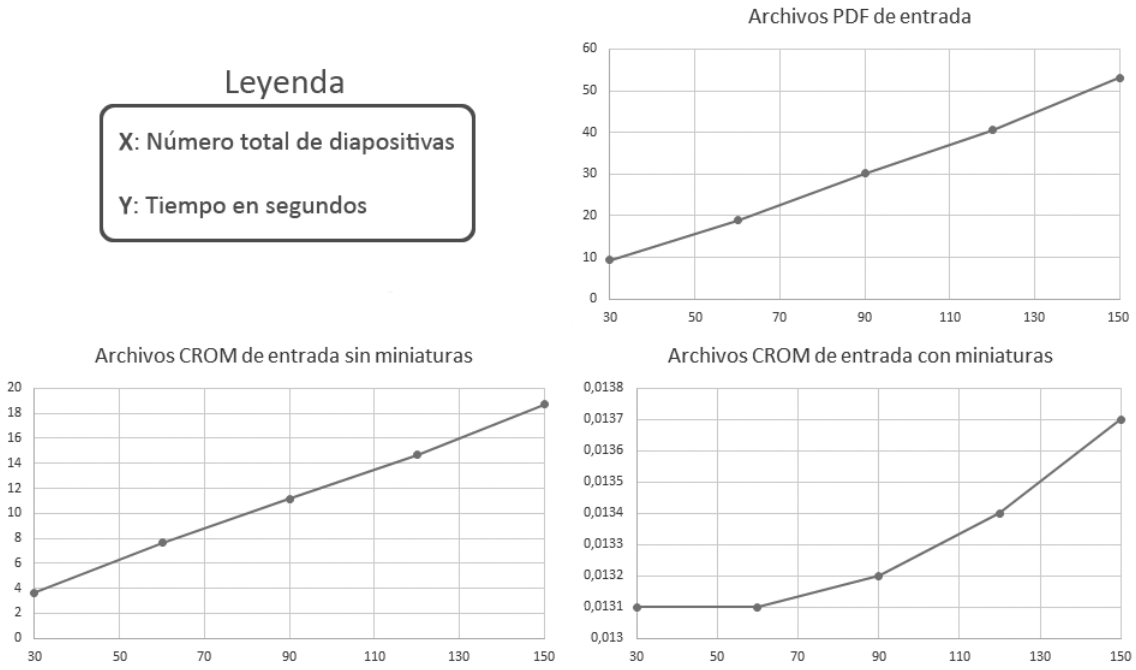
Con el fin de someter la plataforma a ciertos casos de uso extremos, se realizaron una serie de pruebas evaluando la carga que la plataforma genera para el sistema y comprobando a la vez la carga de datos que puede soportar la plataforma en distintos tipos de computador.

7.1.1 Tiempos de carga

Se han realizado una serie de pruebas para comprobar como responde el reproductor de la plataforma ante diferentes tipos de entrada y tamaños de presentación. Los resultados fueron:

- **Archivo PDF de entrada**: Los resultados en este caso no resultaron sorprendentes, ya que se sabía de antemano que el tiempo de carga aumenta en función del tamaño de la presentación. Esto se debe a que a mayor número de imágenes por PDF, mayor tiempo de extracción de las mismas y mayor tiempo de generación de miniaturas.
- **Archivo CROM de entrada sin miniaturas**: Estos resultados son parecidos a los anteriores pero con una curva de crecimiento menor al no necesitar de la extracción de las imágenes de un archivo PDF. El tiempo de carga aumenta con el tamaño debido a que la generación de miniaturas aumenta también en función a esto.
- **Archivo CROM de entrada con miniaturas**: El resultado en este caso es una curva de pequeño crecimiento. Esto se debe a que no requiere nada más que del tiempo de

parseo del archivo CROM, todo lo demás ya está creado y almacenado donde debe. Si el archivo CROM es muy grande, el tiempo de parseo del mismo también es mayor, pero de cara al tiempo de inicio no se notaría, ya que se está hablando de diferencias de tiempo de milisegundos.

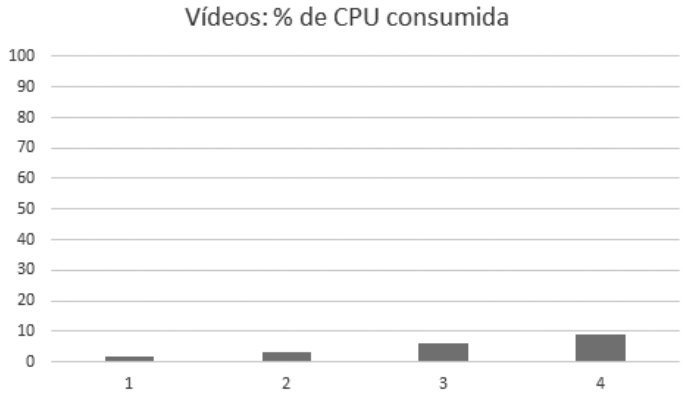


Los resultados obtenidos en general han resultado ser similares en todos aquellos computadores donde se han realizado las pruebas, independientemente de la capacidad computacional de los mismos.

7.1.2 Capacidad para la reproducción de vídeos

Con el fin de determinar la carga que soporta el reproductor a la hora de reproducir vídeos, y la carga que esto genera a su vez para el sistema, se realizaron una serie de pruebas donde se sometía al reproductor a la reproducción simultánea de una serie de vídeos, comprobando en cada caso la tasa de imágenes por segundo que generaba el reproductor. Las pruebas a las que se sometió en este caso al reproductor fueron:

- *Vídeo a pantalla completa* (1).
- *Dos vídeos reproducidos al mismo tiempo* (2).
- *Cuatro vídeos reproducidos al mismo tiempo* (3).
- *Ocho vídeos reproducidos al mismo tiempo* (4).

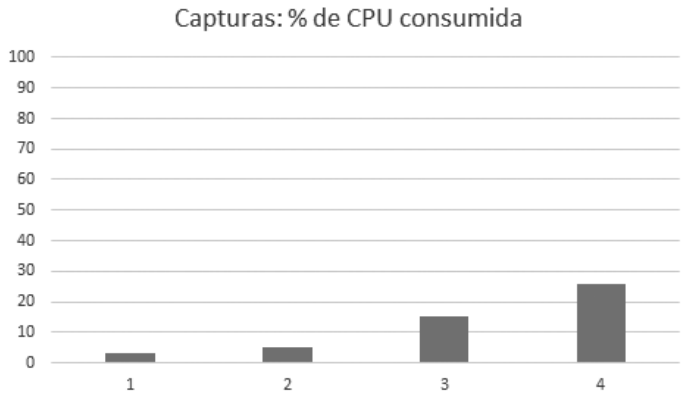


Los resultados que se obtuvieron revelaron que la tasa de imágenes por segundo no se veía afectada por la reproducción de una cantidad inferior a ocho vídeos a la vez. Por el contrario, la carga en el sistema se incrementaba un 1.5 % por cada vídeo que se reproducía en segundo plano.

7.1.3 Capacidad para la captura de programas

Al igual que la prueba con los vídeos, se realizaron una serie de pruebas con las capturas para obtener los límites del reproductor en este aspecto. Las pruebas a las que se sometió en este caso al reproductor fueron:

- *Captura a pantalla completa* (1).
- *Dos capturas simultáneas* (2).
- *Cuatro capturas simultáneas* (3).
- *Ocho capturas simultáneas* (4).



Los resultados que se obtuvieron revelaron que la tasa de imágenes por segundo en este caso si se veía afectada a partir de las cuatro capturas simultáneas, disminuyendo esta tasa

exponencialmente según se iba incrementando dicho número de capturas simultáneas. La carga en el sistema también incrementaba por captura, llegando a un 26 % de CPU y alrededor de 15 FPS con la prueba de ocho capturas simultáneas.

7.2 OBJETIVOS ALCANZADOS

Durante el desarrollo de la plataforma *iCROM* se ha conseguido completar cada uno de los objetivos propuestos en el *Capítulo 3*, tanto específicos como generales.

El objetivo general de la plataforma se ha completado con creces, ya que el resultado del desarrollo ha ido más allá de lo que se pedía en un primer momento. *iCROM* no es sólo un conjunto de herramientas auxiliares para el despliegue de presentaciones efectivas, ahora es algo más, es una plataforma completa con un reproductor propio y caracterizado por hacer ciertas cosas que otros reproductores de presentaciones multimedia no hacen, además de que incorpora una aplicación web para hacer cuestionarios al público de la presentación, aplicación que se puede embeber en el reproductor como una diapositiva más de una presentación o ejecutarse por separado. Gracias a esto, las presentaciones pueden ser gamificadas, aprovechando el canal de retorno del público asistente. De esta manera se consigue la interacción buscada en una presentación, haciendo participar activamente al público de la misma. Si a esto se le suma que consta de una serie de herramientas auxiliares, el resultado es una plataforma completa y fácil de extender con nuevas funcionalidades.

En cuanto a los objetivos específicos, éstos también se han cumplido completamente:

- La plataforma consta de un reproductor, de una aplicación web de preguntas y de un conjunto de herramientas auxiliares, cumpliendo los requisitos que se pedían para cada de una de ellas. El reproductor hace uso de SDL y OpenGL para mostrar las diapositivas junto con su contenido, la aplicación web está basada en tecnologías ampliamente extendidas, libres y de hosting económico, y las herramientas auxiliares son libres y multiplataforma. De esta forma se cumplen los objetivos **A.1**, **A.3**, **A.4**, **B.1**, **B.2** y **C.2**.

- La plataforma hace uso de archivos de secuencia, los cuales utilizan un lenguaje de encriptación del flujo y de la configuración de la presentación basado en YAML. Además, permite archivos PDF como archivos de entrada, generando herramientas auxiliares para tratar con este tipo de archivos. De esta forma se cumplen los objetivos **A.2** y **A.5**.

- La plataforma permite la reproducción de vídeo y de audio, soportando más de dos flujos de vídeo y audio simultáneos. De esta forma se complementa el cumplimiento del objetivo **A.1**.

- La plataforma permite que el reproductor se ajuste a cualquier tipo de resolución gracias a las funciones de reajuste implementadas como funciones auxiliares de dicho reproductor. Éstas permiten ajustar la proyección del reproductor a cualquier tipo de resolución, incluyendo las que se solicitaban en un principio (4:3, 16:9). Además, el reproductor soporta los modos de pantalla en blanco, pantalla en negro y mosaico. Al igual que en el caso anterior, de esta forma se complementa el cumplimiento del objetivo **A.1**.

- La plataforma permite la inclusión de cualquier tipo de diapositiva en el reproductor gracias a las diapositivas de captura, satisfaciendo así aquellos objetivos que pedían tener cierto contenido como diapositiva durante la presentación (preguntas test, visualización de tuits, etc.). De esta forma se cumplen el objetivo **C.1** y se complementa el cumplimiento de los objetivos **B.1** y **B.2**.

- La plataforma permite generar el archivo PDF de la presentación siguiendo una determinada plantilla. Este objetivo se llevó más allá dando la opción al usuario de elegir durante la presentación que diapositivas prefería que se incluyeran en el PDF resultante y cuales no mediante el uso de una tecla de acceso rápido del reproductor. De esta forma se complementa el cumplimiento del objetivo **A.4**.

- El reproductor de la plataforma permite la definición de teclas especiales para la realización de ciertas funcionalidades ajustables a toda la presentación o sólo a una parte de la misma. De esta forma se cumple el objetivo **B.4** y **B.5**.

- La plataforma permite la ejecución de preguntas test como diapositivas gracias a la aplicación web de preguntas. Ésta está diseñada para minimizar el uso del ancho de banda y se puede ejecutar localmente en caso de no disponer de conexión a la red. De esta forma se cumplen el objetivo **B.3** y **C.3**, complementando a su vez el cumplimiento de los objetivos **B.1**, **B.2**, **C.1** y **C.2**.

7.3 TRABAJO FUTURO

Gracias a que la plataforma *iCROM* se ha desarrollado como una plataforma modular, la inclusión de nuevas funcionalidades o módulos no requiere de grandes modificaciones en código, lo que la hace fácilmente extendible.

Algunas posibles mejoras de la plataforma podrían ser:

- El uso de un procesador de lenguajes totalmente propio, que utilizara un analizador léxico, sintáctico y semántico basado en ciertos tokens creados para los archivos CROM. Esto liberaría a la plataforma de la dependencia de YAML para el análisis de este tipo

de archivos y solventaría muchos errores que conlleva el uso de esta tecnología para el análisis de archivos.

Para esta mejora, habría que cambiar el módulo de parseo completamente, manteniendo las palabras clave. Esto podría conllevar un esfuerzo alto, con un tiempo de diseño y desarrollo aproximado de un mes de trabajo.

- La proyección se podría pasar a una proyección perpendicular, ganado así la sensación de profundidad y permitiendo la inclusión de objetos 3D como objetos de diapositiva, nuevos tipos de transiciones más realistas, etc.

  Con el fin de conseguir esta mejora, habría que adaptar la parte de OpenGL del módulo de reproducción, cambiando la proyección y reajustando el ángulo de las diapositivas. Para esta tarea de mejora, se estima entre una y dos semanas de trabajo.

- Las definición de diapositivas se podría optimizar creando una nueva regla en los archivos de secuencia que permitiera referenciar diapositivas ya creadas a las cuales se les añadirían nuevas imágenes, vídeos, capturas o música. De esta manera se evitaría la necesidad de memoria extra para diapositivas similares.

  Para esta tarea de mejora habría que modificar el módulo de parseo y el módulo de reproducción, modificando por encima el módulo de multimedia. Esta mejora conllevaría un esfuerzo medio, estimando el tiempo de trabajo en 2-3 semanas.

- Se podrían añadir teclas que permitieran el acceso directo a diapositivas de la presentación, con el fin de tener siempre a mano aquellas diapositivas relevantes de la presentación, tales como minijuegos, rankings, etc. La funcionalidad al pulsar de nuevo dichas teclas sería inversa, devolviendo al ponente a la diapositiva desde la que accedió mediante este atajo a la actual.

  Al igual que para la mejora anterior, habría que modificar el módulo de parseo y el módulo de reproducción. Al ser una mejora sencilla de realizar, se realización se estima en 2-3 días de trabajo, incluyendo el tiempo para las pruebas.

En el caso de la aplicación web de preguntas, al estar basada en estilos, su interfaz es fácilmente adaptable a cualquier estilo deseado por el usuario. Además, al estar también modularizada, se podrían añadir nuevas funciones que permitieran dar más estadísticas en lo referente a ratio de respuestas, participantes que aciertan y participantes que fallan una pregunta, etc. En cualquier caso, la funcionalidad que mejor se podría añadir a esta aplicación es la capacidad de atender más de una sesión a la vez, consiguiendo así publicar el frontend de la plataforma en un servidor público del cual hicieran uso todos aquellos backends que quisieran generar una sesión de cuestionario nueva.

7.4 CONCLUSIÓN PERSONAL

Durante el desarrollo del actual Trabajo Fin de Grado he podido aplicar muchos de los conocimientos adquiridos a lo largo de mi formación como Ingeniero Informático, utilizando además ciertos conocimientos adquiridos durante mi formación como Experto en Desarrollo de Videojuegos.

Me gusta pensar que *iCROM* es un gran videojuego creado para aquellos profesionales de las presentaciones multimedia que quieren hacer algo diferente en sus presentaciones. Por este motivo, me he esforzado mucho en conseguir un resultado excepcional, algo que pudiera enseñar a mis conocidos y amigos y poder decir "*Esto lo he hecho yo*". La cantidad de desafíos que he tenido que afrontar para conseguir acabar el desarrollo de este proyecto han sido numerosos y variados, pero cada uno de ellos me hacía avanzar, aprendiendo algo nuevo que no sabía, utilizando herramientas que desconocía, etc.

Hace cinco años que empecé mi carrera en este sector y he de decir que año tras año me he ido enamorando más y más de ésta mi profesión. Muchos conocimientos adquiridos durante las asignaturas impartidas en esta carrera me han permitido crear este proyecto desde cero, cosa que hace unos años ni se me hubiese pasado por la cabeza que podría hacer. Terminar con un proyecto como este, el cual ha requerido un gran esfuerzo por mi parte, para mi ha sido la mejor forma de concluir una etapa y empezar otra como Ingeniero Informático.

ANEXO A

MANUAL DE USUARIO

En este anexo se incluyen las instrucciones para conseguir instalar y ejecutar la plataforma **iCROM** en **Linux**. La plataforma está basada en librerías multiplataforma, por lo que la instalación y ejecución en *Windows* no debe suponer ningún problema para el usuario.

A.1 APLICACIÓN pCROM

En esta sección se explicará paso a paso como se debe realizar la tarea de instalación y ejecución de la aplicación *pCROM*.

A.1.1 Instalación

La instalación de *pCROM* requiere de la instalación de las dependencias y de la compilación del código fuente de la aplicación. Las dependencias que deben estar instaladas para que la aplicación funcione sin problemas son las siguientes:

- **Boost** (libboost-filesystem1.55-dev & libboost-system1.55-dev)
- **SDL2** (libsdl2-dev & libsdl2-image-dev)
- **YAML-CPP** (libyaml-cpp-dev)
- **Imagemagick** (imagemagick)
- **PDFJAM** (pdfjam)
- **OpenGL** (libglu1-mesa-dev & libglew-dev)
- **GStreamer** (libgstreamer1.0-dev & libgstreamer-plugins-base1.0-dev)
- **GTK3** (libgtk-3-dev)

- **XDO** (libxdo-dev)

Para instalarlas, se puede ejecutar el comando listado en A.1 en consola. Si alguna de ellas ya se encuentra instalada, ésta será omitida.

Listado A.1: Comando para la instalación de las dependencias de *pCROM*

```
$ sudo apt-get install libboost-filesystem1.55-dev
    libboost-system1.55-dev libsdl2-dev libsdl2-image-dev
    libyaml-cpp-dev imagemagick pdfjam libglu1-mesa-dev libglew-dev
    libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
    libgtk-3-dev libxdo-dev
```

Una vez instaladas las dependencias de la aplicación, el siguiente paso es compilar el código fuente.

Listado A.2: Comando para la compilación de *pCROM*

```
$ cd icrom-player/pcrom/ && make
```

Como se puede ver en el listado A.2, lo primero es situarse en la carpeta con el código fuente de la aplicación, para luego ejecutar el comando **make** sobre la misma. Terminada la compilación, si todo ha ido bien, se habrá creado el archivo ejecutable que permitirá la ejecución de archivos CROM y de archivos PDF, siempre y cuando se indique la ruta relativa o absoluta a dicho ejecutable. Si lo que se desea es que la aplicación pueda ser accesible desde cualquier parte del sistema sin tener que indicar la ruta a su ejecutable, se deberá añadir la ruta a dicho ejecutable en la variable de entorno **PATH** del sistema. También se puede hacer una copia del ejecutable en la carpeta **bin** del usuario actual, la cual es una de las carpetas añadidas por defecto al *PATH* del sistema. Esta carpeta se encuentra en **/home/usuario/bin**. Normalmente esta carpeta no está creada, aunque depende de la distribución Linux que se use. En caso de no existir, habría que crearla antes de copiar el ejecutable.

Listado A.3: Comando para el despliegue de *pCROM*

```
$ mkdir /home/user/bin && cp icrom-player/pcrom/pcrom
    /home/user/bin/.
```

A.1.2 Ejecución

Para ejecutar *pCROM*, hay que tener en cuenta que tipo de archivo se pretende presentar. En caso de ser un PDF, la aplicación acepta (aparte de dicho archivo) un flag que permite salvar el archivo CROM generado a partir del archivo PDF una vez la presentación haya acabado. Esto es útil si se quiere evitar el paso de PDF a archivo CROM para futuras presentaciones. En caso de archivos CROM, con indicar la ruta a dicho archivo bastaría para poder empezar la presentación. La construcción de este tipo de archivo se puede encontrar en el *Anexo B*.

Listado A.4: Reproducción de archivos PDF

```
$ pcrom archivo.pdf [-save]
```

Listado A.5: Reproducción de archivos CROM

```
$ pcrom archivo.crom
```

Cada vez que arranca el reproductor, se realizan una serie de acciones iniciales que dependerán del tipo de entrada. Estas acciones realizan tareas relacionadas con la obtención de las imágenes de la presentación, la generación de las miniaturas de las diapositivas, etc. Si la presentación parte de un archivo PDF, el reproductor tardará más en arrancar que si se parte de un archivo CROM con las miniaturas ya generadas. Por este motivo, se recomienda haber generado el archivo CROM asociado al archivo PDF en cuestión antes de presentar éste a un cierto público. Esta tarea se puede realizar gracias a una de las herramientas auxiliares que se pueden encontrar en el *Anexo D*.

Una vez la presentación haya arrancado, se podrá presentar su contenido utilizando para ello las teclas definidas por defecto para el control del reproductor (*Anexo C*). Algunos consejos útiles a tener en cuenta durante una presentación:

- Si durante la presentación se desea acceder a una diapositiva en concreto que esté separada por varias diapositivas de la actual, se puede utilizar el modo mosaico con el fin de acceder de manera rápida y sencilla a la diapositiva deseada.
- En caso de tener una captura de un programa a la cual se esté redirigiendo las teclas de avance o retroceso de página, el reproductor permite acceder a los valores por defecto que estas tienen pulsando **Shift** a la vez que la tecla deseada.
- En caso de necesitar ocultar la diapositiva actual o mostrar una diapositiva en blanco con el fin de escribir en la zona donde ésta se esté proyectando, el reproductor permite los modos de pantalla en negro y pantalla en blanco pulsando sus respectivas teclas especiales.
- En aquellas diapositivas donde existan varias capturas simultáneas, se ha de tener en cuenta que la redirección de teclas (en caso de existir) se hace para cada una de las capturas. Esto puede suponer una ventaja o un inconveniente, por lo que se ha de tener en consideración a la hora de tener varias capturas simultáneas y redirigir las mismas teclas para cada una de las capturas.
- A la hora de generar el archivo PDF de la presentación hay que tener especial cuidado debido a que cuando se guarda la diapositiva actual junto a su media como imagen, también se guardan los posibles bordes negros que ésta pueda tener. Por este motivo, sólo se recomienda la generación de la versión PDF de la presentación cuando la

resolución de la ventana de la aplicación no obligue a tener bordes negros para reajustar las diapositivas.

A.2 APLICACIÓN qCROM

En esta sección se explicará paso a paso como se debe realizar la tarea de instalación y ejecución de la aplicación *qCROM*.

A.2.1 Instalación

La aplicación web de *qCROM* está pensada para ser utilizada en servidores externos, aunque también se puede usar localmente. En caso de uso en servidores externos, éstos deberán tener instaladas las siguientes dependencias:

- **Apache2** (apache2 & apache2-dev)
- **PHP5** (php5 & php5-dev & libapache2-mod-php5)

Si el servidor tiene instaladas las dependencias, el siguiente paso sería dar permisos de escritura a la carpeta con la aplicación web y la instalación habría terminado. Si en cambio la aplicación se va a usar localmente y no se tienen las dependencias instaladas, para instalarlas se puede ejecutar el comando listado en A.6 en consola.

Listado A.6: Comando para la instalación de las dependencias de *qCROM*

```
$ sudo apt-get install apache2 apache2-dev php5 php5-dev
    libapache2-mod-php5
```

Una vez instaladas las dependencias, la aplicación web se deberá copiar a **/var/www/html**. Al igual que pasa con el servidor, la aplicación requiere de permisos de escritura para funcionar sin problemas, por lo que el siguiente paso sería darle dichos permisos a la carpeta que contiene los archivo de *qCROM*.

Listado A.7: Comando para el desplieque de *qCROM*

```
$ sudo cp -r icrom-player/qcrom/page /var/www/html/qcrom && sudo
    chmod a+w -R /var/www/html/qcrom
```

La aplicación *qCROM* funciona en base a sesiones iniciadas por aplicaciones locales de control. Estas aplicaciones envía las preguntas, reciben los eventos del presentador, las respuestas de los usuarios, llevan el control del estado del cuestionario, etc. Para que dichas aplicaciones funcionen, es necesario instalar las siguientes dependencias.

- **Python3** (python3)
- **Python Request** (python3-request)

Una vez instaladas las dependencias, la aplicación local de control mencionada anteriormente deberá ser capad de ejecutarse sin problemas.

A.2.2 Ejecución

Para empezar un nuevo cuestionario, es necesario ejecutar la aplicación local de control para crear una nueva sesión, ya que la aplicación web siempre está a la espera de nuevas solicitudes para mostrar los datos asociados a un cuestionario.

La aplicación de control que inicia la nueva sesión con las preguntas se encuentra en la subcarpeta **icrom-player/qcrom/control** y utiliza un archivo de configuración que establece ciertas variables necesarias para la ejecución de la aplicación, por lo que la ejecución de dicha aplicación se debe realizar en ese directorio y no en otro. La estructura de este tipo de archivo se puede encontrar en el *Anexo E*.

Listado A.8: Comando para la ejecución de la aplicación de control de *qCROM*

```
$ cd icrom-player/qcrom/control && python3 control.py
```

Una vez iniciada una nueva sesión, el cuestionario habrá empezado y los usuarios podrán registrarse accediendo a la página e ingresando un identificador. Esta sesión requiere de dos tipos de contraseñas, una de administrador y otra de peticiones a la página. Ambas se deben crear en la subcarpeta **misc** de la carpeta que contiene la aplicación web de *qCROM*.

- **admin-pass**: Este es el nombre que ha de tener el archivo con la contraseña del administrador, sin extensión. Dentro deberá contener una única cadena de texto, sin saltos de línea o espacios. De lo contrario, se considerará un archivo no válido y se utilizará la contraseña por defecto (**4dm1n**). Hay que tener en cuenta que la contraseña de administrador se utiliza para que el presentador pueda iniciar sesión como tal en la página, por lo que es muy importante utilizar una contraseña diferente a la de por defecto si no se desea que otra persona física interfiera en la presentación como administrador no deseado.
- **qcrom-pass**: En este caso, este es el nombre del archivo con la contraseña para las solicitudes del backend al frontend. Esta contraseña se utiliza para comprobar que las peticiones que llegan desde la aplicación de control son las que están relacionadas con el cuestionario actual. Al igual que pasa con el archivo de administrador, este archivo no puede contener saltos de línea ni espacios en blanco. De darse el caso, se consideraría

como un archivo no válido y se utilizaría la contraseña por defecto para esta tarea (**q640M**).

Los usuarios que no sean administradores no necesitarán introducir ninguna contraseña, ya que con un identificador bastaría para distinguir un usuario de otro. En el momento que el presentador (con el rol de administrador) inicie el cuestionario, ningún nuevo usuario podrá registrarse para este cuestionario, por lo que no podrá participar en el mismo.

Figura A.1: Captura de las vistas de intro y login, respectivamente, de *qCROM*.

Durante el concurso, la vista que genera la aplicación web para el administrador es diferente que la que se genera para los usuarios. Esto es así porque la vista generada para el administrador tiene por objetivo dar más información de la que da la vista para los usuarios, ya que ésta sólo tiene por objetivo mostrar las posibles respuestas a una pregunta, guardar aquella que el usuario seleccione y mostrar el resultado de preguntas ya contestadas, mientras que la del administrador debe mostrar la pregunta, las respuestas, el posible media asociado a la pregunta, la cantidad de respuestas, etc.

Figura A.2: Captura de la vista de administrador y de usuario desde móvil en *qCROM*.

Para que esto ocurra, se utilizan una serie de cookies que permiten al archivo de la aplicación web encargado de generar la vista, generar una u otra en función de los datos que estas cookies contengan. Éstas se comprueban periódicamente comprobando que no hayan sucedido cambios en las mismas. Si se diera el caso, se generaría una nueva vista que actualizaría a la actual sin recargar la página para conseguir este fin.

Una vez terminado el cuestionario, el presentador podrá finalizar la aplicación de control introduciendo **exit** en la terminal donde se esté ejecutando, y será esta aplicación la que pare la sesión actual y limpie la caché de la página de *qCROM*, eliminando a su vez las cookies de sesión utilizadas por la aplicación web. Todo el concurso se guarda en una base de datos creada con **SQLite** en **/tmp**. Este archivo recibe el nombre de **qcrom-data**, y si se inicia una nueva sesión o se apaga el computador, el contenido de este archivo se perderá. Se recomienda hacer una copia de dicho archivo antes de llevar a cabo una de las dos acciones anteriormente comentadas si se desea guardar el archivo para un futuro uso.

ANEXO B

ARCHIVOS CROM

Los archivos CROM son el punto fuerte de la aplicación *pCROM*. Estos archivos se basan en la idea de crear presentaciones mediante código. Haciendo uso de *YAML*, los archivos CROM permiten establecer desde los *FPS* de la aplicación hasta capturas en tiempo real de programas a los cuales poder enviar las pulsaciones de teclas (que el usuario elija previamente) durante una presentación.

En este anexo, se hará descripción de estos archivos, de su sintaxis y de los posibles errores que un usuario se puede encontrar durante su creación/ejecución.

B.1 DESCRIPCIÓN

Al usar YAML, los archivos CROM no requieren de compilación para poder ser ejecutados por la aplicación *pCROM*, pero si de un análisis previo a la ejecución en busca de posibles errores. Un archivo CROM suele estar formado por tres partes diferenciadas: **Configuración general**, **Diapositivas** y **Teclas especiales**. No todas estas partes son obligatorias, ya que para que *pCROM* pueda arrancar sólo necesita tener la sección de las *Diapositivas*.

Cada una de estas secciones contiene una serie de parámetros que dependiendo de en que sección se apliquen, tendrán uno u otro efecto. El orden de estos parámetros dentro de una sección es irrelevante, es decir, se pueden poner en el orden que el usuario prefiera. Lo único que hay que tener siempre en cuenta es que estos parámetros deben separarse entre sí por comas (’,’), ya que si esto no se hiciera, el archivo estaría mal construido y el parser no lo admitiría. Ejemplo de sección con parámetros separados por comas:

```
1 configuration:
2 {
3     fps_limit: 25,
4     images_directory: "media/images",
5     videos_directory: "media/videos"
6 }
```

B.2 CONFIGURACIÓN GENERAL

```
configuration:
{
    ...
}
```

En la configuración general se definen parámetros que afectan a la aplicación en sí (como puede ser la definición de los FPS del programa) o a la sección de diapositivas, es decir, a todas las diapositivas en general. Se puede tener una configuración vacía o directamente no tenerla ya que los parámetros toman valores por defecto, predefinidos por *pCROM*, si éstos no se definen para una presentación. Muchos de estos parámetros son útiles para aplicar efectos sobre todas las diapositivas, pero si dichos efectos se quieren evitar en una de las mismas, se puede indicar explícitamente que dichos efectos no tengan lugar.

Parámetros de la configuración general:

- **fps_limit**: Establece un límite a los frame por segundo de la aplicación. Si la presentación contiene vídeos, este parámetro deberá tener un valor igual o superior al máximo de los fps de dichos vídeos (para evitar "interferencias" en la reproducción de aquellos cuyos FPS sean superiores a los del programa). Por defecto toma un valor de *60*.

- **transition_default_value**: Define la transición por defecto para todas las diapositivas de la presentación.

- **header_file**: Indica a la aplicación donde se encuentra la imagen de cabecera para poder ponerla en la aplicación *sobre* las diapositivas. Hay que tener en cuenta que esta imagen ha de tener transparencia si se quiere que se vea la diapositiva sobre la que se encuentra. No sólo es útil si se quiere establecer una cabecera para las diapositivas, sino que vale para muchas otras cosas, como por ejemplo, poner *zonas de modificación* sobre la diapositiva original que corrija a ésta en alguno de los sentidos. Esto libera al creador de la presentación a tener que rehacer dicha diapositiva.

- **header_default_value**: Establece el valor por defecto de la cabecera en cada una de las diapositivas. Este valor puede ser "**on**" u "**off**". Por defecto toma el valor de "**off**".

- **fx_default_volume**, **video_default_volume** y **music_default_volume**: Estos tres parámetros tienen por objetivo dar un volumen general a aquellos efectos de sonido, vídeos o música en los cuales no se haya definido un volumen propio. Toma valores entre **0.0** y **1.0**, siendo **1.0** el límite del volumen del ordenador donde se ejecute la presentación. El volumen se puede incrementar a **10.0**, pero no es recomendable ya que se estarían forzando los altavoces del equipo por donde se emita el sonido.

- **video_default_loop** y **music_default_loop**: Estos parámetros tienen por objetivo dar un valor general a los parámetros **loop** de vídeos y música que no lo tengan definido. Este valor puede ser "**on**" u "**off**". Por defecto toma el valor de "**off**".

- **images_directory**: Este parámetro le indica a la aplicación donde encontrar las **imágenes** que se quieran mostrar en la sección de *Diapositivas*. Por defecto, busca en el directorio desde donde se llame al programa.

- **videos_directory**: Este parámetro le indica a la aplicación donde encontrar los **vídeos** que se quieran reproducir en la sección de *Diapositivas*. Por defecto, busca en el directorio desde donde se llame al programa.

- **sounds_directory**: Este parámetro le indica a la aplicación donde encontrar las **sonidos** que se quieran reproducir en la sección de *Diapositivas*. Por defecto, busca en el directorio desde donde se llame al programa.

B.3 DIAPOSITIVAS

```
1 slides:
2 {
3     ...
4 }
```

Figura B.1: Diapositiva con fondo, un vídeo y una imagen de cabecera.

En esta sección se puede definir el orden y el contenido de cada una de la diapositivas para una presentación. Mínimo, una diapositiva debe contener un *número de página* y un *fondo*. El orden en el que se definan las diapositivas no es relevante, ya que éstas se ordenarán

de menor a mayor siguiendo el número de página anteriormente citado. En cuanto al fondo, éste puede ser un objeto del tipo **Imagen**, **Vídeo** o **Captura**. Una diapositiva puede tener cuantos objetos quiera, ya que el límite está en la capacidad que tenga el computador donde se esté ejecutando la aplicación en un momento dado. Por ejemplo, habrá computadores que permitan tener varios vídeos reproduciendo a la vez y otros que solo aguanten un solo vídeo.

Los parámetros base que deben aparecer en una diapositiva son los siguientes:

- **page**: Este parámetro establece el orden en el que aparecerá la diapositiva en la presentación. El valor no tiene porqué seguir el de la diapositiva anterior o preceder al de la siguiente, puede ser cualquier tipo de valor **entero**, ya que será el parser el que, revisando estos valores, establezca un orden entre las diapositivas. Si este valor no aparece, el parser dará un error informando de que falta el valor de número de página de la diapositiva X.

- **background_image/video/capture**: Establece el fondo de la diapositiva. Este fondo puede ser una imagen, un vídeo o una captura. Por defecto, a cada fondo se le debe indicar los parámetros base que acepte cada uno de los objetos del tipo **SlideObject**, salvo por la posición ya que ésta se establece a **[“-1.0:1.0”, “1.0:1.0”, “1.0:-1.0”, “-1.0:-1.0”]**, es decir, pantalla completa. A continuación, se muestra un ejemplo de cada una de ellas:

```
slides:
{
   {page: 1, background_image: "001.png"},
   {page: 2, background_video: "video.mp4"},
   {page: 3,
        background_capture:
        {
            {
                program: "qCROM",
                values_to_send:
                {
                  {key: "Left", value: "Left"},
                  {key: "Up", value: "Up"},
                  {key: "Down", value: "Down"},
                  {key: "Right", value: "Right"}
                }
            }
        }
    }
}
```

B.3.1 Objetos

Como ya se ha dicho, los objetos se dividen en **Imágenes**, **Vídeos** y **Capturas**. La característica que comparten es que son renderizables y se les puede asignar una posición en la pantalla. Por esto, la **Música** no se cuenta como un objeto de diapositiva, pero esto no quita que una diapositiva no pueda tener una o varias pistas de audio reproduciendo a la vez en un momento dado. Al igual que con los vídeos y las capturas, el límite lo establecería el computador donde se esté ejecutando la aplicación en un momento dado.

Posiciones

```
position: ["_:_", "_:_", "_:_", "_:_"]
```

Figura B.2: Ejemplo de posición de una imagen en pantalla.

Una posición es un parámetro que debe aparecer obligatoriamente en todos los objetos de una diapositiva, ya que los objetos se renderizan en dichas posiciones predefinidas por el usuario. Una posición se representa como un **rectángulo**, y para definirlo se debe indicar las esquinas del mismo. El fondo de una diapositiva también es un objeto, salvo por la peculiaridad de que su posición abarca toda la pantalla, por lo que no requiere de que se defina una posición específica.

Los valores por esquinas son pares X:Y, donde tanto X como Y pueden tomar valores entre -1.0 y 1.0. Se pueden definir rectángulos que no tengan la misma resolución que la imagen, vídeo o captura que contengan, ya que las *funciones de reajuste* que tiene la aplicación *qCROM* adaptarán el contenido a dicho rectángulo, añadiendo **letterbox** o **pillarbox** transparentes.

Jugando con las *funciones de reajuste* y una posición, por ejemplo, se puede centrar imágenes de forma fácil y sencilla. Si se desea centrar una imagen verticalmente y alinearla a

la derecha horizontalmente sobre un fondo, se podría utilizar la siguiente posición: **[“0.0:1.0”, “1.0:1.0”, “1.0:-1.0”, “0.0:-1.0”]**. Con esto se estaría seleccionando toda la parte derecha del fondo. Al decirle a la aplicación que ahí se va a colocar una imagen con unas dimensiones distintas a las establecidas por la posición, se llamará una de las *funciones de reajuste*, las cuales añadirán *pillarbox* o *letterbox* invisibles, ajustando la imagen a la posición y logrando el objetivo buscado.

Figura B.3: Ejemplo de combinación las funciones de reajuste y posición.

B.3.2 Imágenes

```
images:
{
    {image_src: " ", position: ["_:_", "_:_", "_:_", "_:_"]},
    {image_src: " ", position: ["_:_", "_:_", "_:_", "_:_"]},
    ...
}
```

Una diapositiva puede tener tantas imágenes superpuestas al fondo como el usuario desee. Sólo requiere que se le indique el nombre de la *imagen* y una *posición* en la pantalla. Esta posición se reajusta en función del fondo, por lo que los valores de -1.0 a 1.0 de X e Y son en función al 100 % del ancho y alto del objeto que haga de fondo. Esto pasa igual con los vídeos y con las capturas. Un ejemplo sería querer poner una imagen sobre un fondo con unas dimensiones de 1024x768. Si este fondo se encuentra en una pantalla con una resolución de 1920x1080, el fondo se centrará y quedarán dos columnas negras a los lados del fondo (*pillarbox*). Al querer poner una imagen sobre el fondo, los valores que pueden tomar las esquinas de la posición de la imagen/vídeo/captura siguen siendo de -1.0 a 1.0, pero el 100 %

que representan dicho valores se aplica a 1024x768 en vez de a la resolución de la pantalla, en este caso 1920x1080.

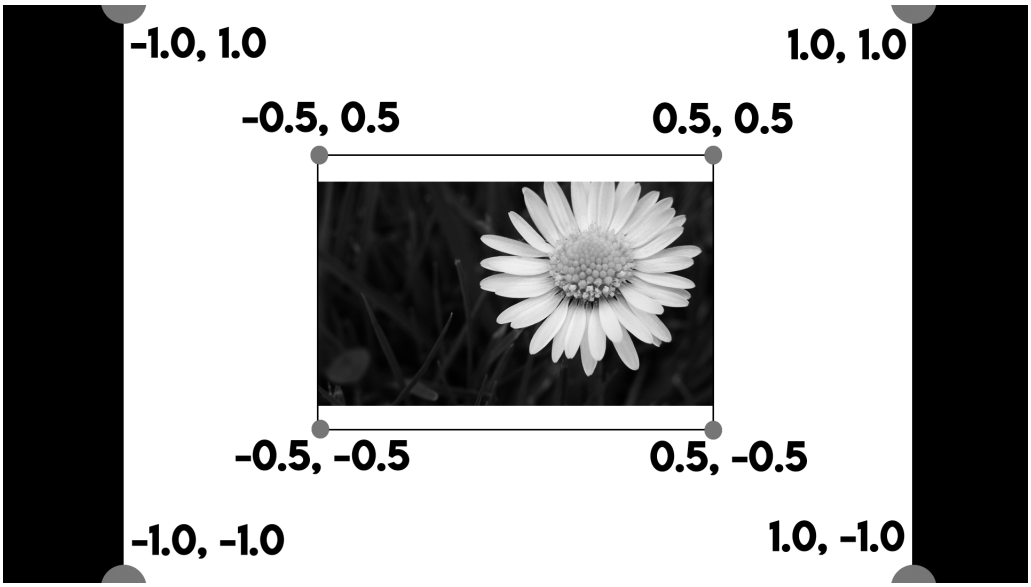


Figura B.4: Ejemplo de posición de una imagen en pantalla reajustada.

En cuanto al nombre de la imagen, esta se establece dando la ruta donde se encuentra dicha imagen en el computador. Esta ruta puede ser absoluta ("/home/usuario/....") o relativa, con la peculiaridad de que la relativa parte de la ruta establecida en la configuración para las imágenes (*images_dir*) o, por defecto, del directorio actual de ejecución.

B.3.3 Vídeos

```
1  videos:
2  {
3     {video_src: " ", loop: "on", volume: 0.5,
4          position: ["_:_", "_:_", "_:_", "_:_"]},
5     {video_src: " ", loop: "off", volume: 0.25,
6          cut_zone: [0, 100, 100, 0]},
7     ...
8  }
```

Una presentación puede tener tantos vídeos como el usuario desee, pero por diapositiva sólo podrá haber tantos como el computador pueda procesar a la vez. Al igual que las imágenes, un vídeo sólo requiere de una ruta de archivo y de una posición para que sea un objeto válido para *qCROM*. Además, los vídeos admiten otros parámetros. Dichos parámetros son los siguientes:

- **loop**: Establece en la aplicación si el vídeo se debe repetir en bucle o no. Este valor puede ser "**on**" u "**off**". Por defecto toma el valor de "**off**".

- **volume**: Establece el volumen que debe tener un vídeo determinado. Por defecto toma 1.0, pero se puede forzar a 10.0.

- **cut_zone**: Aplica una zona de recorte al vídeo. Se define mediante 4 valores, los cuales son: arriba (TOP), izquierda (LEFT), derecha (RIGHT) y abajo (BOTTOM), por ese orden. Estos valores se restan al alto y ancho del objeto y devuelve el resultado para ser renderizado. Este parámetro es útil si se quiere quitar algo de los laterales de un vídeo que no se desea que aparezca en la presentación.

B.3.4 Capturas

```
captures:
{
   {program: " ", position: ["_:_", "_:_", "_:_", "_:_"]},
   {program: " ",
    position: ["_:_", "_:_", "_:_", "_:_"],
    values_to_send:
    {
      {key: " ", value: " "},
      {key: " ", value: " "},
      {key: " ", value: " "},
      ...
     }
    },
   ...
}
```

Al igual que los vídeos, una presentación puede tener tantas capturas como el usuario desee. Aquí el límite está en cuantas aplicaciones permita tener en segundo plano el computador mientras se reproduce la presentación. En este caso, ya no se requiere de una ruta a un archivo a parte de la posición en la pantalla. Ahora se pide el nombre del programa que se desea capturar. Este nombre no tiene porqué ser el del programa, la aplicación busca entre todas las ventanas disponibles ejecutándose en segundo plano y elige la **primera** donde el nombre indicado coincida en su totalidad o en parte con el de la ventana, ignorando si se diferencian en mayúsculas o minúsculas. Las aplicaciones/programas que se deseen capturar deben ejecutarse **en el mismo escritorio donde se reproduce la presentación**, independientemente de si estén maximizadas o minimizadas (aunque esta parte dependerá del gestor de ventanas que utilice el usuario), ya que si se encuentran en otro directorio la herramienta que se utiliza para capturar no encontrará dicha ventana.

Una peculiaridad de este tipo de objetos es que pueden redirigir las pulsaciones que se hagan en el teclado al programa que se captura, para ello se debe indicar mediante pares

key-**value**, donde *key* es la tecla que se quiere reenviar al programa (con valor en SDL[1]) y *value* es el valor que tiene la tecla para X11. Los valores de X11 se pueden obtener ejecutando el siguiente comando en la terminal: ***$ dumpkeys -l***. Al final de este anexo se puede encontrar una tabla que relaciona los valores de SDL con los de X11 para cada una de las teclas.

No todos los programas admiten este envío de teclas, por lo que se debe tener en cuenta a la hora de elegir el programa a capturar.

Otro parámetro que admite es la zona de recorte (**cut_zone**), la cual tiene la misma sintaxis y efecto que en los vídeos.

B.3.5 Música

```
music:
{
    {music_src: " "},
    {music_src: " ", volume: " ", loop: " "},
    ...
}
```

Aunque la música no cuente como objeto de una diapositiva, si puede formar parte de ella. Por diapositiva puede haber varias pistas de audio reproduciendo a la vez, y para ello sólo es obligatorio definir la ruta del archivo a reproducir. También admite otros parámetros como son el volumen (*volume*) y el modo bucle (*loop*), los cuales aplican los mismos efectos que a los vídeos.

B.4 TECLAS ESPECIALES

```
special_keys:
{
    ...
}
```

Las *Teclas Especiales* representan la última sección de un archivo CROM, pero esto no indica que se deban definir al final del archivo. Estas teclas permiten al usuario reproducir efectos de sonido o mostrar estados especiales en la aplicación.

Para definir una de estas teclas, es necesario indicar la tecla que se quiere interpretar, el efecto que tendrá dicha tecla y parámetros auxiliares de dichos efectos.

Los efectos que admite la aplicación son:

- *Modo pantalla en blanco* (**white_screen**)

[1] https://wiki.libsdl.org/SDL_Scancode

- *Modo pantalla en negro* (**black_screen**)
- *Modo pantalla en mosaico* (**mosaic_screen**)

Estos modos tienen unas teclas asignadas por defecto en la aplicación, pero el usuario puede cambiarlas en esta sección para, por ejemplo, adaptar un mando de presentaciones al programa.

```
1  {key: "P", key_effect: "black_screen"}
```

A parte de los efectos anteriormente citados, también se pueden definir teclas para reproducir efectos de sonido. En este caso sólo habrá que indicar (aparte de lo citado al principio) la ruta al archivo a reproducir. Esta ruta puede ser absoluta o relativa, partiendo la relativa de la ruta de los sonidos (*sounds_dir*) indicada en la parte de configuración. También permite indicar el volumen de dicho efecto de sonido con el parámetro volumen (*volume*).

```
1  {key: "S", key_effect: "fx", fx_src: "sonido.mp3",
2      volume: 0.35, slides: [5]}
```

Las teclas especiales se pueden aplicar a todas las diapositivas de la presentación o solo en ciertas diapositivas que el usuario elija. Para hacer esto, las teclas especiales admiten el parámetro **slides** que permite indicar el número de la/s diapositiva/s en la/s que se aplicará dicho efecto. Esta lista es una sucesión de enteros de la forma [1, 2, 3, ...], siendo cada uno de estos enteros el número de página indicado en el parámetro *page* de una diapositiva. Si la definición de este parámetro no aparece en una tecla especial, esto le indicará al programa que esa tecla especial se debe aplicar **en todas las diapositivas** de la presentación.

B.5 COMPARACIÓN DE VALORES SDL-X11

En esta sección se facilita una tabla comparativa con los valores de SDL y X11 para las teclas más utilizadas. La mayoría de los valores no cambian de SDL a X11, y dependiendo del teclado puede que ciertas teclas no sean reconocidas (por ejemplo la tecla **ñ**).

Tabla B.1: Comparativa teclas de dirección de valores SDL-X11.

Tecla	Valor SDL	Valor X11
Abajo	Down	Down
Arriba	Up	Up
Izquierda	Left	Left
Derecha	Right	Right

Tabla B.2: Comparativa valores de función SDL-X11.

Tecla	Valor SDL	Valor X11
F1	F1	F1
F2	F2	F2
F3	F3	F3
F4	F4	F4
F5	F5	F5
F6	F6	F6
F7	F7	F7
F8	F8	F8
F9	F9	F9
F10	F10	F10
F11	F11	F11
F12	F12	F12

Tabla B.3: Comparativa valores numéricos SDL-X11.

Tecla	Valor SDL	Valor X11
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9

Tabla B.4: Comparativa valores keypad SDL-X11.

Tecla	Valor SDL	Valor X11
0	Keypad 0	KP_0
1	Keypad 1	KP_1
2	Keypad 2	KP_2
3	Keypad 3	KP_3
4	Keypad 4	KP_4
5	Keypad 5	KP_5
6	Keypad 6	KP_6
7	Keypad 7	KP_7
8	Keypad 8	KP_8
9	Keypad 9	KP_9
+	Keypad +	KP_Add
-	Keypad -	KP_Subtract
Intro	Keypad Enter	KP_Enter
,	Keypad ,	KP_Comma
/	Keypad /	KP_Divide
*	Keypad *	KP_Multiply

Tabla B.5: Comparativa valores letras minúsculas SDL-X11.

Tecla	Valor SDL	Valor X11
a	A	a
b	B	b
c	C	c
d	D	d
e	E	e
f	F	f
g	G	g
h	H	h
i	I	i
j	J	j
k	K	k
l	L	l
m	M	m
n	N	n
ñ	-	-
o	O	o
p	P	p
q	Q	q
r	R	r
s	S	s
t	T	t
u	U	u
v	V	v
w	W	w
x	X	x
y	Y	y
z	Z	z

Tabla B.6: Comparativa valores letras mayúscula SDL-X11.

Tecla	Valor SDL	Valor X11
A	A	A
B	B	B
C	C	C
D	D	D
E	E	E
F	F	F
G	G	G
H	H	H
I	I	I
J	J	J
K	K	K
L	L	L
M	M	M
N	N	N
Ñ	-	-
O	O	O
P	P	P
Q	Q	Q
R	R	R
S	S	S
T	T	T
U	U	U
V	V	V
W	W	W
X	X	X
Y	Y	Y
Z	Z	Z

Tabla B.7: Comparativa resto de valores SDL-X11.

Tecla	Valor SDL	Valor X11
Espacio	Space	space
Retroceso	Backspace	BackSpace
Tabulador	Tab	Tab
Insert	Insert	Insert
Intro	Return	Return
Esc	Escape	Escape
Ctrl Izquierda	Left Ctrl	CtrlL
Ctrl Derecha	Right Ctrl	CtrlR
Shift Izquierdo	Left Shift	ShiftL
Shift Derecho	Right Shift	ShiftR
Alt Izquierdo	Left Alt	Alt
Alt Gr	Right Alt	AltGr
Avance de página	PageUp	Next
Retroceso de página	PageDown	Prior

ANEXO C

ATAJOS DE TECLADO

En este anexo se listan aquellos atajos de teclado establecidos por defecto, tanto para *pCROM* como para *qCROM*.

C.1 ATAJOS DE TECLADO DE pCROM

- **Tab**: Activa el *modo mosaico*. En este modo, los atajos de teclado son:
  - **Enter**: Selecciona una diapositiva y la muestra a pantalla completa.
  - **Tecla de dirección derecha y Av. página**: Pasa a la siguiente diapositiva.
  - **Tecla de dirección izquierda y Re. página**: Pasa a la diapositiva anterior.
  - **Tecla de dirección arriba**: Pasa a la diapositiva superior.
  - **Tecla de dirección abajo**: Pasa a la diapositiva inferior.
- **W**: Activa el *modo pantalla en blanco*. Al pulsar de nuevo la tecla, desactiva este modo.
- **B**: Activa el *modo pantalla en negro*. Al pulsar de nuevo la tecla, desactiva este modo.
- **S**: Guarda el buffer actual de OpenGL como una imagen.
- **M**: Inicia o pausa el media. Se cuenta como media la música, los vídeos y las capturas.
- **C**: Cambia la visibilidad del puntero.
- **F11**: Pone la pantalla en modo pantalla completa. De este modo se consigue salir pulsando Esc.
- **F5**: Reinicia la presentación. No tiene efecto en estados especiales como modo mosaico, pantalla en blanco o pantalla en negro.
- **Tecla de dirección derecha y Av. página**: Pasa a la siguiente diapositiva.
- **Tecla de dirección izquierda y Re. página**: Pasa a la diapositiva anterior.

C.2 ATAJOS DE TECLADO DE qCROM

- **Tecla de dirección derecha y Av. página**: Pasa a la siguiente pregunta.
- **Tecla de dirección izquierda y Re. página**: Pasa a la pregunta anterior.
- **Tecla de dirección arriba**: Muestra el *ranking global* del cuestionario.

ANEXO D

HERRAMIENTAS AUXILIARES

En este anexo se listan todas aquellas herramientas auxiliares que *pCROM* ofrece a los usuarios de la aplicación con el fin de facilitar el despliegue de sus presentaciones.

D.1 PDF EN IMÁGENES

```
$ pcrom -split archivo.pdf -resolution WxH
```

pCROM permite la división de un PDF en las imágenes que lo componen. Además, permite dar una opción para ajustar el alto y un ancho de las imágenes que se van a obtener. Si el flag *-resolution* no se indica, las imágenes resultantes saldrán con la resolución que tuvieran en un principio. Todas las imágenes se almacenarán en una carpeta con el mismo nombre que tenga el PDF en cuestión.

D.2 ARCHIVO CROM BASE

```
$ pcrom -generate_default nombre_del_nuevo_archivo
```

pCROM permite generar un archivo CROM de ejemplo con la base para empezar un nuevo proyecto de presentación. Es útil si no se quiere partir de cero a la hora de crear una nueva presentación.

D.3 ARCHIVO PDF A ARCHIVO CROM

```
$ pcrom -convert archivo.pdf
```

pCROM permite presentar archivos PDF, pero cada vez que se presenta un archivo de este tipo, el programa debe obtener el media asociado al mismo (imágenes). Este proceso se repite cada vez que el archivo PDF se pretende presentar. Por ello, éste se puede pasar a un archivo CROM antes y realizar las presentaciones directamente con el archivo CROM generado.

pCROM también permite no borrar el archivo CROM temporal generado al presentar un PDF directamente, añadiendo la opción *-save* al ejecutar la aplicación. Esta opción guarda el media y el archivo CROM con lo básico para poder arrancar la presentación de forma que se tarde menos que partiendo del PDF.

```
$ pcrom archivo.pdf -save
```

D.4 GENERACIÓN DEL ARCHIVO PDF DE LA PRESENTACIÓN

Para poder generar el archivo PDF asociado a una presentación CROM, es necesario elegir cada una de las diapositivas que se desea que aparezcan en dicho PDF. La aplicación permite guardar el buffer de OpenGL en un momento dado pulsando la **tecla S** (similar a una captura de pantalla). Al hacer esto, la aplicación guarda dicho buffer como una imagen en un directorio temporal que es eliminado una vez la presentación finaliza. Antes de eliminar dicho directorio, se dará la opción de pasar las imágenes guardadas a un archivo PDF. Si el usuario elige la opción de generar el PDF, esté se generará en el directorio actual. En caso contrario, las imágenes guardadas serán eliminadas junto con el directorio temporal.

ANEXO E

ARCHIVOS AUXILIARES DE qCROM

En este anexo, se describirán en detalle los tipos de archivos utilizados por la aplicación *qCROM*, así como temas importantes en relación a su creación.

E.1 ARCHIVOS DE CONFIGURACIÓN

Este tipo de archivo permite a la aplicación de control de *qCROM* conocer de antemano cierta información sobre un determinado cuestionario. Esta información se indica en el archivo mediante el uso de una serie de *palabras reservadas*. Éstas son las siguientes:

- **site**: Indica la localización de la página de *qCROM*.
- **pass**: Indica la contraseña que permiten a las peticiones de la aplicación ser aceptadas por la página. Por defecto, esta contraseña es **q640M**. Esta contraseña debe almacenarse junto a la de administración en la carpeta **misc** de la página de *qCROM*.
- **refresh**: Indica el intervalo de tiempo entre una petición a la página y otra. Por defecto, este valor es **1.0**.
- **quiz_file**: Indica la ruta al archivo con las preguntas. La construcción de este tipo de archivos se puede encontrar en el *Anexo E*.
- **users**: Indica el archivo con los usuarios que utilizarán mandos auxiliares para participar en el concurso. Esta variable es opcional, por lo que si no se tienen usuarios que vayan a participar con este tipo de mandos, esta variable no tiene porque aparecer en el archivo de configuración. La construcción de este tipo de archivos también se puede ver en el *Anexo E*.

E.2 ARCHIVOS DE CUESTIONARIO

qCROM tiene por objetivo dar la capacidad de poder realizar cuestionarios al posible público de una presentación. Los archivos que son utilizados por la aplicación para dicha tarea constan de un *título de cuestionario* y de una *serie de preguntas*. Éstas pueden estar acompañadas de un media, el cual sólo podrá ser una imagen o un vídeo. Un ejemplo de archivo podría ser el que se puede ver a continuación.

```
1  Test de Prueba
2  ~~~
3
4  Última letra del alfabeto griego
5  --
6  + Omega;
7  - Alpha;
8  - Zeta;
9
10 ==
11
12 {}
13
14 ==
15
16 ¿Qué se puede ver en la siguiente imagen?
17 --
18 - Un pájaro;
19 - Un avión;
20 + Un hombre v...¿volando?;
21
22 [prueba.png]
```

Figura E.1: Ejemplo de archivo con preguntas.

Como se puede observar, existen ciertos caracteres especiales que delimitan las secciones e indican al parser de que tipo es cada una de ellas. Estas secciones pueden ser:

- **Título del cuestionario**: La sección con el título del cuestionario debe indicarse al principio del documento y no es opcional. Para indicar que es la sección del título, es necesario subrayar éste con tres virgulillas consecutivas. De esta forma se le estará indicando al parser que esa sección contiene el título y no otra cosa.

```
1  Test de Prueba
2  ~~~
```

Figura E.2: Ejemplo de título de un archivo de cuestionario.

- **Elemento del cuestionario**: El otro tipo de sección que puede aparecer en este tipo de archivo se delimitan con un par de iguales (=) al final de las misma. Dentro de este tipo de sección pueden aparecer dos cosas: *una pregunta junto con sus respuestas y un posible media*, o *una zona de espera*.

– **Las Preguntas** constan de un texto asociado a la pregunta en cuestión, una serie de respuesta (hasta seis) y de un posible media. Para indicar el texto de la pregunta, éste debe ser subrayado con un par de guiones (-). En caso de las respuestas, éstas deberán empezar por el símbolo **+** o por el símbolo **-** y terminar por un punto y coma (**;**). Sólo una de ellas deberá empezar por el símbolo **+**, ya que de esta forma se le indicará al parser que esa es la respuesta correcta. Por último, las preguntas pueden tener una imagen o un vídeo asociado a la misma, el cual se indicará entre corchetes al final de la sección de la pregunta. El media asociado a una pregunta debe estar localizado en la carpeta con la aplicación web de *qCROM*, ya que será en esa carpeta en la cual la aplicación buscará los archivos. Dicha carpeta tiene por nombre **quiz_media** y se encuentra situada como una subcarpeta de la raíz de la aplicación web. En caso de no existir, ésta deberá ser creada por el usuario.

```
1 Es la <strong>bebida alcohólica</strong> más importante de Japón
2 --
3 - Umeshu;
4 + Sake;
5 - Awamori;
```

Figura E.3: Ejemplo de pregunta con etiquetas **HTML**.

En los textos asociados a la pregunta o a sus respuestas pueden aparecer **etiquetas HTML** (similares a ****), las cuales serán aceptadas por el parser e interpretadas por la aplicación web que muestra las preguntas.

– **Los Descansos** permiten hacer paradas entre preguntas de forma que el ponente pueda mostrar una nueva tanda de diapositivas antes de seguir con el cuestionario. Estos descansos se le indican al parser mediante un par de corchetes dentro de una sección.

E.3 ARCHIVOS DE USUARIO

```
1 #      User ID          | Clicker ID |
2 # ----------------------|------------|
3 Jose Adolfo Nosecual     | 649DD3
4 Manuel Adolfo Nosequien | 649CE5
5 ...                      | ...
```

Figura E.4: Ejemplo de archivo con usuarios.

Los archivo de usuario son utilizados por *qCROM* para hacer una precarga de usuarios a la base de datos. Los usuarios que se permiten cargar de esta forma serán aquellos que vayan a utilizar mandos auxiliares durante la presentación. La aplicación *qCROM* se ha diseñado para ser compatible con los mandos auxiliares de la marca **TurningTechnologies**, la cual asocia

un *identificador* a cada mando, recibiéndose la información de las pulsaciones en forma de valores **ID**-**Valor**.

La creación de este tipo de archivos es sencilla. Para agregar un usuario basta con añadir una nueva linea al archivo en la cual se deberá escribir el nombre del usuario y el identificador del mando, separando ambos mediante una barra vertical (|). También se permiten comentarios y lineas vacías, con el fin de hacer el documento más legible para el usuario. Para que una linea se cuente como comentario, ésta deberá empezar por almohadilla (**#**).

Figura E.5: Mando auxiliar compatible con *qCROM* creado por **TurningTechnologies**.

ANEXO F

EJEMPLOS DE USO

En este anexo se facilitan una serie de ejemplos completos de presentaciones multimedia creadas por y para la plataforma. Estos ejemplos se dividen en función de los objetivos que se intentan alcanzar utilizando un determinado tipo de objeto de diapositiva. Por ejemplo, crear una presentación multimedia que permita tener vídeos tanto a pantalla completa como a modo de objeto.

F.1 PRESENTACIONES MULTIMEDIA CON ARCHIVOS CROM

El uso de archivos CROM en la plataforma permite realizar una serie de acciones que no se pueden conseguir mediante el uso de archivos PDF, ya que con éstos lo único que se puede presentar son imágenes estáticas con transiciones a corte. Por ese motivo, se han creado una serie presentaciones para que sirvan de ejemplo ante ciertas necesidades que puedan surgir a la hora de crear una presentación con esta plataforma.

Estos ejemplos se dividen en función de la complejidad de la presentación:

- **Archivos de presentación básicos**: Este ejemplo se puede caracterizar por utilizar solamente imágenes estáticas para la presentación, junto algún que otro audio o efecto de sonido.

```
configuration:
{
  fps_limit: 30,

  images_directory: "media/images",
  sounds_directory: "media/sounds"
}
```

  Como se puede observar, se hace uso de una configuración muy simple en la que sólo se indica la tasa de imágenes por segundo junto a los directorios que contienen las

imágenes y los sonidos de la presentación. La tasa de imágenes por segundo se establece en 30 debido a que no existen vídeos o capturas que requieran de una tasa de imágenes superior. Bajando esta tasa se consigue que el programa pese menos para el sistema.

Con un total de cinco diapositivas, la complejidad en este ejemplo radica en el uso de una pista de audio en la dispositiva número 4, y un efecto de sonido en la diapositiva 5. La pista de audio tiene definido el volúmen al que se desea reproducir, al igual que el efecto de sonido. En cambio, la pista de audio no tiene definido un valor para su repetición en bucle, por lo que esta pista solamente se reproducirá una única vez.

```
1  slides:
2  {
3    {page: 1, background_image: "001.png"},
4    {page: 2, background_image: "002.png"},
5    {page: 3, background_image: "003.png"},
6    {page: 4, background_image: "004.png",
7      music:
8      {
9        {
10         music_src: "music.mp3",
11         volume: 0.70
12       }
13     }
14   },
15   {page: 5, background_image: "005.png"}
16 }
17
18 keys:
19 {
20   {key: "E", key_action: "fx", fx_src: "fx_effect.wav", volume:
         0.40, slides: [5]}
21 }
```

Atendiendo a la sección de teclas especiales se puede observar como se define una tecla para la reproducción de un efecto de sonido. Mediante la variable *key_action* se establece el tipo de efecto, y mediante la variable *slides* la cantidad de diapositivas sobre la que ésta será interpretada como tal. Si esta última variable no se indicara, la tecla sería interpretada en todas las diapositivas de la presentación.

- **Archivos de presentación medios**: En este caso, se han catalogado como archivos de presentación de dificultad media aquellos que entre sus diapositivas se pueden encontrar vídeos como objetos o como fondo a pantalla completa. Los vídeos no tienen mucha más complejidad de la que puede presentar un pista de audio, pero si requieren de una posición en pantalla, de un posible recorte de alguno de sus laterales, etc. Además, en la configuración hay que tener en cuenta ciertos aspectos que pueden afectar de manera

directa a como se puedan mostrar los vídeos en pantalla, ya que si no se ajusta la tasa de imágenes por segundo a los vídeos, estos pueden verse con pequeños saltos entre los fotogramas que los componen.

```
configuration:
{
  fps_limit: 60,

  fx_default_volume: 0.15,
  music_default_volume: 0.25,
  video_default_volume: 0.35,

  music_default_loop: "on",

  images_directory: "media/images",
  videos_directory: "media/videos",
  sounds_directory: "media/sounds"
}
```

La configuración ha tomado un papel más importante al establecer ciertos valores por defecto que permiten ahorrar líneas de código a la hora de definir cada una de las diapositivas. En este caso, los valores han permitido determinan los volúmenes por defecto para efectos de sonido, música y vídeo. Además, también se ha establecido el modo de reproducción de las pistas de audio para que estas se reproduzcan en modo bucle (*music_default_loop: "**on**"*).

```
slides:
{
  {page: 1, background_image: "001.png"},
  {page: 2, background_video: "video_example1.mp4"},
  {page: 3, background_image: "002.png"},
  {page: 4, background_image: "003.png",
    music:
    {
      {music_src: "music.mp3"}
    }
  },
  {page: 5, background_image: "004.png",
    videos:
    {
      {
        video_src: "video_example2.mp4",
        loop: "on",
        position: ["-0.05:-0.05", "0.81:-0.05", "0.81:-0.91",
            "-0.05:-0.91"]
      }
    }
```

```
21      },
22      {page: 6, background_image: "005.png"},
23      {page: 7, background_image: "006.png",
24        music:
25        {
26          {music_src: "music2.mp3"}
27        },
28        videos:
29        {
30          {
31            video_src: "video_example3.mp4",
32            cut_zone: [120, 0, 0, 0],
33            position: ["-0.05:-0.05", "0.81:-0.05", "0.81:-0.91",
                  "-0.05:-0.91"]
34          }
35        }
36      }
37    }
```

En este caso, para las diapositivas han definido un total de siete. En la diapositivas número dos se puede encontrar un vídeo a pantalla completa. La peculiaridad de este tipo de vídeos es que su valores de volumen y repetición vienen de la configuración global, por lo que si en esta configuración no se establecen dichos valores, éstos tomarán los que el reproductor les asigna por defecto.

En las diapositivas cuatro y siete se pueden encontrar diferentes pistas de audio, donde su volumen y su modo de repetición están definidos en la configuración, evitando así tener que definirlos uno a uno por pista de audio. Por último, en la diapositiva cinco y siete también se pueden encontrar vídeos a modo de objetos de diapositiva. El de la diapositiva cinco está definido para reproducirse en modo bucle, mientas que la peculiaridad del vídeo de la diapositiva siete es que recorta 120 unidades en su parte superior, por lo que se muestra un vídeo recortado del original que sólo se reproduce una vez.

```
1   keys:
2   {
3     {key: "E", key_action: "fx", fx_src: "fx_effect.wav", slides:
          [3, 6]},
4     {key: "N", key_action: "black_screen"}
5   }
```

Por último, en la sección de teclas especiales podemos encontrar dos elementos. El primero de ellos permite la reproducción de un efecto de sonido en las dispositivas tres y seis, mientras que en segundo permite la activación del modo de pantalla en negro, el cual es interpretado desde cualquier diapositiva al no definir la variable *slides*. Este

modo de pantalla en negro puede aplicarse en esta presentación de dos formas distintas, mediante el uso de la tecla por defecto o mediante el uso de la tecla especial. Se podría acceder a dicho modo pulsando cualquiera de esas dos teclas, y se podría salir de él de la misma manera, ya que al reproductor lo que le importa es el efecto y no la tecla que lo origina.

- **Archivos de presentación avanzados**: En esta última categoría, se ha creado un ejemplo que permite utilizar capturas como fondo de la aplicación y como objeto de una diapositiva, permitiendo además un tipo de transición distinto hasta los ahora utilizados. Además, permite la definición de un archivo de cabecera para ser mostrado sobre todas las diapositivas de las presentación.

```
configuration:
{
  fps_limit: 60,

  transition_default_value: "slide",

  header_file: "media/header.png",
  header_default_value: "on",

  images_directory: "media/images",
  sounds_directory: "media/sounds"
}
```

En este caso, la configuración es sencilla. En ella se cambia la transición por defecto a *slide* y se establecen las variables relacionadas con el archivo de cabecera. Estas son la localización del archivo a mostrar, junto al valor por defecto de esta variable en cada una de las diapositivas.

```
slides:
{
  {page: 1, background_image: "001.png"},
  {page: 2, background_capture:
    {
      {
        program:"blender",
        values_to_send:
        {
          {key: "G", value: "G"},
          {key: "R", value: "R"},
          {key: "S", value: "S"},
          {key: "Right", value: "X"},
          {key: "Left", value: "Y"},
          {key: "Up", value: "Z"},
          {key: "0", value: "0"},
```

```
17              {key: "5", value: "5"},
18              {key: "9", value: "9"},
19              {key: "Space", value: "Return"},
20            }
21          }
22        }
23       },
24      {page: 3, background_image: "002.png"},
25      {page: 4, background_image: "003.png",
26        music:
27        {
28          {music_src: "music.mp3"}
29        }
30      },
31      {page: 5, background_image: "004.png",
32        captures:
33        {
34          {
35            program: "Twitter",
36            position: ["-0.05:-0.05", "0.81:-0.05", "0.81:-0.91",
                  "-0.05:-0.91"]
37          }
38        }
39      }
40  }
```

En esta segunda parte del archivo CROM de ejemplo, se definen un total de cinco diapositivas, de las cuales sólo dos disponen de una captura a modo de objeto o de fondo. De estas dos capturas, la más interesante es la de la dispositiva número dos. En esta diapositiva, la captura se muestra como fondo a pantalla completa y permite redirigir una serie de teclas. Al redirigirse las teclas de dirección (izquierda y derecha), para poder salir de la diapositiva actual hará falta pulsar la combinación de **Shift+Dirección** con el fin de que ese evento no sea redirigido al programa capturado.

BIBLIOGRAFÍA

[1] A. Munshi, D. Ginsburg and D. Shreiner: *OpenGL(R) ES 2.0 Programming Guide*. Addison-Wesley Professional, 2008.

[2] Adobe: *Archivos PDF*. Disponible en URL `https://acrobat.adobe.com/es/es/why-adobe/about-adobe-pdf.html`, 2016.

[3] B. Stroustrup: *The C++ Programming Language*. Addison-Wesley Longman Publishing Co. Inc., 2013.

[4] Barry W. Boehm: *Software engineering - as it is*. IEE Proceedings, 1976.

[5] Cameron Newham, J. Vossen, Carl Albing and Jp Vossen: *Bash Cookbook: Solutions and Examples for Bash Users*. O'Reilly Media Inc., 2007.

[6] Carina Soledad González González, Alberto Mora Carreño: *Técnicas de gamificación aplicadas en la docencia de Ingeniería Informática*. Disponible en URL `http://www.aenui.net/ojs/index.php?journal=revision&page=article&op=viewArticle&path%5B%5D=152&path%5B%5D=290`, 2015.

[7] CCM: *El protocolo HTTP*. Disponible en URL `http://es.ccm.net/contents/264-el-protocolo-http`, 2016.

[8] D. Shreiner, G. Sellers, B. Licea Kane and J.M. Kessenich: *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Addison-Wesley Professional, 2013.

[9] David Vallejo, Cleto Martín: *Desarrollo de Videojuegos. Un enfoque Práctico. Volumen 1. Arquitectura del Motor. Páginas 111-152.* CreateSpace Independent Publishing Platform, 2015.

[10] Erich Gamma, Richard Helm, Ralph Johnson John Vlissides: *Patrones de Diseño*. Addison-Wesley, 1995.

[11] Gamma, Erich *et al.*: *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995, ISBN 0-201-63361-2.

[12] Garr Reynolds: *Presentation Zen: Simple Ideas on Presentation Design and Delivery*. New Riders, 2008.

[13] Ivar Jacobson, Grady Booch, James Rumbaugh: *The Unified Software Development Process*. Addison-Wesley, 1999.

[14] James Paul Gee: *What digital games have to teach us about learning and literacy*. Palgrave Macmillan, 2007.

[15] Joaquín Mª López Muñoz: *Boost en 5 minutos*. Disponible en URL `http://www.arcos.inf.uc3m.es/~cpp-day/wp-content/uploads/2013/06/boost.pdf`, 2013.

[16] John J. Medina: *Exprime tus neuronas: 12 reglas básicas para ejercitar nuestra mente*. EDICIONES GESTION 2000, 2011.

[17] José Luis Ramírez Cogollor: *GAMIFICACIÓN*. RC LIBROS (SC LIBRO), 2014.

[18] Kevin Werbach, Dan Hunter: *For the win: How game thinking can revolutionize your business*. Wharton Digital Press, 2012.

[19] Lorna Jane Mitchell: *Secure Coding in C and C++*. O'Reilly Media Inc., 2015.

[20] María del Carmen Gómez Fuentes: *Notas del curso Bases de Datos*. Disponible en URL `http://www.cua.uam.mx/pdfs/conoce/libroselec/Notas_del_curso_Bases_de_Datos.pdf`, 2013.

[21] Microsoft: *Comparación de los sistemas de archivos NTFS y FAT*. Disponible en URL `http://windows.microsoft.com/es-es/windows-vista/comparing-ntfs-and-fat-file-systems`, 2006.

[22] Nancy Duarte: *Slide:ology*. O'Reilly Media Inc., 2008.

[23] Richard E. Mayer: *The instructive animation helping students build connections between words and pictures*. Disponible en URL `http://visuallearningresearch.wiki.educ.msu.edu/file/view/mayer+%26+anderson+(1992).pdf`, 1992.

[24] Richard E. Mayer: *Multimedia Learning*. Cambridge University Press, 2012.

[25] Richard E. Mayer: *Research-based principles for designing multimedia instruction*. Disponible en URL `http://hilt.harvard.edu/files/hilt/files/background_reading.pdf`, 2014.

[26] Robert C. Seacord: *PHP Web Services*. Packt Publishing Ltd., 2013.

[27] SCRUM: *Qué es SCRUM*. Disponible en URL `https://proyectosagiles.org/que-es-scrum`.

[28] Shaun Mitchell: *SDL Game Development*. Addison-Wesley Professional, 2013.

[29] Susan M. Weinschenk: *100 things every presenter needs to know about people*. New Riders, 2012.

[30] Tomás Escudero Escorza, Ana Delia (coords.) Correa Piñero: *Investigación en innovación educativa: algunos ámbitos relevantes*. La Muralla, 2012.

[31] William Ricardo: *Programación Modular*. Disponible en URL `http://progmodular.blogspot.com.es/`, 2012.