Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Escuela de Ingeniería y Arquitectura
Universidad Zaragoza

Proyecto fin de Carrera
Ingeniería en Informática
Curso 2011/2012

Diseño y desarrollo de una pizarra software para el entrenamiento táctico de deportes de equipo. Aplicación al fútbol sala

Alberto Bolsa Blasco

Directores: Pedro Álvarez y Sandra Baldassarri

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Junio de 2012

Diseño y desarrollo de una pizarra software para el entrenamiento táctico de deportes de equipo. Aplicación al fútbol sala

RESUMEN

Hoy en día la mayoría de los entrenadores de clubs deportivos utilizan pizarras y rotuladores para describir a sus jugadores los ejercicios o jugadas que deben realizar durante los entrenamientos o partidos. Este método tiene una limitación clara, los ejercicios o jugadas se pierden una vez se borra la pizarra. Ésto supone un problema ya que los mismos ejercicios deben explicarse muchas veces durante la misma temporada, lo cual implica dibujar repetidas veces un ejercicio.

En este proyecto se ha desarrollado una PIZARRA SOFTWARE que subsana este problema y añade una serie de funcionalidades que dotan a la aplicación de mucha flexibilidad, adaptando un sistema tradicional a las nuevas tecnologías.

La pizarra desarrollada permite crear nuevos ejercicios o jugadas y almacenarlos en el repositorio de la aplicación. Dichos ejercicios se pueden reproducir, editar, borrar o exportar desde la propia aplicación. Para la creación y edición se dispone de una paleta de elementos de representación apropiados para cada deporte. Esta paleta puede ser actualizada desde la propia aplicación, añadiendo nuevos paquetes de elementos y deportes. La reproducción se produce en forma de animación con las opciones habituales de los reproductores de vídeo. La exportación de ejercicios se puede realizar en formato vídeo, animación o pdf. Además, también se pueden importar nuevos ejercicios desde ficheros externos.

Adicionalmente se han desarrollado una serie de herramientas que complementan la funcionalidad básica de la aplicación. Se dispone de un grabador de audio que se sincroniza con la pizarra y permite añadir una pista de sonido explicativa del ejercicio. La pizarra permite incluir imágenes a través de una libreta de jugadores para ser utilizados como elementos en los ejercicios. Se ha desarrollado también un protocolo que comunica la pizarra con un servidor web para exportar ejercicios y así ser vistos desde un navegador a través de prototipos de visor, también creados en el contexto del proyecto. El proyecto se completa con asistente de creación de nuevos paquetes de deportes y un editor de temas, ambos desarrollados como aplicaciones web.

El diseño de la aplicación se ha basado en varios patrones de diseño: Modelo-Vista-Controlador, Arquitectura N-Capas, Delegación o Composición, así como metodología de diseño UML. Para la implementación del proyecto se ha utilizado el lenguaje de programación Actionscript 3, sobre la plataforma de ejecución Adobe AIR y el sistema Adobe Flash. Además se han desarrollado algunos componentes en Java2EE, HTML, javascript(+ AJAX) y PHP.

Como resultado del proyecto, se ha implementado una herramienta que cumple con todos los requisitos marcados inicialmente. Ésta resulta de utilidad para las personas para las que ha sido desarrollada, estando preparada para futuras ampliaciones o cambios.

Agradecimientos

Agradecimientos

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El presente documento es la memoria del proyecto fin de carrera "Diseño y desarrollo de una pizarra software para el entrenamiento táctico de deportes de equipo. Aplicación al fútbol sala", realizado por Alberto Bolsa y codirigido por Pedro Javier Álvarez y Sandra Baldassarri.

En este primer capítulo se introduce el contexto general del proyecto, así como las motivaciones para realizar el mismo. Posteriormente, se realiza un breve análisis de los objetivos generales del proyecto y se explica la terminología que es seguida a lo largo del resto del documento. Finalmente, se describe la estructura de la memoria.

1.1. Contexto y Motivación

Habitualmente, los entrenadores utilizan pizarras convencionales para explicar los distintos ejercicios que sus jugadores deben realizar en un entrenamiento o las jugadas que deben ejecutar durante un partido. Este método es común en un amplio abanico de disciplinas deportivas, por ejemplo, fútbol, fútbol sala, baloncesto, balonmano, etc.

No obstante, este método tiene una limitación clara: los ejercicios o jugadas explicadas se pierden una vez que el entrenador borra la pizarra. No conservar esta información es una limitación del sistema tradicional. Las acciones explicadas suelen redibujarse de la misma manera muchas veces a lo largo de la temporada. Los jugadores deben memorizar las jugadas y los ejercicios sobre el limitado tiempo en el que el dibujo está en la pizarra. El entrenador debe recordar la gran cantidad de ejercicios programados, en más de una sesión para poder utilizarlos en los entrenamientos o partidos.

Por lo tanto, la utilización de una aplicación que permita crear, editar, almacenar, reproducir, importar y exportar ejercicios permitiría subsanar estas limitaciones así como automatizar algunas de las tareas comunes de un entrenador. Además de estas

características principales, una serie de funcionalidades como compartir ejercicios, crear documentos con los ejercicios, generar vídeos o importar nuevos deportes, dotarían de mayor flexibilidad a la aplicación.

Durante el desarrollo del sistema y en su futura implantación se contará con el Apoyo del club RED STARS, club de fútbol sala con sede en Zaragoza, Sala "estación de Servicio Sástago", equipo amateur de fútbol sala, Halle Fussbal Braunschweig, y FiveStars, equipo ilerdense de paintball que participa en la máxima competición europea.

1.2. Objetivos generales del proyecto

El objetivo principal de este Proyecto Fin de Carrera es diseñar e implementar una pizarra software orientada a deportes de equipo que facilite a los entrenadores la labor de elaborar y explicar ejercicios o jugadas a través de medios más avanzados que los que se utilizan tradicionalmente.

El entrenador que utilice el sistema desarrollado lo hará a través de un portátil o un tablet PC durante los entrenamientos, donde tiene una mayor utilidad, o los partidos de su equipo. Alternativamente podrá también utilizarlo desde un computador de sobremesa para crear las jugadas y luego utilizarlas desde otros dispositivos portátiles como los mencionados anteriormente.

La característica principal de la aplicación será la creación de jugadas o ejercicios a través de un interfaz gráfico. Estos ejercicios o jugadas constituirán la base para el resto de las funcionalidades de la aplicación.

En la representación de un ejercicio deben considerarse un amplio abanico de elementos: jugadores propios y contrarios, pelota, conos, vallas, cuerdas, etc. Algunos de estos elementos se mueven durante la ejecución del ejercicio, principalmente, los jugadores y la pelota. El ejercicio resultante representará estos desplazamientos así como las acciones que realizan los distintos elementos a lo largo de la duración del ejercicio. Cada uno de los distintos deportes disponibles tendrá su propio conjunto de elementos representativos. Los deportes con sus elementos podrán ser importados desde la propia aplicación.

Estos ejercicios se podrán reproducir a través de un visor con las opciones habituales de un reproductor de vídeo: iniciar, pausar, avanzar escenas, etc. Así mismo dicha animación podrá estar acompañada de un audio que explique sus detalles más significativos y que se reproducirá de manera sincronizada con la representación visual del ejercicio.

Por otro lado, la pizarra también integrará: un repositorio de jugadas almacenadas y las operaciones necesarias para su gestión, consulta (búsquedas simples) y reproducción; la posibilidad de crear documentos PDF[4] que contengan la descripción y explicación de uno o más ejercicios y/o jugadas; edición de los ejercicios

almacenados en el repositorio y, opcionalmente, un mecanismo de copias de seguridad a nivel de ejercicios y/o jugadas.

Como aplicaciones independientes de la pizarra, se desarrollarán: un grabador de audio, que se sincronizará automáticamente con la aplicación para añadir las grabación a la escena que se esté editando o creando; un visor de la pizarra para poder ser utilizado en una página web; un creador de paquetes, que agilizará la creación de nuevos deportes para importarlos a la aplicación y un editor de temas, que hará más fácil la creación de nuevas plantillas para la interfaz de usuario de la aplicación.

1.3. Terminología

En este apartado se definen una serie de conceptos necesarios para la correcta interpretación de esta memoria.

Los elementos clave de la aplicación son los **ejercicios y las jugadas**, ambos se podrían definir como una secuencia ordenada de acciones, durante un periodo de tiempo que deben realizar los jugadores para llegar a cabo un determinado objetivo, éste es el que diferencia los dos conceptos.

- **Ejercicio:** Secuencia de acciones que deben realizar uno o varios jugadores para lograr un objetivo concreto en el marco de un entrenamiento. Habitualmente en estas acciones se emplea material deportivo típico de este tipo de entrenamiento (conos, vallas, pivotes, redes, balones, etc.).
- **Jugada:** Secuencia coordinada de acciones que los jugadores de un equipo deben realizar para conseguir un objetivo concreto en el marco de un partido.

En el contexto de la aplicación se van a utilizar como las acciones que realizan un conjunto de jugadores a lo largo de un periodo discreto de tiempo. Estos ejercicios como resultado producirán un vídeo, una animación o un documento.

Éstos son los componentes fundamentales de los ejercicios o jugadas.

- **Elemento:** Objeto o persona que interviene en un ejercicio. Los elementos pueden ser estáticos (conos, porterías, etc.), asociados a una posición fija, ó dinámicos (jugadores, balones, etc.) aquellos que pueden moverse.
- **Movimiento:** Desplazamiento de un elemento de una posición a otra, puede ser de distintos tipos (zigzag, circular, continuo, discontinuo, etc.), y este movimiento está descrito por una trayectoria de 1 o más puntos.
- **Trayectoria:** Camino seguido por un elemento dentro de un movimiento a través de 1 ó más puntos, dependiendo del tipo de movimiento, estos puntos definen un movimiento u otro.

En la Figura 1.1 se puede ver el movimiento de tipo zig-zag continuo, de un elemento (cuadrado, que en fútbol sala representa un jugador local) a través de una trayectoria de 3 puntos.

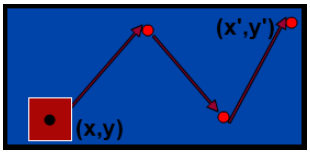


Figura 1.1: Movimiento con una trayectoria zig-zag

Figura 1.1: Movimiento de ejemplo.

- **Escena:** Conjunto de movimientos que intervienen en un ejercicio durante un determinado periodo de tiempo. Es el conjunto de movimientos realizados a partir de un estado inicial.

En la Figura 1.2 se representa una escena con dos movimientos, uno de tipo circular continuo y otro de tipo circular discontinuo.

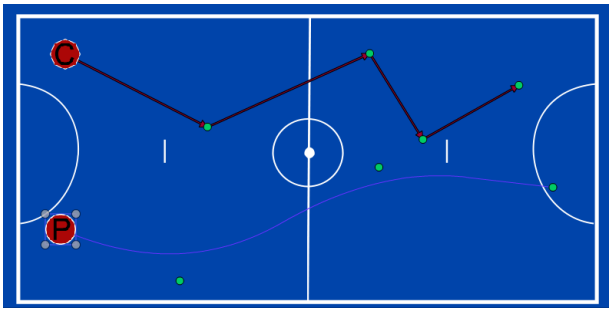


Figura 1.2: Representación de una escena

De entre los distintos resultados que produce la aplicación cabe destacar dos que visualmente son iguales, pero internamente son muy diferentes.

- **Vídeo:** Fichero de imágenes que se puede reproducir con cualquier reproductor de vídeo.
- **Animación:** Conjunto de secuencias generadas con la aplicación y que sólo son reproducibles con pizarra o complementos de la misma.

Estas dos formas de reproducción se diferencian en que los vídeos se reproducen de manera externa a la pizarra, y las animaciones de manera interna.

En la Figura 1.3 se puede apreciar la relación entre los distintos elementos.

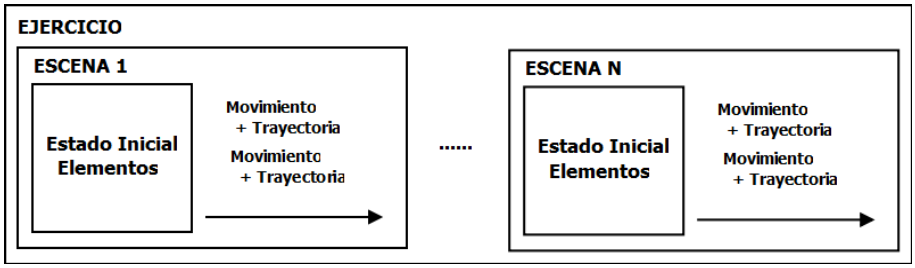


Figura 1.3: Relación entre los distintos términos

El ejercicio está compuesto por varias escenas. Cada una de las escenas está definida por un estado inicial de cada uno de los elementos, y cada uno de estos elementos realiza un movimiento de un determinado tipo con una trayectoria concreta, lo que lo posiciona en el estado inicial de la siguiente escena.

1.4. Estructura del documento

Este documento está dividido en dos partes: memoria y anexos. La primera parte explica de manera concisa las distintas fases del desarrollo del proyecto y sus resultados, conforme a la siguiente estructura.

1. **Introducción:** Se establece el punto de partida y el contexto de la aplicación a muy alto nivel, explicando las motivaciones para realizar la misma, los objetivos y una serie de conceptos clave necesarios para entender bien el resto del documento.

2. **Análisis:** En este capítulo se describe en detalle el problema abordado. Después se expone el estudio de algunas aplicaciones comerciales que resuelven parcialmente este problema que permite ver los puntos fuertes y débiles de las mismas. A partir de la información obtenida con la descripción del problema y el estudio de las aplicaciones existentes se establecen los objetivos y los correspondientes requisitos funcionales y no funcionales de la solución desarrollada en el marco de este proyecto. Después se expone un estudio acerca de las posibles tecnologías para la implementación del sistema, y se detallan las herramientas utilizadas. Toda la labor de análisis se resume en este capítulo.

3. **Diseño:** Partiendo del análisis expuesto en el capítulo anterior, se describe primero el entorno de ejecución de la aplicación y se identifican los distintos tipos de usuarios. Posteriormente se expone la arquitectura del sistema, describiendo los tres niveles en los que se ha basado este diseño, para después hacer un análisis más profundo de cada uno de ellos. El siguiente apartado aborda el diseño de la base de datos y las relaciones existentes entre las distintas entidades. Por último, se expone el diseño del interfaz de usuario.

4. **Resultados:** Una vez finalizada la implementación de la solución, en este capítulo se explican las funcionalidades de la aplicación resultante. Posteriormente se presenta una evaluación de la aplicación desde el punto de vista del usuario y de la propia aplicación.

5. **Gestión del Proyecto:** En esta sección se expone la metodología empleada para el desarrollo del proyecto, la planificación inicial y real, un pequeño estudio de riesgos del proyecto, una estimación de los esfuerzos realizados y para terminar, las herramientas de gestión que se han utilizado a lo largo del ciclo de vida del proyecto.

6. **Conclusiones:** En este último capítulo se presenta una reflexión del trabajo llevado a cabo a lo largo de todo el proyecto, extrayendo varias conclusiones, tanto técnicas como personales. Finalmente se exponen algunas vías de trabajo futuro basándose en la aplicación realizada.

La segunda parte del documento incluye los Anexos, con información más detallada de algunos apartados de la memoria (Análisis, Diseño, Pruebas), el manual de usuario de la aplicación, los resultados de los tests de usuario y la bibliografía utilizada.

Capítulo 2

Análisis del Sistema

En este capítulo primero se describe en detalle el problema abordado. Después se expone el estudio de algunas aplicaciones comerciales que resuelven parcialmente este problema. Del estudio previo se deducen los puntos fuertes y débiles de estas aplicaciones. A partir de la información obtenida con la descripción del problema y el estudio de las aplicaciones existentes se establecen los objetivos y los correspondientes requisitos funcionales y no funcionales de la solución desarrollada en el marco de este proyecto. Finalmente se presenta un estudio acerca de las posibles tecnologías para la implementación del sistema y se detallan las herramientas utilizadas. Toda la labor de análisis se resume en este capítulo.

2.1. Descripción del problema

El problema abordado en este proyecto proviene de una necesidad que surge durante los entrenamientos de tácticas en deportes de equipo y durante el desarrollo de los correspondientes partidos, ya que hoy en día, la mayoría de los entrenadores de clubs deportivos utilizan pizarras y rotuladores para describir a sus jugadores los ejercicios o jugadas que deben realizar. Este método tiene una limitación clara, los ejercicios o jugadas se pierden una vez se borra la pizarra.

Que no se conserven los ejercicios implica que el entrenador debe dibujar el ejercicio cada vez que necesita explicárselo a sus jugadores. Los jugadores deben memorizar algunos de los ejercicios y jugadas, y deben hacerlo a través de la pizarra durante el tiempo en el que están dibujados, este tiempo es en la mayoría de los casos, limitado.

Una vez se ha dibujado en la pizarra, se pierde la noción de la secuencia en la que han sucedido los eventos. Si se desea plasmar ejercicio o jugada compleja, el dibujo en una pizarra convencional no queda tan claro al haber cruces de líneas y solapamiento de dibujos.

El número de jugadas y ejercicios de cualquier equipo suele ser de un tamaño lo suficientemente grande como para que el entrenador tenga que memorizarlas todas.

2.2. Aplicaciones Existentes

Actualmente existen distintas aplicaciones comerciales que permiten la creación de ejercicios y jugadas de una manera digital. De ellas se han seleccionado dos de las que más se asemejan a la solución que se desea construir, por sus características, teniendo en cuenta los puntos fuertes y débiles de las mismas, que luego servirán como guía a la hora de realizar el proyecto. Además de estas dos aplicaciones se han estudiado otras, Skaut notebook [7], SportCode sport office [9] o Dartfish team pro [8].

Para contextualizar se va a estudiar los deportes a los que es aplicable el sistema, cuál es la funcionalidad básica que ofrece la aplicación y el coste de la misma.

2.2.1. Easy Animation

Easy animation [5] es la aplicación más completa de las contempladas en este estudio. Dispone versiones de la misma para numerosos deportes como fútbol, balonmano, baloncesto, voleybol, hockey o hockey sobre hielo. Los elementos se deben posicionar en unidades discretas de tiempo, lo que implica, por ejemplo realizar las trayectorias circulares completamente a mano, y hace que el usuario pierda de alguna manera el concepto de escena o paso como tal.

La aplicación permite editar cualquier tipo de ejercicio, con mayor o menor dificultad, con una interfaz sencilla y bonita, sin embargo el sistema está compuesto por excesivo módulos adicionales que le añaden la funcionalidad de manera poco homogénea. La aplicación no permite introducir audio ni exportar a un servidor remoto la información de los ejercicios creados. Cada uno de estos deportes se venden por separado a un precio aproximado de 150$, la unidad.

Permite exportar los ejercicios en vídeo, pero también se trata de un módulo de pago adicional, con un coste de 150$. La aplicación está muy cuidada estéticamente, y permite el cambio de los modelos de los elementos del deporte del que se disponga, también como un paquete adicional de pago.

En la Figura 2.1 se puede ver una instantánea de la aplicación.

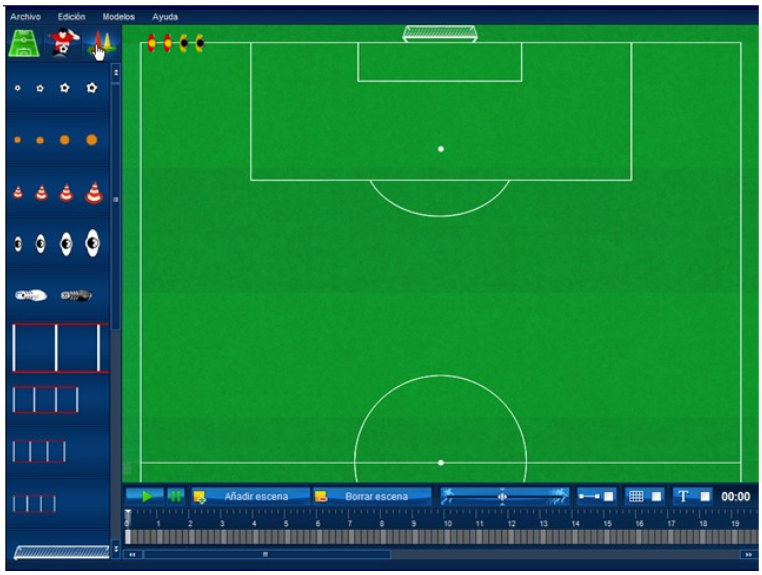


Figura 2.1: Easy animation

2.2.2. Jes-soft Playbook

La aplicación Jes-soft Playbook [6], al igual que la anterior es independiente para cada deporte de los disponibles, entre los que se encuentran fútbol, baloncesto, o fútbol americano entre otros.

Es bastante más simple que la anterior, ya que no cuenta con una línea del tiempo, y el desplazamiento entre escenas se realiza mediante una caja con el número de la misma. El concepto es el mismo, un editor de ejercicios para un deporte concreto, con una interfaz estilo windows.

Dependiendo del deporte las distintas posibilidades de representación de líneas que ofrece la aplicación se adaptan, las posibilidades de configuración no son muchas, y la disposición de los elementos hace que ocupe el mismo espacio en pantalla el panel de edición y las opciones, haciendo la paleta de dibujo muy pequeña. Una aplicación sencilla que permite crear, editar y visualizar ejercicios simples, sin mayor posibilidad de exportación o ampliación de deportes o elementos sobre el mismo sistema instalado.

El precio de la aplicación es de 134.50$ por licencia y deporte. En la Figura 2.2 se puede ver una instantánea de la aplicación.

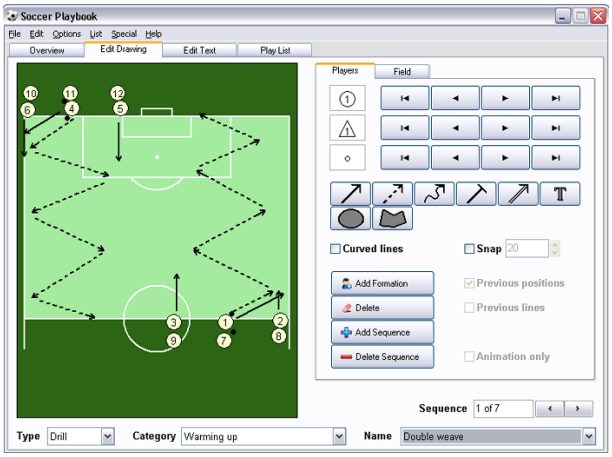


Figura 2.2: Jes-soft Soccer playbook

2.2.3. Conclusiones

Tanto las aplicaciones sobre las que se ha profundizado en los puntos 2.2.1 y 2.2.2, como el resto de las estudiadas tienen los mismos problemas: son aplicaciones disponibles para varios deportes, pero a través de la instalación de distintas aplicaciones, es decir se podrían considerar específicas para un deporte en concreto.

No permiten la interacción con otros sistema ajenos a la propia aplicación, sus posibilidades a la hora de exportar los ejercicios creados son muy limitadas. No permiten exportar las jugadas en documentos pdf, ni en formato flv(excepto easy-animation), ni disponen de visores web a través de los que se permita compartir los ejercicios creados con los otros jugadores.

Ninguna de las aplicaciones incluye el modo pizarra tradicional en el que se guarden los trazos dibujados por el entrenador, ni permiten la importación de audios descriptivos. Los sistemas de creación de las aplicaciones estudiadas resultan complicados para el usuario no experto. Además son aplicaciones con un elevado coste económico.

2.3. Requisitos del sistema

Sabiendo las características propias de la aplicación que se quiere realizar y tras estudiar cómo funcionan otras aplicaciones comerciales, conociendo sus puntos fuertes y débiles, se puede hacer una especificación concreta de los requisitos de la pizarra a desarrollar.

El sistema se desarrollará en base a una serie de objetivos clave: la creación de los ejercicios y jugadas, las transformaciones y reproducciones sobre los mismos y algunas funcionalidades añadidas al sistema.

La funcionalidad base de la aplicación es la creación de los ejercicios, que se realiza a través de la definición una secuencia ordenada de escenas. En este proceso se pueden añadir distintos elementos como jugadores, balones, material deportivo, conos, etc. A estos elementos se les podrá asignar distintos movimientos y trayectorias dentro de cada escena, inicialmente líneas rectas, zigzag, curvas, y cada una de ellas tanto continuas como discontinuas. Dentro de las distintas escenas de la aplicación se podrá definir el tipo de movimiento de la animación ya sea movimiento concurrente de todos los elementos o movimiento en serie de los mismos. También se podrá añadir un audio explicativo de la escena a través de la misma aplicación o grabarlo e incluirlo directamente mediante la sincronización entre la aplicación y el grabador de audio también desarrollado en el proyecto.

Los ejercicios creados, se podrán editar, exportar o borrar después de su creación. Las posibilidades de exportación de esta versión son variadas para poder comunicarse con distintas plataformas y usuarios, se podrán exportar los ejercicios en ficheros pizarra, pdf, ficheros de vídeo o subirlos a un servidor web a través de la misma aplicación.

La aplicación permitirá la exportación en vídeo convencional y orientado a reproductores de Internet, por lo tanto el formato escogido para este apartado es flash vídeo (.flv) [10], los vídeos generados en este formato podrán ser reproducidos desde gran variedad de reproductores de vídeo disponibles tanto para PC, MAC, como GNU/Linux. Por otra parte, también se podrán generar animaciones propias de la aplicación, y que se utilizarán tanto para reproducir las secuencias como para importarlos al sistema, para ello se utilizarán ficheros de pizarra (.piz). Por último es interesante generar una versión imprimible de los ejercicios, que se podrá guardar, enviar a los jugadores o generar libros de jugadas, para ello se podrán exportar los ejercicios en formato documento portable (.pdf).

Se podrán importar ejercicios desde el formato de la aplicación (.piz), así como instalar paquetes(.zip) con nuevos deportes o actualizar los ya existentes.

Una vez creado o importado un ejercicio la aplicación permitirá guardarlo, editarlo y reproducirlo, tanto hacia delante como hacia atrás, pararlo o pausarlo, así como pasar escenas completas. Esta reproducción se podrá realizar a través del reproductor de animaciones de la aplicación, que permitirá configurar algunos de los parámetros de la visualización, con el reproductor de vídeo, y con el reproductor web.

Para llegar hasta los ejercicios deseados, ya sea para reproducirlos o exportarlos, la aplicación contará con un sencillo sistema de búsqueda a partir de cualquiera de los campos descriptivos del ejercicio.

Como una de las funcionalidades extra del sistema se desarrollará una libreta de jugadores a través de la cual se podrán generar las descripciones de los mismos. Esta libreta, permitirá la captura de las fotos de los jugadores, ya sea a través de la cámara del computador o a través de una foto externa. Para obtener sólamente la cara de

los jugadores se realizará reconocimiento facial sobre las distintas imágenes.Una vez obtenidas las caras de los jugadores, se podrán utilizar los mismos como elementos de la aplicación a la hora de crear un ejercicio.

De cara a posibles futuras ampliaciones y la elaboración del sistema basado en web se desarrollará un protocolo de comunicación con un servidor. Este protocolo se basará en un servicio web basado en la librería nuSoap[13] que permitirá el envío de información de forma segura entre la aplicación y el servidor además de pequeños paquetes de actualización de propiedades. También se portará el reproductor desarrollado y un nuevo reproductor basado en HTML, Javascript y CSS.

En el apartado A.1 Requisitos completos de la aplicación, se incluye la especificación completa de los requisitos de la aplicación.

2.4. Análisis de tecnologías y decisiones

Para la elaboración del proyecto se ha realizado un estudio previo de las posibles tecnologías para implementar el mismo. Este estudio se ha basado en tres de las que se consideraron más adecuadas en el momento inicial: Adobe Flex [15], Adobe Flash [14] y Java [16].

Los parámetros de comparación son muy diversos y se basan en aspectos relativos a la eficiencia, el resultado final y la mantenibilidad del sistema, entre otros.

Como resultado de este estudio se realizó una presentación a los directores del proyecto con las conclusiones obtenidas del estudio, y se tomó una decisión basada tanto en valores subjetivos, como en un método objetivo y formal de decisión aplicado sobre las distintas tecnologías. El sistema de decisión se basa en factores críticos que todas las alternativas deben cumplir, y en factores no críticos, a los cuales se les asigna una puntuación, y mediante la aplicación las fórmulas del criterio (Cuadro A.3) se obtiene una valor numérico para cada una de las alternativas, siendo el menor, el teóricamente más adecuado.

Finalmente, la opción escogida fue Adobe Flash sobre AIR ya que cumplía todas las necesidades planteadas en los requisitos, para la realización de las animaciones proveía de un soporte nativo de gran ayuda, además ser multiplataforma y facilita su relación con el mundo de la Web. Este sistema se describe de manera resumida en el Apartado 2.4.1.

Se pueden ver los resultados completos del estudio en A.2 Estudio de Tecnologías.

2.4.1. Adobe Flash + AIR

Adobe Flash (anteriormente llamado Macromedia Flash) es una aplicación en forma de estudio de animación que trabaja sobre fotogramas, destinada a la producción y entrega de contenido interactivo para las diferentes audiencias multiplataforma.

Los archivos de Flash, que tienen generalmente la extensión de archivo SWF, pueden aparecer en una página web para ser vista en un navegador, o pueden ser reproducidos independientemente por un reproductor Flash. Los archivos de Flash aparecen muy a menudo como animaciones en páginas Web y sitios Web multimedia, y más recientemente Rich Internet Applications [19] mediante Adobe AIR.

AIR es un entorno de ejecución versátil, ya que permite que el código Flash, HTML o JavaScript existente se reutilice para construir programas tradicionales de escritorio. Adobe lo define como un entorno de ejecución sin navegador para que Rich Internet Applications se puedan desplegar en el escritorio, en lugar de un framework de aplicaciones. Las diferencias entre cada paradigma de despliegue proporciona a ambas ventajas y desventajas. Por ejemplo, una Rich Internet Applications desplegada en un navegador no requiere instalación, mientras que una desplegada con AIR requiere que sea empaquetada, firmada digitalmente, e instalada en el sistema de archivos local de los usuarios.

Estas son las características que hacen de AIR un candidadato ideal para este sistema, ya que permite un almacenamiento de datos local rápido, y permite la generación de un instalador que hace que la implantación en el sistema del usuario sea sencillo.

AIR actualmente tiene cuatro maneras de trabajar con datos:

- Servidor de base de datos a través de servicios web
- Archivo local de XML
- Base local de datos de SQLite enviadas con AIR
- Almacenamiento de cifrado local incluido con AIR

Lo cual dota a los sistemas creados sobre AIR de una gran flexibilidad en cuanto a almacenamiento local se refiere.

2.4.2. Herramientas

En esta sección se enumeran y definen otras herramientas software complementarias utilizadas para la realización de las aplicaciones que componen el proyecto

- **Adobe Flash CS4** [14] como compilador general para el proyecto, y herramienta para generar la interfaz de usuario.
- **Adobe Flash CS5.5** como compilador general para el proyecto, y herramienta para generar la interfaz de usuario.
- **Adobe Flash CS6** como compilador general para el proyecto, y herramienta para generar la interfaz de usuario.

- **Netbeans 6.8** [20] como compilador de Java para la generación del reproductor de audio y prueba de tecnologías.
- **Xampp** [26] como servidor web y motor de bases de datos MySQL.
- **phpMyAdmin** [27] como editor web de bases de datos MySQL.
- **SQLite Database Browser** [28] como editor de bases de datos SQLite.
- **Notepad++** [33] como editor general de ficheros de texto.
- **Adobe Flex** [34] para las pruebas tecnológicas.

Capítulo 3

Diseño de la Aplicación

Partiendo del análisis expuesto en el capítulo anterior, se describe primero el entorno de ejecución de la aplicación, sus entradas y salidas, así como sus posibles usuarios. Posteriormente, se expone la arquitectura del sistema, describiendo los tres niveles en los que se ha basado este diseño, para después hacer un análisis más profundo de cada uno de ellos. El siguiente apartado aborda el diseño de la base de datos. Luego se expone el diseño del interfaz de usuario. Por último se expone el diseño técnico basado en la tecnología empleada para la implementación del proyecto.

3.1. Entorno de la aplicación

La aplicación podrá instalarse y ejecutarse en cualquier computador con los sistemas operativos Windows, MAC OS X o GNU/Linux, que disponga del Runtime Adobe AIR [17]. El diagrama de despliegue del sistema se expone en la Figura 3.1.

En el sistema existen dos tipos de usuarios bien definidos en relación con las actividades que pueden realizar: por un lado, los **entrenadores**, que son los encargados de la gestión de los ejercicios, y los **jugadores**, que son los usuarios fineles la información generada con la pizarra.

El entrenador puede crear y modificar los ejercicios a través de la pizarra y posteriormente exportar la parte que desee a un fichero externo, que será el que utilicen los jugadores. El entrenador es también el encargado de la actualización del sistema (elementos, ejercicios o deportes), a partir de los ficheros correspondientes, o de información que él mismo hubiera exportado utilizando la pizarra.

El jugador utiliza la información generada con la aplicación, ya sea viendo la animación o el vídeo generado con la pizarra, o estudiando la jugada a través de un documento pdf obtenido del repositorio de la aplicación.

En la Figura 3.1 se puede apreciar un diagrama de despliegue de los distintos sistemas. La aplicación puede ser utilizada tanto por el entrenador, para generar información o actualizar el sistema, como por los usuarios para consumir esta información. La aplicación es capaz de sonectarse con un sistema remoto para consumir información y enviar los ejercicios al sistema remoto, que a su vez servirá esta información a través del navegador.

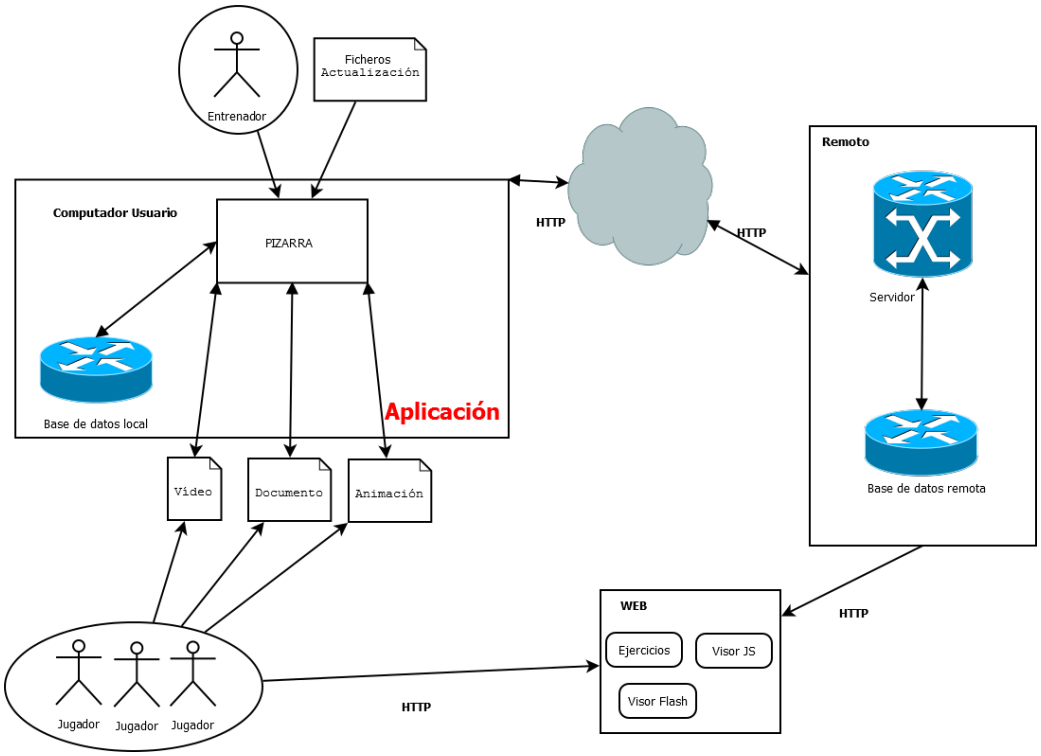


Figura 3.1: Diagrama de Despliegue

Para exportar la información a un servidor remoto, y así poder ser visualizada a través de intenet, se utilizan llamadas *Http* con un formato específico. Cuando el servidor recibe esas llamadas, escribe la información correspondiente a los ejercicios en la base de datos remota, y responde con el resultado de la transacción. A partir de esta información remota, se puede acceder vía Web a los ejercicios exportados por un usuario y visualizarlos en el visor html.

La aplicación utiliza para su funcionamiento una base de datos local en SQLite[28], de la que se extrae la información almacenada de los ejercicios, la información de los deportes, los elementos, las preferencias, etc. El funcionamiento y diseño de dicha base de datos se explica en el apartado 3.4.

3.2. Arquitectura del sistema

En esta sección se procederá a la descripción detallada del diseño de la aplicación de manera funcional y técnica.

3.2.1. Diseño funcional

A partir de la toma de requisitos de la aplicación, que se detalla en la seccion A.1, se procedió a ala definición de las distintas máscaras de las que consta la aplicación a alto nivel. Este diseño se realizó en base a wireframes[30], que se muestran en la sección B.2 de los anexos.

Una vez detallada la funcionalidad de la aplicación a nivel de interfaz gráfica de usuario, así como la interacción con el mismo, se comenzó con un diseño funcional de alto nivel e independiente de la tecnología. Este diseño a alto nivel se realizó siguiento el patrón **Modelo-Vista-Controlador(MVC)** [31]. En este modelo se separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres niveles distintos.

- **Modelo:** Contiene los datos y la lógica para manejar el estado de la aplicación. Es responsable de responder acerca del estado de la aplicación y actualizarse en los cambios de estado.

- **Vista:** Presenta la interfaz de usuario y el estado de la aplicación en pantalla. Es responsable de la comunicación con el usuario.

- **Controlador:** Gestiona las entradas del usuario para cambiar el estado de la aplicación. Es responsable de determinar cómo las vistas responden a la interacción con el usuario.

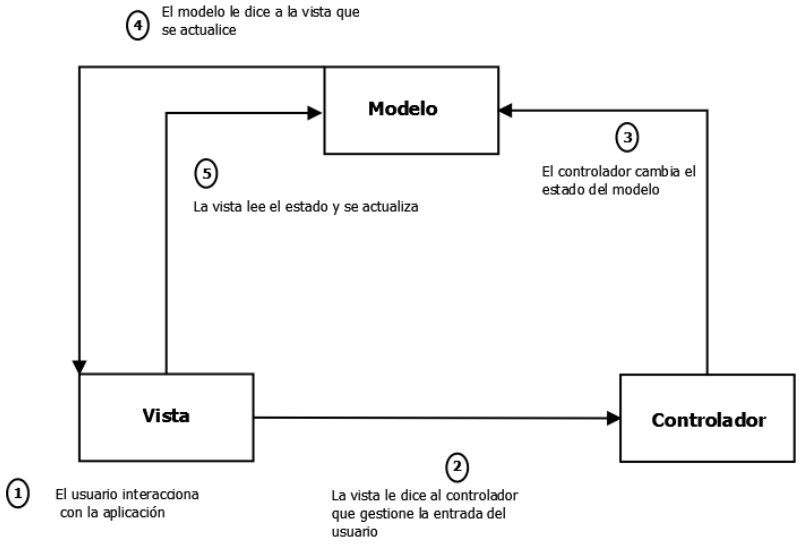


Figura 3.2: Funcionamiento MVC + Observer

La ventaja del patrón MVC radica en la separación anteriormente nombrada sin solapar las responsabilidades de cada capa.

A su vez se ha utilizado el patrón **Observer** [32], que define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. Esto permitirá implementar de una manera sencilla un sistema en el que muchos componentes dependen de los eventos de otros componentes.

En la Figura 3.2 se puede observar el funcionamiento del patrón, de la manera en la que se utilizará en el proyecto y la interacción entre las distintos elementos. La relación entre el modelo y la vista realizara a través de las suscripciones del patrón observer anteriormente citado, por lo tanto es una comunicación indirecta.

Esta arquitectura de la aplicación se compone de tres niveles distintos: la **interfaz** gráfica de usuario, los **componentes** software de la aplicación y la **base de datos**.

Para el diseño de la *interfaz de usuario* se han utilizado diagramas de navegación que se convirtieron en prototipos software que se sometieron a distintas pruebas de usabilidad y de validación. Como se puede ver en la parte superior de la figura 3.3, esta parte se sometió a varias iteraciones de pruebas, tanto de cliente como de usuarios, hasta conseguir la navegación y disposición más adecuados. Esta capa se corresponde con las distintas vistas y subvistas del patrón MVC, esta parte se describe con más detalle en la Sección 3.5.

El *núcleo del sistema* se diseñó mediante un diagrama de componentes. El objetivo de este diagrama fue descomponer las distintas funcionalidades de la aplicación para cumplir todos sus objetivos. Posteriormente, cada componente fue refinado a

nivel de diseño en los correspondientes diagramas de clases mostrados en el Anexo B. Esta parte encapsula las acciones a realizar en los controladores del patrón MVC. La sección 3.3 profundiza en este apartado.

La capa inferior del sistema es la *base de datos*, como se puede observar en la figura. Esta parte se diseñó mediante un esquema entidad relación que representa los datos sobre los que funciona la aplicación. Para la implementación de dicha base de datos se transformó dicho esquema en Relaciones, y finalmente se convirtió dichas relaciones en tablas SQL que se insertarán en el SGBD utilizado por la aplicación, en este caso MySQL. En la Sección 3.4 se detalla este diseño.

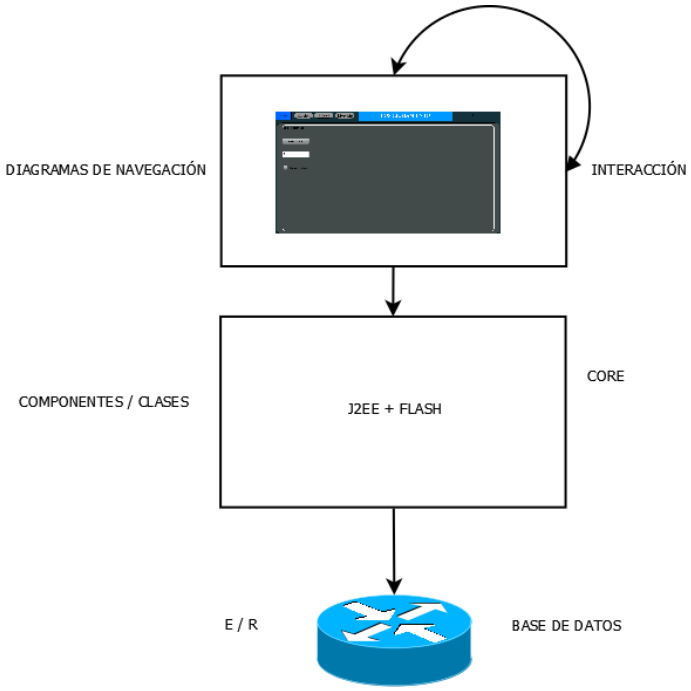


Figura 3.3: Esquema de alto nivel del sistema completo

Cada una de estas capas se describe en detalle en los siguientes apartados del capítulo así como en el Anexo B.

3.3. Diseño de componentes de la aplicación

Para describir la estructura funcional a alto nivel de la aplicación se ha utilizado un diagrama de componentes. Este diagrama describe elos componentes de la aplicación y las dependencias existentes entre los mismos.

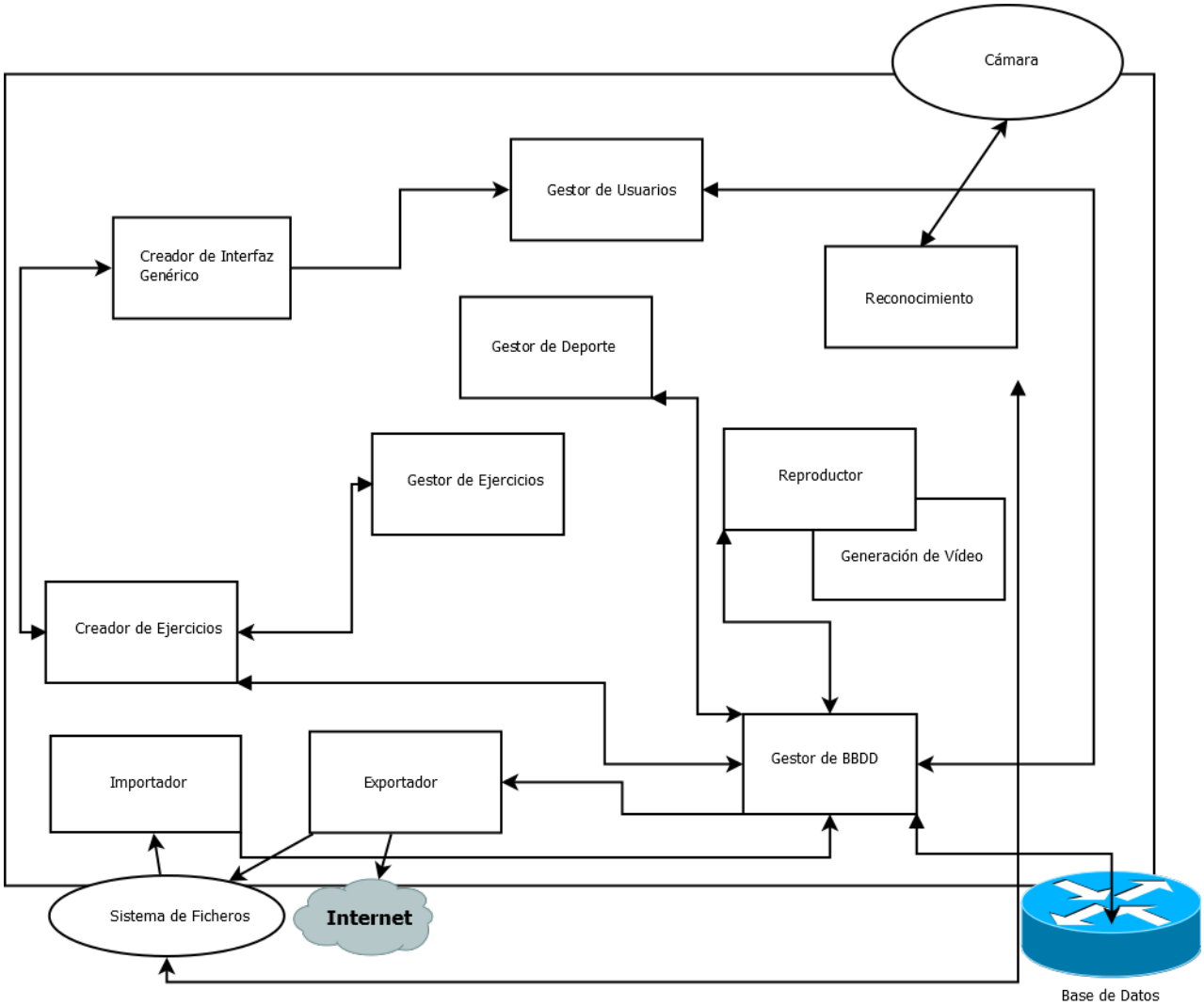


Figura 3.4: Diagrama de componentes

En la Figura 3.4 se puede observar dicho diagrama, en el que cada caja representa una un componente funcional del sistema completo, mientras que las flechas que unen las cajas son las interacciones entre los componentes. Una representación inicial a este nivel tiene sentido ya que se trata de un sistema complejo con funcionalidades muy distintas entre sí, y a partir de este diagrama, se podrá descomponer cada una de ellas como un subproblema más pequeño y diseñarlas utilizando los diagramas de clases y estados de UML. Estos diagramas UML utilizarán clases comunes a otros componentes, que serán reutilizadas.

En la Figura 3.4 se puede ver que los componentes interaccionan con varios sistemas externos: una cámara web, una conexión a Internet, la base de datos y el sistema de ficheros local. Estos elementos están situados son externos al sistema de información desarrollado y han sido representados en óvalos.

En la parte inferior de la Figura 3.4 se pueden ver los componentes relacionados con la gestión de datos. Por un lado, se necesitan componentes que sean capaces de *importar* distintos tipos de información desde el sistema de ficheros del computador. También se necesita un componente que *exporte* la información desde la pizarra al computador del usuario (sistema de ficheros) o a algún servidor remoto (Internet). Como fuente y destino de la información de los anteriores componentes se encuentra la *base de datos* de la aplicación. Por lo tanto se necesita un componente que se encargue de la interacción con dicha base de datos. Este gestor de Base de Datos será usado por otros componentes para la interacción con el repositorio de información.

En la parte izquierda de la Figura 3.4 se encuentran los componentes de *gestión de ejercicio*s y de *creación de ejercicios*. La gestión se encarga de obtener los ejercicios de la base de datos y ponerlos en el formato adecuado para su manipulación, mientras que la creación de ejercicios se encarga de todo lo relacionado con las herramientas que se proporcionan al usuario para dibujar un nuevo ejercicio.

En la parte derecha de la figura se encuentran los componentes de *reproducción* y *generación de vídeo*. Han sido representados como dos componentes solapados ya que la gestión de vídeo se apoyará en la reproducción para producir el fichero de vídeo resultante.

El componente de *reconocimiento* se encarga de, a partir de la cámara o una imagen, generar un fichero con la cara de un jugador, que posteriormente podrá ser utilizado dentro del editor de ejercicios.

Además de estos componentes, se incluyen algunos más que realizan tareas generales de gestión de los elementos de la aplicación, como el *gestor de usuarios*, que permite ver la información de las personas que utilizan la aplicación, y el *gestor de deportes*, que permite la administración de los distintos paquetes de ejercicio instalados en el sistema.

Por último, en la esquina superior izquierda, se ve el componente *Creador de interfaz genérico*, que se encarga de tareas generales relacionadas con la interfaz y la creación de algunos componentes genéricos(Clases base de Vista, inclusión del título, registro de eventos para la navegación de manera genérica). En este componente se apoyarán las vistas y subvistas que se crearán al utilizar el patrón MVC.

Los ejercicios no son sólo almacenados en base de datos. temporalmente también pueden ser alamacenados en el componente Gestor de Ejercicios, que actua como una "caché" de ejercicios. Consideramos relevante describir cómo este componente almacena estos ejercicios. La Figura 3.5 describe la disposición lógica de los elementos que forman un ejercicio.

Los ejercicios forman una jerarquía. Cada ejercicio contiene escenas, que agrupan los movimientos de elementos en un periodo discreto de tiempo. Cada uno de estos movimientos que componen una escena está asociado a un elemento, y describe una trayectoria concreta a través de una seri de puntos.

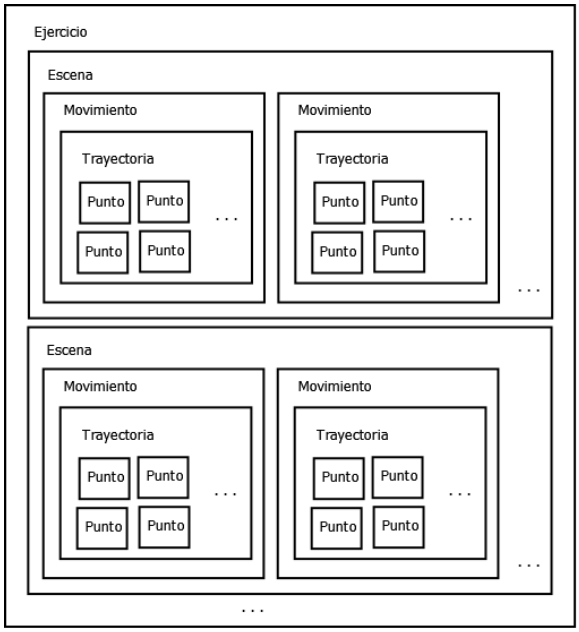


Figura 3.5: Disposición lógica de un ejercicio

Utilizando la agregación de UML, el diagrama de clases de esta parte aparece en la Figura 3.6.

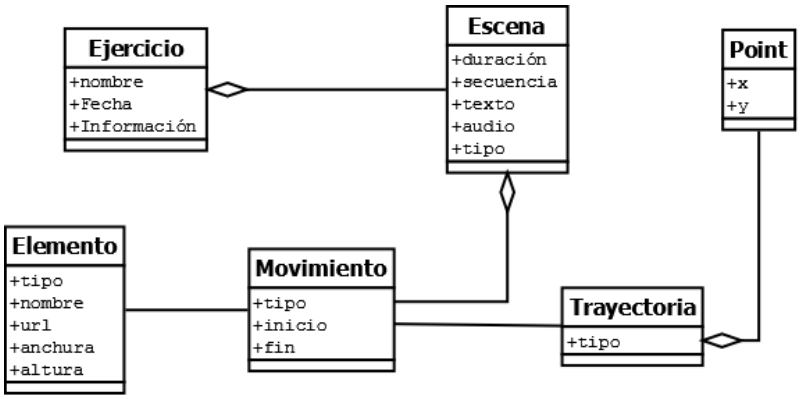


Figura 3.6: Diagrama de clases de ejercicio

Las distintas entidades representadas en el diagrama lógico están encapsuladas en clases, y cada una va agregando los elementos que la componen hasta llegar a los puntos que forman una trayectoria en último lugar. Para caracterizar el elemento que realiza el movimiento, se relaciona la clase elemento con la clase movimiento.

Para una explicación más detallada y el resto de los diagramas ver el Anexo B Diseño del Sistema.

3.4. Diseño de la base de datos de la aplicación

A continuación, la figura 3.7 muestra el diagrama Entidad -Relación de la aplicación.

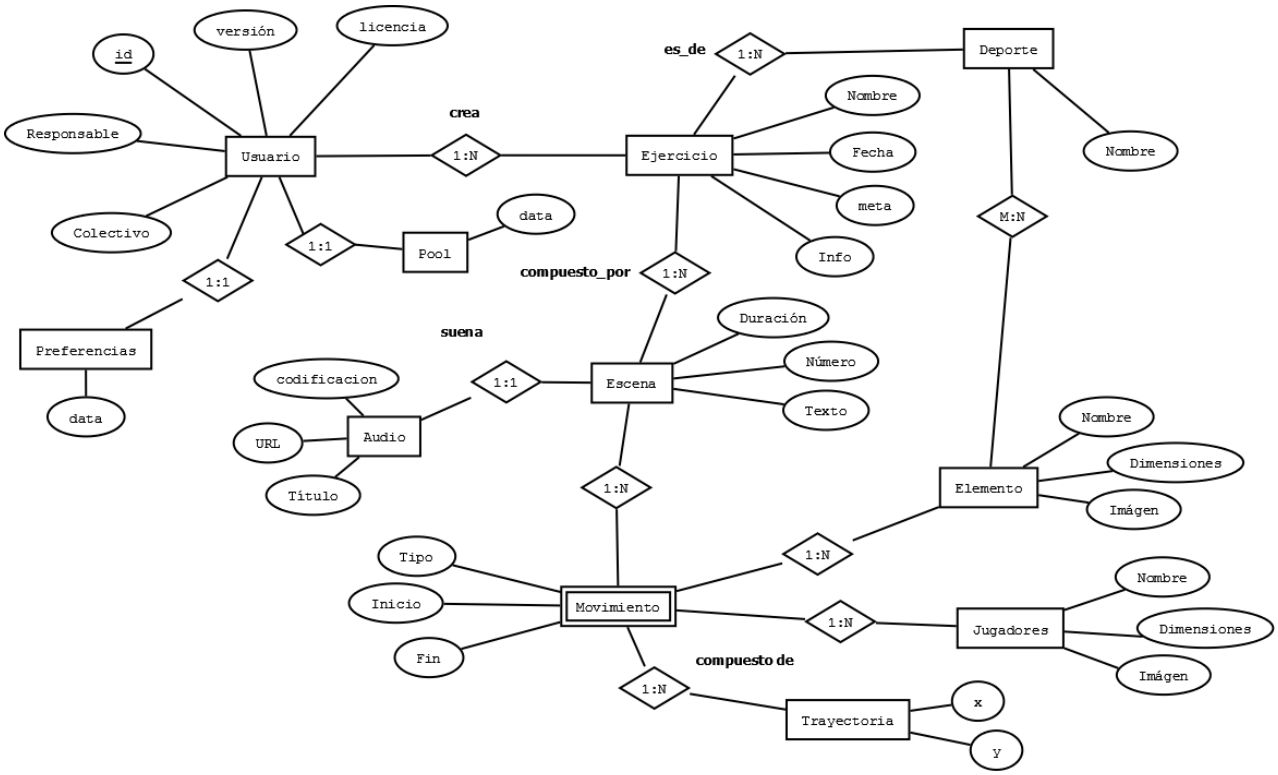


Figura 3.7: Esquema Entidad-Relación

En la figura se puede ver como las entidades y las relaciones son muy similares a las clases del diseño explicado en el punto anterior. Se dispone de una entidad *Ejercicio* que contiene la información general y que está compuesto por *escenas*, y éstas, a su vez tienen varios *movimientos* que están formados a su vez por varios *puntos* que representan una Trayectoria circular, lineal o zig-zag.

También se almacenan en la base de datos las preferencias generales del usuario, que incluyen los colores de las trayectorias o el comportamiento de la aplicación ante distintos eventos entre otros. Los jugadores del equipo en sus correspondientes entidades *Usuario* y *Jugadores* también son guardados en el modelo de datos, siendo las urls a las imágenes que posteriormente mostrará la aplicación la parte más importante.

Por último un deporte está compuesto por varios elementos distintos, que pueden ser comunes a varios deportes. Estos *elementos* junto con los jugadores son lo que se representa en movimientos, es decir las transiciones en las escenas o bien son elementos de deportes (jugadores, balón, etc), o bien son las caras de los jugadores.

El paso a relacional de dicho esquema así como un ejemplo de la utilización del mismo para un ejercicio en concreto se explica en el Anexo B.

3.5. Diseño del interfaz

El diseño de la interfaz gráfica de usuario del proyecto se abordó desde tres puntos de vista distintos: organización de los elementos del producto y usabilidad, la combinación de colores, y la compatibilidad con tablet PCs.

3.5.1. Organización de los elementos

El elemento más importante es el editor de jugadas. Este elemento supone la mayor parte de las entradas del usuario hacia la aplicación y su correcto y adecuado funcionamiento depende de la interfaz escogida. Este modelo de editor debe contener una serie de elementos básicos para poder alcanzar el objetivo:

- **Selector de elementos:** Elemento que permita añadir objetos a la escena.
- **Línea del Tiempo:** Componente que permite al usuario moverse entre las distintas encenas.
- **Editor de la escena:** Panel principal sobre el que se distribuyan los elementos.
- **Selector de Terrenos:** Paleta con los distintos fondos.
- **Edición de trayectorias:** Sistema de definición de trayectorias.

Para el desarrollo del editor se realizaron varios prototipos sobre los que se fueron haciendo cambios. En la Figura 3.8 se muestra la primera versión de este prototipo de editor.

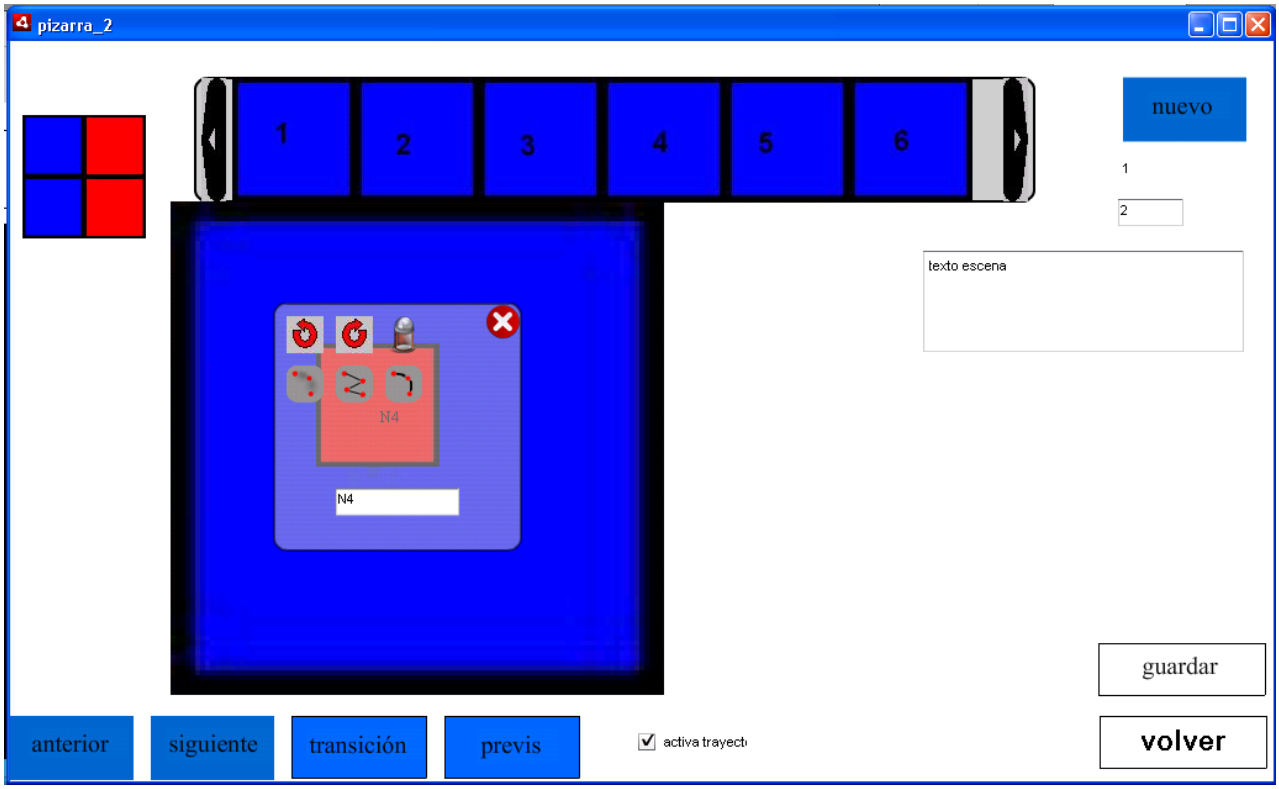


Figura 3.8: Primera versión del interfaz

Esta primera versión contaba con el selector de terrenos en la ventana del editor y los elementos se disponían en un solo panel en la parte izquierda de la aplicación(Cuadrados rojos y azules). Los elementos se arrastrában sobre la zona azul que representa el panel de la escena. Para realizar cualquier acción sobre un elemento hay que hacer doble click y un menú contextual aparece sobre el mismo, donde se puede eliminar, crear la trayectoria o cambiar la etiqueta del mismo.

Para cambiar entre escenas se pulsaba sobre el botón nuevo, y para moverse entre las mismas sobre anterior o siguiente.

En este primer prototipo se detectó que un panel encima del elemento podría impedir el acceso a otros elementos y fue una solución descartada a la hora de gestionar las opciones de los elementos. También hizo falta diseñar una mejor manera de moverse entre escenas, ya que los botones de anterior y siguiente no resultaban intuitivos.

A partir de estas pautas obtenidas con el prototipo se procedió a especificar un segundo prototipo que se puede ver en la Figura 3.9. La especificación completa está disponible en el Anexo B.

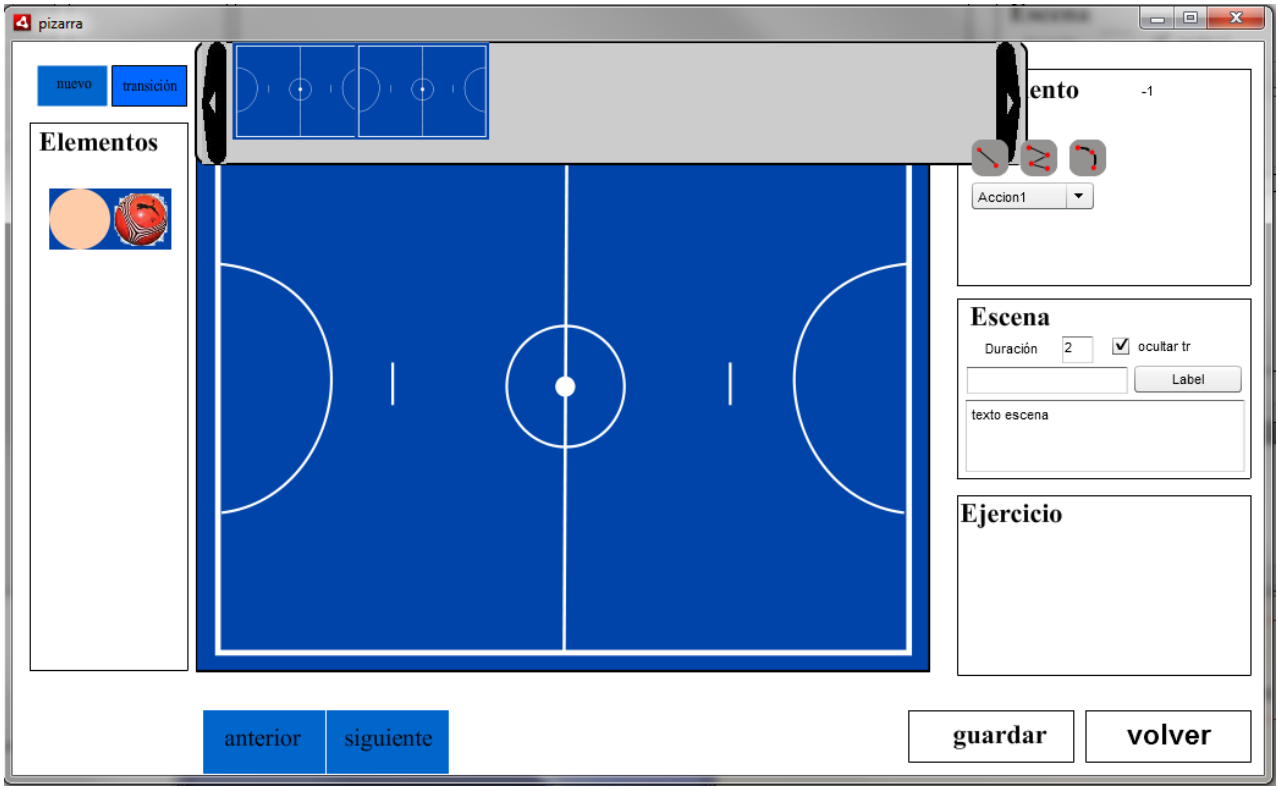


Figura 3.9: Segundo Prototipo

Sustituyendo el panel modal que aparecía sobre el elemento, en esta versión se experimentó con varios paneles a la derecha para configurar los elementos y también se hizo el selector de terrenos desplegable desde la parte de arriba.

Se observó que no es necesario tener el selector de terrenos presente en todo momento en la edición y se puede desplazar a la pantalla previa, donde se introduce la información del ejercicio, y que quede fijo durante la edición gráfica del mismo. La información que aparece en los paneles de la derecha, ocupa un 20 % de la pantalla, y no se utiliza con mucha frecuencia, por lo tanto un sistema alternativo que emplee ese espacio en panel de dibujo resulta mas interesante.

La siguiente versión fue la versión final del editor de ejercicios en la Figura 3.10.

Figura 3.10: Versión final editor

Esta versión se complementa con un línea del tiempo con las miniaturas de las posiciones de las escenas en la parte inferior de la pantalla, y un sistema de pestañas para realizar la edición de los parámetros, encima del panel de edición. La selección de los terrenos se realiza desde la pantalla anterior, así se puede utilizar el espacio de este elemento para el resto de componentes.

3.5.2. Combinación de colores

Para la selección de las paletas de colores utilizadas en la aplicación se tuvo en cuenta en contexto en el que la aplicación se podría utilizar, en este caso, la mayoría del tiempo de uso es dentro de la edición o la reproducción de ejercicios, por lo tanto con el terreno presente. Los terrenos más comunes son, el azul de goma de fútbol sala y balonmano, el verde césped de fútbol y el naranja parquet de baloncesto. La paleta escogida tiene que contrastar con estos colores para que el terreno no se integre por completo en la aplicación y no se vea bien.

Para la elección de la paleta se ha utilizado la aplicación Adobe kuler[37] en la Figura 3.11, que permite la elección de colores partiendo de combinaciones base. Además ofrece toda la información de los colores en las distintas codificaciones para luego utilizarla en el código fuente.

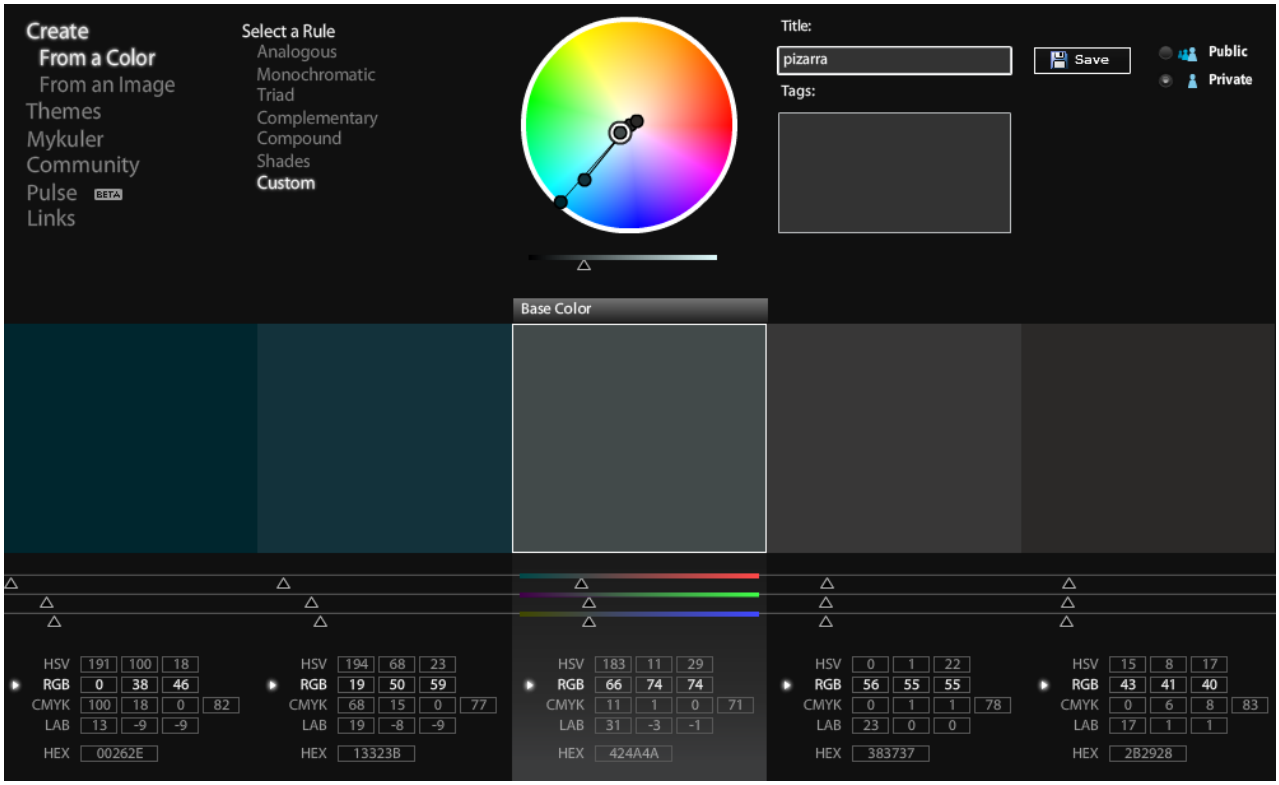


Figura 3.11: Paleta de colores

3.5.3. Compatibilidad con tablet PC

La aplicación hace uso para su funcionamiento normal del botón derecho y de los eventos de MouseOver (pasar el ratón por encima de elementos para cambiar su estado), como por ejemplo en la aparición de las barras de scroll al pasar por encima el ratón. Estas características ofrecen una gestión del espacio mejor ya que aparecen sólo cuando son necesarias si se está usando un ratón para manejar la aplicación, pero no están disponibles para usuarios de tablet PC. Como éstos son usuarios potenciales, se ha habilitado una opción de configuración para establecer cuando se está utilizando la aplicación desde un tablet PC.

Cuando esta opción está activada, las acciones que se activaban con clicks derechos de ratón ahora se activan con dobles clicks, mientras que los elementos que aparecían al hacer MouseOver, se fijan a la interfaz, como las barras de scroll.

3.6. Diseño Técnico

Una vez definidos los componentes funcionales de la aplicación se debe definir a nivel técnico la aarquitectura de la aplicación en la tecnología sobre la que se

implementará la solución.

Como se describió en la Sección 3.3, la implementará basándose en los patrones MVC y Observer. A esto hay que añadir una implementación con arquitectura n-capas [39], que se describe a continuación.

El patrón de diseño en n-capas fue definido por Microsoft para su utilización en proyectos software basados en el framework .net.

La idea de la arquitectura es la separación de la aplicación en distintas partes poco acopladas, y que permiten la abstracción de distintos componentes de manera genérica, lo cual mejora la escalabiliadad del sistema. La pila simplificada sobre la cual se trabajará en este proyecto se puede ver en la Figura 3.12

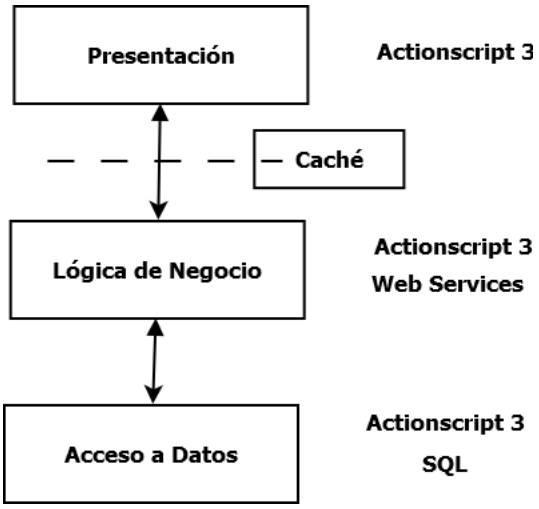


Figura 3.12: Modelo n-capas

La capa de presentación define como se visualiza la aplicación. La capa de caché almacena información que se envía de manera frecuente entre la capa de Presentación y la de lógica de negocio. Ésta última es la que sabe que debe hacer la aplicación, y se encarga de comunicar las interacciones del usuario con la capa de acceso a datos, que es la que da persistencia al sistema.

Para hacer funcionar ambos patrones de diseño a la vez, se ha realizado la siguiente implementación.

Las Vistas engloban todos los componentes gráficos de la aplicación y sólamente la parte gráfica. De esta manera, si se desea cambiar el front-end o realizar un port a otro lenguaje compatible, como pudiera ser en este caso FLEX, sólo se sebería cambiar esta parte del sistema.

Los Controladores de cada pantalla capturan y tratan los eventos de los controles en la vista, y se encargan de llamar a las funciones de la lógica de negocio, la parte

de la aplicación que sabe que acciones realizar. Es esta capa la que accede a los datos a través de los modelos.

Debido a que los datos que se deben tratar en la aplicación no son de un tamaño excesivo, y que esta información se intercambia de manera frecuente entre las distintas partes, se ha optado por un objeto *modelo* común, ya que el impacto en el rendimiento de la aplicación es inperceptible, y simplifica las labores de programación de manera considerable.

Este objeto modelo es el encargado de llamar a las funciones de la capa de acceso a datos. Cada entidad de la base de datos tiene su clase de acceso a datos, por lo tanto cuando se necesita información acerca de una de ellas se debe llamar a la función correspondiente de su Data Access Object. estos objetos suponen una abstracción de alto nivel sobre las acciones comúnes de lectura, actualización, borrado e inserción. Cuando sea necesario se pueden extender estas clases para obtener funcionalidad más específica. La mayor ventaja del sistema es que si se desea migrar a otro sistema gestor de base de datos, o fuente de datos, se debería añadir una implementación concreta, pero el proxy de la capa de acceso a datos seguiría siendo el mismo, por lo tanto no se deberían hacer cambios en la aplicación.

3.6.1. Arquitectura de Componentes

Los elementos gráficos de los que se dispone en el framework por defecto son muy limitados, y el contexto de la aplicación hizo necesaria la implementación de un gran número de controles específicos. Para mantener la coherencia entre los distintos elementos se diseñó una interfaz común para los mismos. La especificación completa se encuentra en el anexo B.

Capítulo 4

Resultados del Proyecto

Una vez finalizada la implementación de la solución final, en este capítulo se explican las funcionalidades de la aplicación resultante. Posteriormente. se dedica un apartado a las distintas evaluaciones realizadas sobre el sistema, primero las de usuarios y después las de la aplicación, ambas tanto intermedias como finales.

4.1. Resultado Final

En esta sección se muestra el resultado final de la aplicación, explicando cada una de las funcionalidades principales que la integran. Estas funcionalidades están descritas con más detalle en la especificación de requisitos A.1 y en los apartados 2.1 Descripción del problema, 2.3 Requisitos del sistema y en el Anexo A.

La aplicación permite crear ejercicios y jugadas a partir del editor que se muestra en la Figura 4.1:

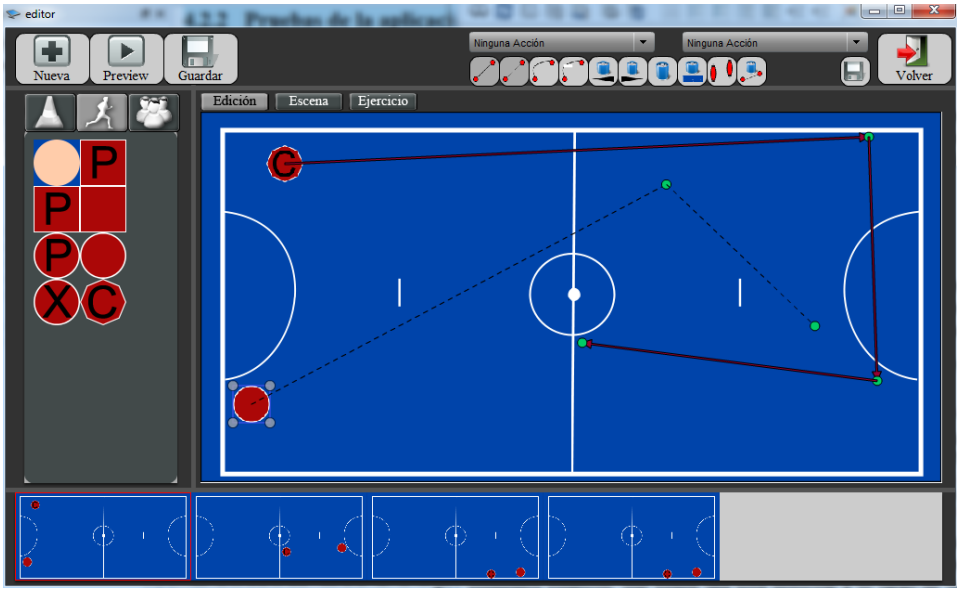


Figura 4.1: Editor de la aplicación

Desde la ventana de edición se puede configurar, a través de sus correspondientes pestañas, todos los elementos de la animación. En la parte izquierda de la pantalla están los selectores de elementos, donde se pueden seleccionar: los elementos estáticos, que tienen una sola posición a lo largo de la animación; los dinámicos, que se pueden mover describiendo trayectorias; y, los personalizados, que se obtienen mediante fotos o la cámara.

La información del ejercicio se puede editar pulsando sobre la pestaña Ejercicio, y también se puede configurar la escena pulsando la pestaña Escena. Esta configuración permite establecer el tiempo de las transiciones, grabar o importar un audio, escribir un texto descriptivo de la escena o determinar si la escena se reproducirá con los movimientos de los elementos concurrentes o serializados.

Para moverse entre las distintas escenas, la aplicación dispone de una línea de tiempo en la parte inferior que permite seleccionar una escena concreta tanto para editar las posiciones de los elementos como sus transiciones.

A través de este mismo editor, se puede hacer una previsualización de la escena que se está editando para verificar si se adecua con lo deseado.

Una vez finalizada la creación y la edición del ejercicio, la aplicación permite guardarlo en la base de datos o descartarlo, pulsando el botón *guardar* o *volver* respectivamente

Para insertar un audio en una escena, la aplicación dispone de una utilidad que graba el sonido del micrófono y sincroniza automáticamente el resultado con la escena que se esté editando en la aplicación. Una imágen de esta aplicación se puede ver en la Figura 4.2.

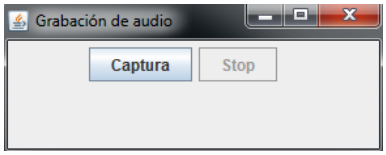


Figura 4.2: Grabador de audio

La aplicación permite acceder a los ejercicios almacenados en la base de datos, ordenarlos de acuerdo a alguno de los campos o realizar un búsqueda mediante un cuadro de texto. Una vez encontrado el ejercicio deseado, se podrá editar, eliminar o visualizar la animación.

La aplicación permite modificar un ejercicio almacenado previamente. Esta edición se realiza a través de una interfaz idéntica a la de la figura 4.1, pero con la configuración del ejercicio seleccionado, resultando en una modificación del mismo o una variante, que se guarda como un ejercicio nuevo.

La animación de un ejercicio se reproduce en el visor que se muestra en la figura 4.2. A través del visor se puede desplazar la animación con la barra inferior, se puede parar, pausar, desplazarse al inicio o al final, así como reproducir hacia delante o hacia atrás. También se puede configurar la velocidad de la animación en relación a la establecida en la creación de la misma, desde -5 hasta 5 veces la velocidad.

El sistema realiza una configuración automática de la relación de los tiempos de la animación con los audios. Así pues si el audio tiene más duración que la animación ésta se reproduce y espera a que termine el audio para comenzar la animación de la siguiente escena.

El visor permite especificar si se desean visualizar las líneas que representan las trayectorias que siguen los elementos marcadas sobre la animación y también permite escalar automáticamente a pantalla completa.

Figura 4.3: Visor de la pizarra

Como se ha explicado a lo largo de esta memoria, la aplicación permite importar y exportar distintos elementos. La carga de nuevos elementos en la aplicación se realiza desde la funcionalidad de importación a la que se puede acceder a través del menú principal, y permite seleccionar ficheros con ejercicios, elementos o una imagen de la base de datos. Estos ficheros provienen de paquetes estándar (actualización de elementos o deportes) o de exportaciones previas u otros usuarios.

La pizarra también dispone de un módulo de exportación de elementos, que se utiliza a través de la pantalla de la figura 4.3, donde se selecciona el ejercicio y el formato en el que se desea exportar de entre los disponibles.

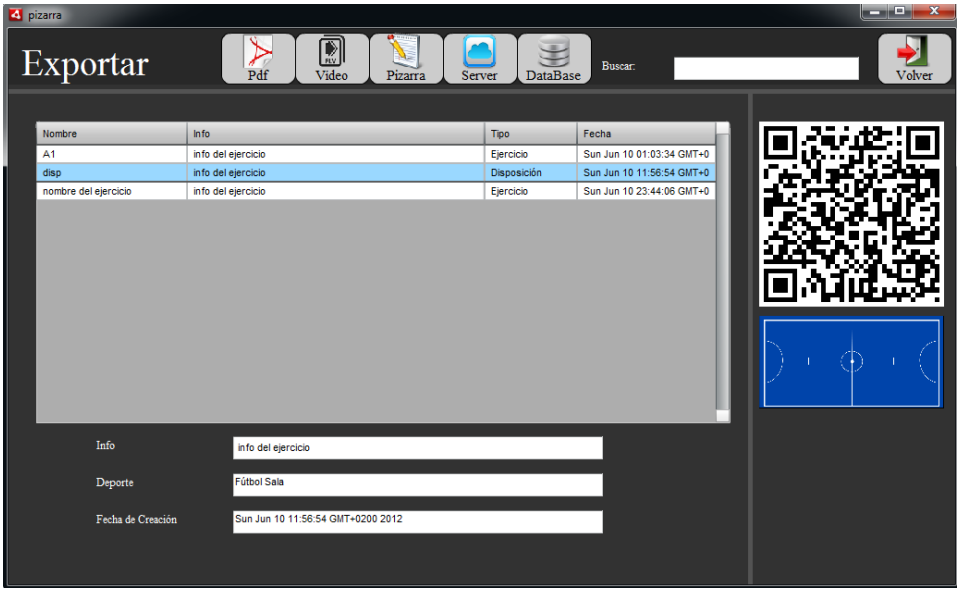


Figura 4.4: Exportación

- **Vídeo:** Se creará un fichero Flash Video flv[10] con el vídeo de la animación del ejercicio seleccionado.

- **PDF:** Se exportará un documento con la descripción del ejercicio paso a paso.

- **piz:** Se creará un fichero de animación que será reproducible a través de la propia pizarra. Este fichero también se podrá importar de nuevo en la aplicación.

- **Servidor:** Se exportará el ejercicio seleccionado a un servidor remoto con el usuario establecido en la aplicación. Estos ejercicios podrán ser listados o reproducidos en la interfaz Web.

La aplicación cuenta con un editor de las preferencias del usuario, figura 4.4, donde se puede seleccionar los colores de las líneas, los nombres por defecto de los ficheros, la url o IP del servidor remoto, la calidad del vídeo generado o configurar el modo Tablet-PC.

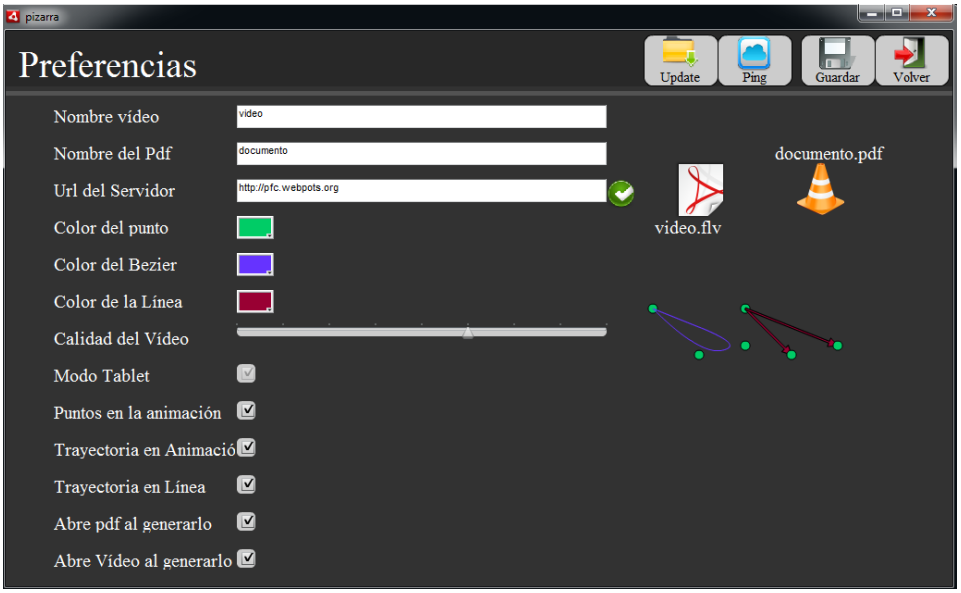


Figura 4.5: Preferencias

La pizarra dispone de una utilidad de libreta de jugadores, figura 4.5, que permite asociar jugadores a la aplicación para posteriormente utilizarlos como elementos en la creación de las jugadas. Éstos se añaden tanto desde ficheros de imagen previamente creados, como con un reconocimiento facial a través de una webcam y utilizando una librería de reconocimiento de objetos [35].

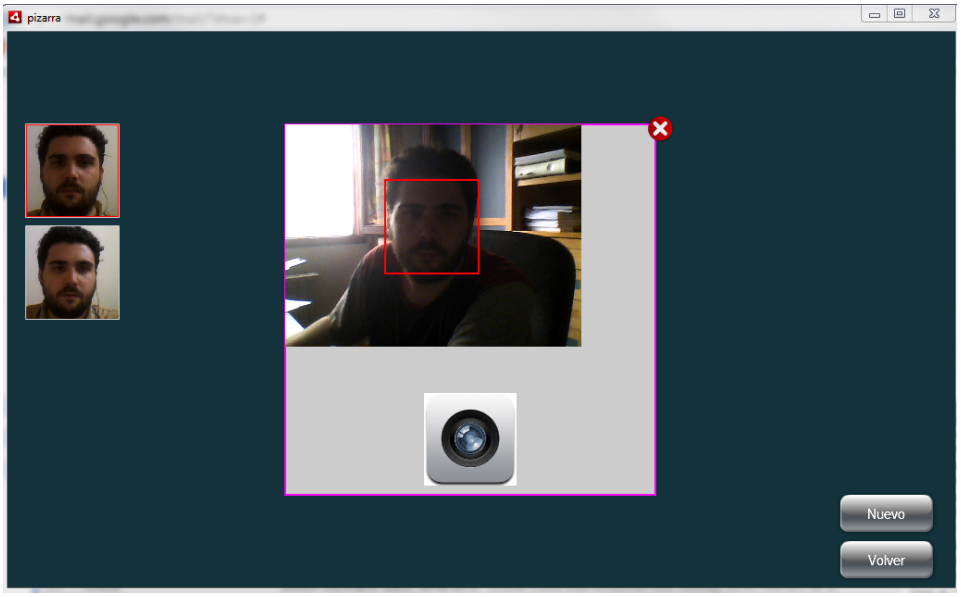


Figura 4.6: Sistema de reconocimiento facial integrado

Por último se describen las funcionalidades de la parte Web. Una vez exportados

los ejercicios, se pueden visualizar todos los ejercicios subidos por el usuario, y posteriormente reproducirlos con el visor HTML de la figura 4.7.

Figura 4.7: Visor Web

4.2. Evaluación de la aplicación

Para evaluar el correcto funcionamiento de la aplicación se han realizado varias pruebas sobre la misma. Con estas pruebas se persiguió un doble objetivo: con los tests sobre usuarios se buscó estudiar de usabilidad y realismo de los vídeos con usuarios potenciales, y posteriormente se realizaron pruebas de integración y del sistema completo que permitieron eliminar los fallos y bugs del sistema.

4.2.1. Evaluación de Usuarios

Las pruebas de usuario fueron realizadas a 5 usuarios distintos con distintos niveles, tanto de conocimientos deportivos, como informáticos, y cuyas descripciones completas se encuentran en el Anexo D.

Las conclusiones principales que se esperaban obtener de estos tests son: 1)Determinar valores para una serie de parámetros de la percepción del vídeo; 2)evaluar cómo realiza el usuario una serie de tareas críticas; 3)comprobar si es capaz de realizar ciertas acciones. Las conclusiones esperadas no son acerca de la usabilidad general de la aplicación para afectar drásticamente a la confección de la misma y a su interfaz gráfica de usuario.

A la hora de visualizar las animaciones que se realizan con la aplicación, existen múltiples configuraciones tanto del tipo de movimiento como de la duración de los mismos.Por lo tanto, para intentar establecer los valores más aproximados a la realidad se realizaron una serie de pruebas con usuarios reales, a los que se les mostró el mismo ejercicio con distintas configuraciones para que valorasen cuál era la configuración más próxima a la realidad, o la que les parecía más adecuada.

Se realizaron también tests de carácter general sobre la aplicación, pidiendo a los usuarios que realizaran una serie de tareas con la aplicación.

Los resultados de los tests de vídeo así como una visión más detallada de los mismos se puede ver en el Anexo D.

4.2.2. Pruebas de la aplicación

Para comprobar el correcto funcionamiento de la aplicación se han realizado tests unitarios de cada uno de los componentes realizados durante la implementación de los mismos.

Una vez implementados todos los componentes, se realizaron tests de integración del sistema completo basándose en Casos de uso de la aplicación. En el diseño de la batería de pruebas se buscó describir los escenarios más comunes de utilización de la pizarra para comprobar que se podían realizar correctamente.

Los tests se pasaron satisfactoriamente o permitieron descubrir errores en la aplicación, que fueron subsanados, para pasar la prueba en una segunda o tercera iteración de la misma.

Los resultados completos de las pruebas realizadas se pueden ver en el Anexo C Pruebas del sistema.

Por último se realizaron pruebas de aceptación por parte del cliente, en este caso los directores del proyecto donde se realizaros los últimos cambios de disposición de elementos y configuración de algunos parámetros y constantes.

Capítulo 5

Gestión del Proyecto

En esta sección se expone la metodología empleada para el desarrollo del proyecto, la planificación, tanto inicial como real, un pequeño estudio de riesgos del proyecto, una estimación de los esfuerzos realizados y, para terminar, las herramientas de gestión que se han utilizado a lo largo del ciclo de vida del proyecto.

5.1. Metodología

La metodología empleada está basada em um el de *Desarrollo en Cascada modificado* [38], figura 5.1, con un modelo puro hasta diseño y revisiones iterativas entre diseño y codificación para poder adaptarse a los cambios y corregir los errores.

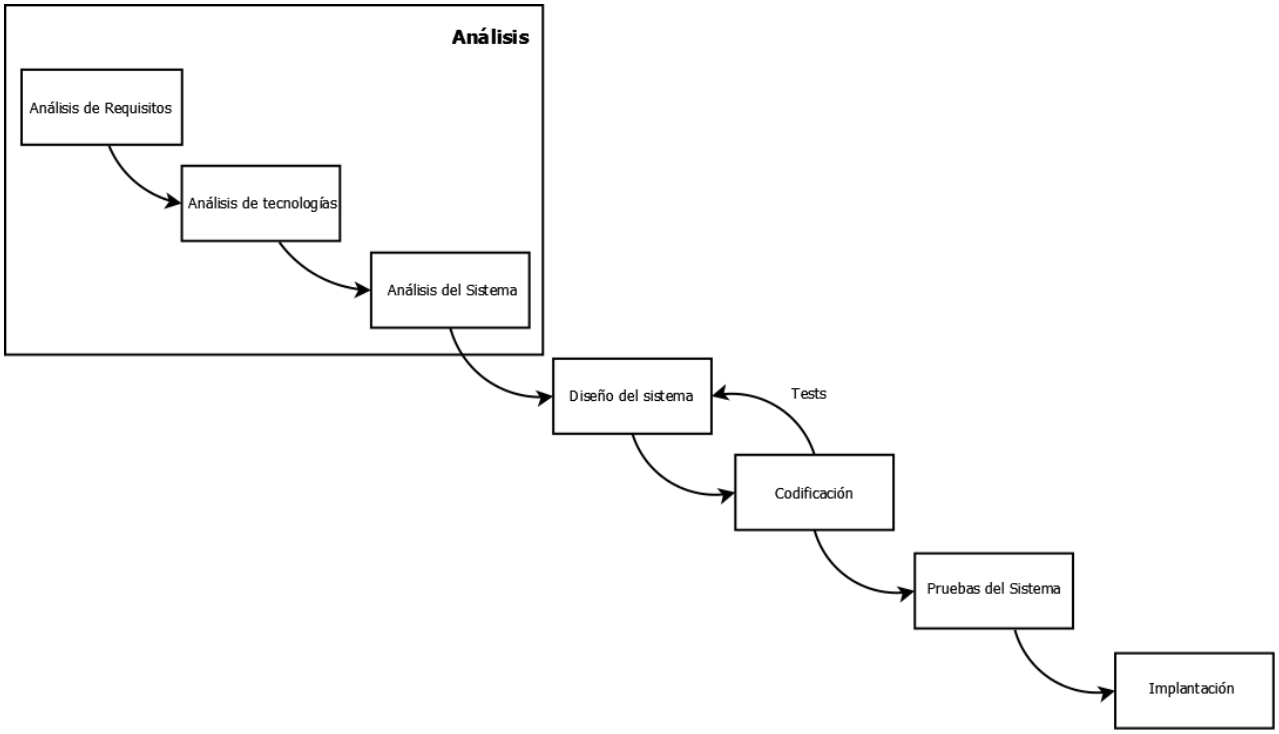


Figura 5.1: Ciclo de vida del Proyecto

El objetivo de la fase de análisis fue obtener un listado de los requisitos que debería tener la pizarra, así como ver con qué tecnologías podrían implementarse. Los resultados de esta fase fueron el documento de especificación de requisitos, el estudio de las posibles tecnologías y la planificación del proyecto. Estos Documentos conforman el Anexo A de la presente memoria.

A partir de estos requisitos, se elaboraron los documentos de diseño de la primera versión de la aplicación, que se sometió a tests, y que produjeron varias iteraciones con cambios tanto en la aplicación como en los documentos de diseño.

Como se describe en el apartado 3.3 Diseño de la interfaz, los tests con los directores del proyecto hicieron de evaluaciones del cliente, donde se validaron las interfaces de usuario para cumplir con los requisitos. Estos tests realizados durante la codificación fueron necesarios también para establecer una serie de parámetros por defecto de la aplicación como se puede ver en el Anexo D Evaluaciones por parte de los usuarios.

Después de la fase de codificación se sometió a la aplicación a una batería de pruebas que comprobaron el correcto funcionamiento de la misma, se puede ver el listado completo con los resultados en el Anexo C.

5.2. Planificación y riesgos

En la sección 5.2.1 se incluyen los diagramas de Gantt que muestran la evolución del proyecto a lo largo del tiempo separado por actividades, y en la sección 5.2.2, la gestión de riesgos que se ha llevado a cabo en el proyecto.

5.2.1. Diagramas de Gantt

El proyecto comenzó el 6 de noviembre de 2009 y la duración estimada del mismo fue de 9 meses. El resultado de la planificación inicial llevada a cabo ha sido el que se puede ver en la figura //TODO: Completar con la fecha de entrega

Figura 5.2: Diagrama de Gantt Inicial

Se observa que la diferencia con el planteamiento inicial es el prolongación de la fase de implementación por los numerosos problemas de carácter general, del compilador y del lenguaje, encontrados en esta etapa.

5.2.2. Riesgos

Al comienzo del proyecto se decidió seguir un plan simple de gestión de riesgos donde se identificaron varios, y se plantearon estrategias de mitigación o eliminación de los mismos[12].

Riesgo 1: El lenguaje de programación no permite realizar alguno de los requisitos básicos

El *riesgo* es claro, a la hora de implementar la aplicación uno de los requisitos funcionales no se puede cumplir utilizando el lenguaje escogido. El *problema* asociado a la manifestación de este riesgo es la necesidad de implementar esa parte mediante otra tecnología, con la consiguiente pérdida de homogeneidad del sistema, ó incluso la necesidad de comenzar la implementación del sistema en un lenguaje de programación que sí permita realizar el requisito en cuestión.

Para *eliminar este riesgo*, se planteó la realización de un estudio de las posibles tecnologías con las que se podría implementar la pizarra, con los requisitos funcionales como base, y realizando pruebas reales de cada una de las características necesarias.

El *resultado del plan de contingencia* para este riesgo fue satisfactorio.

Riesgo 2: El lenguaje de programación es desconocido

El *riesgo* es el desconocimiento de la tecnología Adobe AIR. El *problema* que puede producir este riesgo es que los plazos de entrega estipulados en la planificación no se cumplan.

Para *mitigar* este riesgo se realizará una planificación con unos plazos más amplios para la etapa de implementación.

El resultado del *plan de contingencia* aplicado no ha sido satisfactorio, ya que las correcciones sobre el plan de implementación no fueron lo suficientes, y se tardó más tiempo del estipulado.

5.3. Esfuerzos

En el siguiente apartado se muestran una gráfica con la distribución de las horas empleadas para la realización del proyecto. Las mediciones se han realizado en días estándar de trabajo de 8 horas.

En la figura 5.4 se puede observar la dedicación en horas a cada una de las tareas. En este número de horas se incluye un tiempo razonablemente grande de aprendizaje y estudios previos que ocupa el 14 % del total del proyecto.

En el resto de los apartados, el reparto del tiempo es razonable en proyectos software con un 30 % de análisis y diseño, 13 % de documentación y 46 % de implementación.

El total de tiempo empleado en la realización del proyecto ha sido de 752 horas.

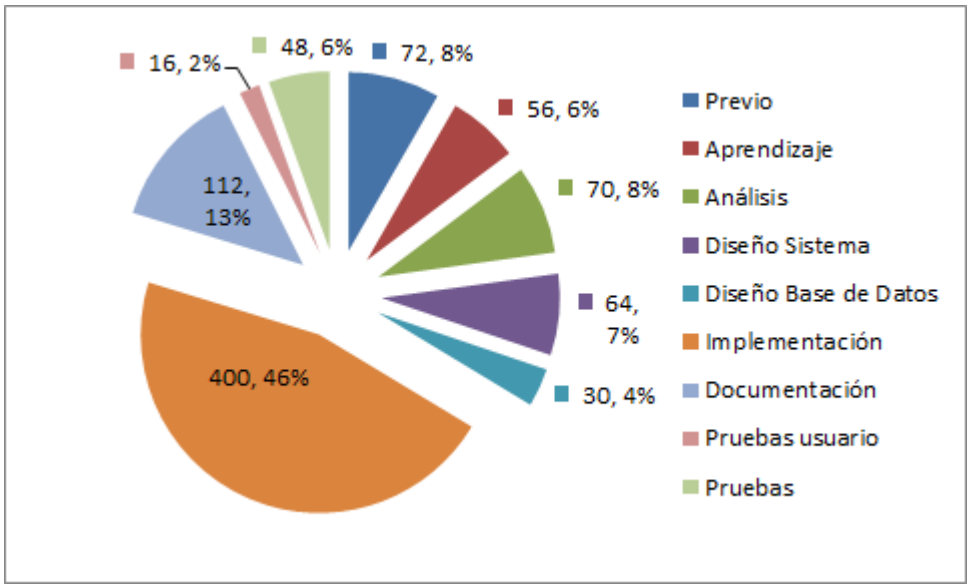


Figura 5.3: Tiempos detallados

5.4. Herramientas de gestión

En esta sección se muestra un listado de las aplicaciones de gestión utilizadas durante el desarrollo del proyecto, acompañadas de una breve descripción de la utilización de la herramienta.

- **Microsoft Word** [21] como editor simple de textos para la documentación de la aplicación.
- **Microsoft Excel** [21] como herramienta de hoja de cálculo para el control de tiempos, y los resultados de las evaluaciones de los usuarios.
- **Adobe Photoshop** [?] para la edición de las imágenes de la aplicación así como la creación de algunas.
- **Microsoft Paint** para la creación de imágenes simples y la captura de pantallas.
- **Dia Diagrammer** [23] para la generación de diagramas de análisis, diseño, bases de datos, y pequeñas imágenes.
- **Inkscape [24]** para la creación de los modelos de los escenarios y otros elementos.
- **Gantt Project [25]** para la gestión temporal del proyecto.
- **Lyx** [?] para la escritura de la memoria del proyecto.

Capítulo 6

Conclusiones

En este último capítulo se presenta una reflexión del trabajo llevado a cabo a lo largo de todo el proyecto, extrayendo varias conclusiones, tanto técnicas como a nivel personal. Finalmente, se exponen algunas vías de trabajo futuro basándose en la aplicación realizada.

6.1. Conclusiones Técnicas

La primera conclusión tras la realización del proyecto es que los objetivos planteados inicialmente, tanto en la propuesta como en la especificación de requisitos, se han cumplido a la finalización del proyecto.

- Se ha desarrollado una aplicación que es capaz de crear ejercicios, jugadas o disposiciones tácticas para los distintos deportes disponibles en la primera versión de la aplicación.
- Los ejercicios creados se pueden exportar en diversos formatos, como fichero nativo de la aplicación, vídeo flv, pdf o a un servidor remoto.
- Los deportes se pueden actualizar a través de la propia aplicación de manera sencilla.

Aparte de estos requisitos se ha abordado la implementación de una serie de características extras, como la utilización de una API de reconocimiento facial para poder utilizar caras obtenidas a través de una webcam como elementos de la aplicación.

El visor se ha portado de la aplicación a Flash para poder utilizarlo en una página web, haciendo la conversión de la capa de datos para extraer la información de una base de datos remota en vez de la base de datos local.

Además de este visor, se ha elaborado otro prototipo utilizando los estándares del W3C [36], utilizando HTML, javascript y CSS.

Los principales problemas encontrados en el proyecto están relacionados con el lenguaje de programación y más en concreto con el compilador del mismo. Se han encontrado algunos bugs en el mismo que han retrasado el trabajo de implementación. La compilación del proyecto completo ha resultado sumamente lenta y la expresividad de los errores mostrados no ayuda a la solución de los mismos.

6.2. Conclusiones Personales

Las aportaciones personales de este proyecto son numerosas y se explican a continuación.

Otro aspecto importante ha sido aprender una tecnología nueva desde cero, en un tiempo razonable, y ser capaz de implementar el proyecto de manera limpia y mantenible en un futuro. Esta nueva tecnología me obligó a adaptarme a un sistema de programación completamente orientado a eventos con cambios importantes en el orden lógico de ejecución de las instrucciones en según que partes del código.

Utilizar una tecnología nueva puede traer problemas. En este caso algunos han sido graves y, al tratarse de un sistema semi-desconocido, el tiempo para resolverlos ha sido considerable.

Trabajar para un "cliente" real es un aspecto fundamental en la formación de un ingeniero en Informática. He mejorado mi capacidad en este aspecto, sabiendo adaptarme a los cambios propuestos y a realizar el sistema de acuerdo a la especificación del cliente.

Elaborar un proyecto de tamaño medio desde cero y con un equipo de una persona es una tarea más compleja de lo que a priori podría parecer. Documentar un proyecto completo día a día es una tarea que no había llevado a cabo antes durante el desarrollo de la carrera, y es una necesidad crucial en la empresa privada.

Dedicarme por completo a un proyecto requiere más concentración de la esperada, ya que estar trabajando varios meses en la misma aplicación, y sólo en ella, requiere liberar la mente en algunas ocasiones.

Este proyecto, además me aporta conocimientos avanzados en una tecnología muy utilizada en la actualidad en casi cualquier página web.

6.3. Trabajo futuro y ampliaciones

A continuación se exponen una serie de características que podrían incorporarse a la aplicación en el futuro:

- A pesar de que la aplicación es bastante completa en cuanto a los distintos formatos que soporta (vídeo, documentos, datos, etc.), cada día el número de

versiones de estos formatos aumenta, haciéndose versiones para los distintos sistemas operativos. En este aspecto se podría trabajar en el futuro para ampliar el número de formatos soportados por la aplicación, sobre todo de vídeo. Formatos interesantes podrían ser avi o mkv. Esta exportación implica el estudio de cómo se codifica byte a byte cada uno de los formatos y convertir la animación.

- También se podría mejorar la exportación de vídeo actual para añadir en los vídeos generados el audio que se reproduce con la animación.

- Para generar los audios explicativos de las escenas se podría utilizar una solución de Text To Speech (T.T.S.) a partir de los textos guardados de las mismas, y así automatizar completamente el proceso. Incluso se podría hacer un sistema más sofisticado que interpretara la jugada y generase automáticamente este audio con la tecnología T.T.S.

- A partir de la meta-información guardada en cada ejercicio, se podría hacer un buscador semántico de jugadas.

- Finalmente, en cuanto se lance la nueva versión de Adobe AIR, que permita ejecución de aplicaciones o comandos, y grabación de audio, se podría integrar el sistema completo, y sustituir la aplicación Java grabadora de audio por una versión dentro de la pizarra, o llamar a esta aplicación java directamente. La aplicación se ha diseñado pensando en que esto va a ocurrir en un futuro cercano y los cambio para esta mejor son mínimos.

Capítulo 7

Modelos

7.1. Motivación

Esto es un ejemplo de Sección de un Capítulo.

Y esto un ejemplo de insercíon de una Figura 7.1 y de una Tabla 7.1.

Figura 7.1: Pie de la figura

Tiempo dedicado por tarea

Tarea	Tiempo
Estudio Previo	x
Análisis y Diseño	x

Tabla 7.1: Pie de la tabla

Esto es un ejemplo de enlace a bibliografía:

- *LaTeX* [1] para la elaboración de la documentación. Los componentes utilizados fueron:
  - *MiKTeX 2.8* [2] como distribución *TeX/LaTeX* para Microsoft Windows XP.
  - *LyX 1.6.5* [3] como entorno gráfico para escribir documentos en *LaTeX*.

Anexos

Apéndice A

Análisis

A.1. Requisitos completos de la aplicación

Requisitos completos de la aplicación:

Código	Descripción
	Requisitos generales
RF-1	La aplicación permitirá crear la descripción de un ejercicio, una jugada o una disposición
RF-2	La aplicación permitirá añadir o eliminar elementos de representación y el movimiento de
RF-3	La aplicación permitirá asignar un nombre y un conjunto de metadatos descriptivos a cad
RF-4	La aplicación permitirá guardar y, posteriormente, recuperar un ejercicio, jugada o dispos
RF-5	La aplicación permitirá editar las escenas (añadiendo, eliminando o modificando escenas
RF-6	La aplicación permitirá añadir un audio explicativo de un ejercicio, jugada o disposición
RF-7	La aplicación permitirá reproducir el video y audio de un ejercicio, jugada o disposición t
RF-8	La aplicación permitirá establecer la velocidad de reproducción de un video y reproducir
RF-9	La aplicación permitirá realizar búsquedas de ejercicios, jugadas o disposiciones tácticas
	Representación de los elementos
RF-10	La aplicación permitirá configurar los elementos de representación en función de un depo
RF-11	La aplicación permitirá representar diferentes campos de juego (futbol sala, futbol, balon
RF-12	La aplicación permitirá representar jugadores de distintos equipos sobre un modelo de ca
RF-13	La aplicación permitirá representar balones, conos, vallas, cuerdas y otros elementos habi
RF-14	La aplicación permitirá representar el movimiento (línea recta, curva, zigzag, etc.) de un
RF-15	La aplicación permitirá representar el movimiento (golpeo, pase, etc.) de un balón en el c
RF-16	La aplicación permitirá coordinar el movimiento de varios jugadores y/o el balón en el co
	Otros requisitos a nivel de aplicación
RF-17	El acceso a la aplicación será seguro.
RF-18	La aplicación integrará un mecanismo de control de licencias temporal.
RF-19	La aplicación podrá comunicarse utilizando un protocolo propietario con un servidor de a
RF-20	La aplicación será capaz de realizar copias de seguridad (creación de copia y recuperación

Tabla A.1: Requisitos funcionales

Código	Descripción
RNF-1	La aplicación será local y ejecutable sobre un ordenador de tipo Tablet-PC o Netbook (o
RNF-2	Para la creación o edición de las escenas, la aplicación permitirá la utilización d
RNF-3	La aplicación será desarrollada en base a la idea de produc
RNF-4	Las tecnologías de desarrollo fundamental de la aplica

Tabla A.2: Requisitos no funcionales

A.2. Estudio de Tecnologías Pizarra Digital

Para la elaboración de proyecto se ha realizado un estudio previo de las posibles tecnologías para implementar el mismo. Este estudio se ha basado en tres de las que se consideraron más adecuadas en el momento inicial, Adobe Flex, Adobe Flash, Java.

Los parámetros de comparación son muy diversos y se basan en aspectos relativos a la eficiencia, el resultado final y la mantenibilidad del sistema, entre otros.

Como resultado de este estudio se realizó una presentación a los directores del proyecto con las conclusiones obtenidas del estudio, y se tomó una decisión basada tanto en valores subjetivos, como en un método objetivo y formal de decisión aplicado sobre las distintas tecnologías.

Primero se van a contextualizar cada una de manera resumida, y en sucesivas secciones se describirá cada una de ellas para cada uno de los parámetros estudiados.

A.2.1. Tecnologías estudiadas

Las tecnologías estudiadas han sido Flex, Flash y Java, aunque se describe brevemente Actionscript como lenguaje utilizado tanto por Flash como por Flex y Adobe AIR como el entorno de ejecución de Flash y Flex en escritorio.

Adobe Flex

Adobe Flex es un kit de desarrollo lanzado por Macromedia para el desarrollo de Rich Internet Applications multiplataforma. Las aplicaciones se pueden escribir usando Adobe Flash Builder o utilizando el compilador de Flex gratuito.

Flex desarrolla las Interfaces gráficas de usuario usando un lenguaje XML llamado MXML. Flex tiene varios componentes y características que aportan funcionalidades tales como Servicios Web, objetos remotos, arrastrar y soltar, columnas ordenables, gráficas, efectos de animación y otras interacciones simples. El cliente sólo carga la aplicación una vez, mejorando así el flujo de datos frente a aplicaciones basadas en HTML (eg.PHP, ASP, JSP, CFMX), las cuales requieren de ejecutar plantillas en el servidor para cada acción. El lenguaje y la estructura de archivos de Flex buscan el desacoplamiento de la lógica y el diseño, utilizando MXML para el marcado y actionscript para la programación.

Proceso de desarrollo de una aplicación Flex:

- Definir un interfaz de aplicación usando un conjunto de componentes predefinidos (formularios, botones,...)
- Ordenar estos componentes en el diseño del interfaz de usuario
- Usar estilos y temas para definir el diseño visual
- Añadir comportamiento dinámico (una parte de la aplicación interactuando con otra, por ejemplo)
- Definir y conectar a servicios de datos según sea necesario (servicios http)

- Compilar el código fuente en un archivo SWF que funcione en el reproductor Flash.

Al utilizarse conjuntamente con la infraestructura AIR, permite el acceso a bases de datos y ficheros locales, así como la generación de un instalador multiplataforma.

Adobe Flash

Adobe Flash (anteriormente llamado Macromedia Flash) es una aplicación en forma de estudio de animación que trabaja sobre "fotogramas", destinado a la producción y entrega de contenido interactivo para las diferentes audiencias multiplataforma.

Los archivos de Flash, que tienen generalmente la extensión de archivo SWF, pueden aparecer en una página web para ser vista en un navegador, o pueden ser reproducidos independientemente por un reproductor Flash. Los archivos de Flash aparecen muy a menudo como animaciones en páginas Web y sitios Web multimedia, y más recientemente Rich Internet Applications mediante Adobe AIR.

AIR es un entorno de ejecución versátil, ya que permite que el código Flash, HTML o JavaScript existente se reutilice para construir programas tradicionales de escritorio. Adobe lo define como un entorno de ejecución sin navegador para que Rich Internet Applications se puedan desplegar en el escritorio, en lugar de un framework de aplicaciones. Las diferencias entre cada paradigma de despliegue proporciona a ambas ventajas y desventajas. Por ejemplo, una Rich Internet Applications desplegada en un navegador no requiere instalación, mientras que una desplegada con AIR requiere que sea empaquetada, firmada digitalmente, e instalada en el sistema de archivos local de los usuarios.

Sin embargo, esto último ofrece almacenamiento local ilimitado y acceso al sistema de archivos, mientras que las aplicaciones desplegadas en el navegador están limitadas por las restricciones del mismo, y donde los datos, por lo general, se eliminan periódicamente. Sin embargo, en la mayoría de los casos, las Rich Internet Applications almacenan los datos de los usuarios en sus propios servidores, pero la capacidad para consumir y trabajar con datos en el sistema de archivo local de un usuario permite una mayor flexibilidad cuando una aplicación está trabajando sin conexión.

AIR actualmente tiene cuatro maneras de trabajar con datos:

- Servidor de base de datos a través de servicios web
- Archivo local de XML
- Base local de datos de SQLite enviadas con AIR
- Almacenamiento de cifrado local incluido con AIR

Lo cual dota a los sistemas creados sobre AIR de una gran flexibilidad en cuanto a almacenamiento local se refiere.

Java

Java es un lenguaje de programación desarrollado en Sun Microsystems y lanzado en 1995. Es un lenguaje de programación de carácter general, orientado a objetos y multiplataforma, tiene una sintaxis similar a la de C++. Los programas Java se ejecutan sobre una máquina virtual a través del bytecode, un código intermedio más abstracto que el código máquina. En 2007 Sun liberó el código fuente de la plataforma Java, convirtiéndolo en Software Libre casi en su totalidad. La elección de Java frente a otras alternativas se debe, además de lo ya nombrado, a las siguientes características:

- Portabilidad. Puesto que los programas Java se ejecutan sobre la máquina virtual de Sun, las aplicaciones funcionan en cualquier sistema operativo para el que esté implementada la plataforma Java. Esto hace que el código sea independiente de la plataforma y evita crear código específico para cada sistema operativo que se quiera soportar. Por este mismo motivo, las aplicaciones Java también son independientes de la plataforma hardware.
- La gestión de memoria dinámica en Java se realiza de manera transparente al programador. Cuando se instancia un nuevo objeto, automáticamente se reserva el espacio en memoria necesario para este. En cuanto a la liberación, Java cuenta con un sistema de recolección de basura, que se encarga de detectar los objetos ya inaccesibles por el programa, y liberar el espacio que ocupan. Esto evita que el programador tenga que preocuparse de la gestión de memoria, evitando sus problemas derivados.
- Java cuenta con varios entornos de desarrollo, que integran múltiples características para facilitar ciertas tareas en el proceso de programación. En la siguiente sección se explican las ventajas del uso de este tipo de software.
- Bibliotecas externas. El reciente auge de Java ha concentrado en gran medida desarrollo para esta plataforma, lo que hace que existan numerosas bibliotecas para extender la funcionalidad de las aplicaciones que se desarrollen. Además, gracias las características de la máquina virtual comentadas anteriormente, la integración de bibliotecas externas a una aplicación en desarrollo es sencilla.

Actionscript 3

Actionscript es un lenguaje de script propiedad de Adobe, y basado en ECMAScript, utilizado comúnmente para el desarrollo de aplicaciones en las plataformas de adobe como Adobe Flash, Adobe AIR, o Adobe Flex.

Inicialmente fue diseñado para controlar animaciones vectoriales en 2D, pero las posteriores evoluciones del mismo lo convirtienron en un lenguaje para la creación de Rich Internet Appliactions.

La versión 3 de Actionscript es un lenguaje de programación orientado a objetos, permitiendo mayor control del código y reusabilidad cuando se están escribiendo aplicaciones complejas. Permite la programación en paquetes y en ficheros de código fuente separados.

Permite integrar también porciones de código junto con la interfaz (acciones en .fla, o fragmentos en mxml). La programación en este lenguaje es muy dependiente de eventos, y para cualquier instrucción compleja ha de definirse un función a ejecutar cuando la acción se ha completado.

Actualmente la ampliación de este lenguaje con librerías es aún escasa, y la variación entre versiones es muy grande, lo cual dificulta portar la aplicación a versiones superiores del lenguaje.

A.2.2. Requisitos

Los siguientes parámetros se consideran indispensables para la aplicación por lo tanto se deben poder realizarse con cada una de las tecnologías, o la tecnología se considerará inválida para la realización del proyecto.

Estos requisitos se extraen directamente de algunos de los requisitos funcionales de la aplicación.

Drag & Drop

Para la edición de los ejercicios es necesario que se puedan arrastrar y soltar elementos hacia la escena o en la misma. Esta función tiene que ser, fácil de implementar, y la sensación tiene que ser de fluidez.

[RF2] La aplicación permitirá añadir o eliminar elementos de representación y el movimiento de éstos a una escena (jugadores, balones, material de entrenamiento, etc.)

[RF5] La aplicación permitirá editar las escenas (añadiendo, eliminando o modificando escenas individuales) de un ejercicio, jugada o disposición táctica previamente guardada, creando una variante de la misma o una nueva versión.

- **Java:** Se puede conseguir el efecto de *Drag & Drop* utilizando la clase nativa graphics 2D, por lo tanto el requisito se cumpliría.
- **Flash:** El lenguaje utilizado dispone de soporte nativo para *Drag & Drop*, cumple el requisito y se desarrolló una demo para comprobarlo.

- **Flex:** Utilizando Actionscript se puede conseguir el efecto *Drag & Drop*, se cumple el requisito.

Almacenamiento local de datos

Tres requisitos funcionales hacen necesario almacenar datos dentro del disco duro del usuario.

[RF3] La aplicación permitirá asignar un nombre y un conjunto de metadatos descriptivos a cada ejercicio, jugada o disposición táctica.

[RF4] La aplicación permitirá guardar y, posteriormente, recuperar un ejercicio, jugada o disposición táctica en formato edición (la secuencia de escenas originales y eventos) o en formato de vídeo

[RF20] La aplicación será capaz de realizar copias de seguridad (creación de copia y recuperación de copia) de los ejercicios y su metadatos.

- **Java:** Permite leer y escribir en casi todas las Bases de Datos existentes en el mercado así como en ficheros locales
- **Flash:** Utilizando la plataforma adobe AIR es posible la lectura y escritura directa tanto de Base de Datos, como ficheros Locales. El Sistema Gestor de Bases de Datos utilizado por AIR es SQLite, un Sistema Relacional. Cumple el requisito y para comprobarlo se realiza una demo.
- **Flex:** Utilizando la plataforma adobe AIR es posible la lectura y escritura directa tanto de Base de Datos, como ficheros Locales. El Sistema Gestor de Bases de Datos utilizado por AIR es SQLite, un Sistema Relacional. Cumple el requisito y para comprobarlo se realiza una demo.

Reproducción de audio

La aplicación debe poder reproducir audio, al menos para las explicaciones de las escenas, de acuerdo con el siguiente requisito funcional.

[RF6] La aplicación permitirá añadir un audio explicativo de un ejercicio, jugada o disposición táctica (el audio será a nivel de escena).

- **Java:** Permite la reproducción y la grabación de audio en casi cualquier formato y se ha desarrollado una aplicación que graba audio a través del micrófono por defecto.
- **Flash:** Permite la reproducción de audio en formato mp3 a través de código Actionscript.
- **Flex:** Permite la reproducción de audio en formato mp3 a través de código Actionscript.

Animación

En una aplicación que se basa en la animación, es fundamental, que éstas sean fáciles de configurar y sean lo suficientemente continuas con distintos tipos de imágenes y tiempos.

[RF7] La aplicación permitirá reproducir el vídeo y audio de un ejercicio, jugada o disposición táctica.

[RF8] La aplicación permitirá establecer la velocidad de reproducción de un vídeo y reproducir hacia delante, parar o hacia atrás el ejercicio, jugada o disposición táctica.

- **Java:** Mediante temporizadores y el borrado de la pantalla se puede conseguir un efecto similar a una animación. Se desarrollo un prototipo de sistema de animaciones, en el que se probó con distintos tipos de imagen y tiempos, y se concluyó que las animaciones no resultaban lo suficientemente fluidas para las necesidades de la aplicación.

- **Flash:** Dispone de un motor de animación nativo, accesible y configurable desde código (Tweens) o mediante el editor.

- **Flex:** Similar sistema al de Flash, pero sólo accesible desde código Actionscript.

A.2.3. Otros parámetros

Aquí se describen otros aspectos de las tecnologías deseables, pero no indispensables para la consecución del proyecto, ni asociados a ningún requisito funcional.

Ajuste del lenguaje y su representación con metodologías de diseño

Es importante que el código fuente, escrito en el lenguaje elegido, sea fácilmente transcribible desde los diagramas de la metodología de diseño escogida.

- **Java:** Ajuste ideal tanto en representación UML como en la generación automática de código.

- **Flash:** No es mala, ya que actionscript es un lenguaje Orientado a Objetos.

- **Flex:** No es mala, ya que actionscript es un lenguaje Orientado a Objetos, en cuanto a la interfaz es similar a HTML.

Estructuración y modularidad del lenguaje

Para que un proyecto sea completamente satisfactorio, éste debe poderse mantenerse fácilmente. Éste aspecto está íntimamente relacionado con la estructuración y la modularidad del código en el lenguaje escogido.

- **Java:** La programación se realiza mediante ficheros de texto con el código fuente.
- **Flash:** Editor Gráfico para la interfaz y ficheros con código para la programación.
- **Flex:** Ficheros con código, diferenciando estructuración y presentación de funcionalidad.

Facilidad de conversión del sistema a formato Web

Un factor importante para el futuro del proyecto es la facilidad con la que se podría portar el visor de ejercicios para ser utilizado en una página web.

- **Java:** Desde Java se pueden generar applets realizando algunos cambios en el interfaz de usuario, y cambios pequeños en el acceso a datos.
- **Flash:** Se puede generar ficheros swf que se pueden utilizar en páginas web mediante la etiqueta object, para adaptar el visor de la aplicación a la versión web, se tendría que adaptar la capa de acceso a datos.
- **Flex:** De manera similar a Flash, se podría generar un swf y utilizarlo en un proyecto HTML cambiando la capa de acceso a datos.

Velocidad

Es fundamental que la aplicación se ejecute bien y lo más rápido posible.

- **Java:** La ejecución de java es lenta y poco fluida en las animaciones, mientras que es rápida en el resto, además permite la programación en varios hilos para aumentar la velocidad de ejecución en algunas partes de la aplicación.
- **Flash:** Es rápido en las animaciones, algo más lento en el resto del código.
- **Flex:** Es rápido en las animaciones, algo más lento en el resto del código..

Ampliación del sistema

El proyecto está creado desde su inicio con su escalabilidad y las futuras ampliaciones en mente, por lo tanto es recomendable que el código generado sea mantenible fácilmente y por personas diferentes a las que lo desarrollaron en un primer momento.

- **Java:** Sencilla con un proyecto bien documentado.
- **Flash:** Necesidad de revisión de ficheros gráficos con elementos de la aplicación.
- **Flex:** Sencilla con un proyecto bien documentado.

Integración con otros componentes

De cara a futuras modificaciones del sistema, es necesario conocer las posibles ampliaciones que se pueden realizar con el lenguaje escogido, así como la comunicación con otros sistemas y lenguajes.

- **Java:** Dispone de una amplia comunidad que desarrolla librerías para casi cualquier necesidad.
- **Flash:** Sistema cerrado de ejecución, algunas librerías escritas en Actionscript 3.
- **Flex:** Sistema cerrado de ejecución, algunas librerías escritas en Actionscript 3.

Generación de documentos

Una funcionalidad muy interesante es la generación de documentos pdf, para ello se debe estudiar la existencia de la infraestructura necesaria.

- **Java:** Java dispone de una API nativa para la generación de pdf.
- **Flash:** Dispone de la librería Alive pdf para la generación de documentos.
- **Flex:** Dispone de la librería Alive pdf para la generación de documentos.

Coste de codificación

Según el lenguaje es muy diferente el coste de codificación del proyecto, separando la interfaz y la programación de las funciones en general.

- **Java:** En general grande, la interfaz se implementa completamente a través de programación, la funcionalidad también pero se hace de manera más potente
- **Flash:** Se hace una generación gráfica de la interfaz a través de un sistema drag&drop, la programación de la funcionalidad es más simple en algunos fragmentos orientado a script.
- **Flex:** La interfaz se genera por marcado, de forma sencilla, la funcionalidad es simple.

Coste de aprendizaje

Para la estimación del tiempo necesario para el desarrollo, es importante conocer cual es el tiempo necesario para conocer tanto el lenguaje como las librerías necesarias para la implementación del proyecto.

- **Java:** El lenguaje es conocido y las librerías están bien documentadas luego el coste es pequeño
- **Flash:** La tecnología es completamente nueva, pero el lenguaje es similar a otros conocidos, por lo tanto el coste sería medio
- **Flex:** La tecnología es también completamente desconocida, y el sistema es algo más complicado que flash, luego el coste sería medio-alto

Compatibilidad

El sistema estará desarrollado con la concepción de producto, por lo tanto la compatibilidad con distintos sistemas es importante, en este caso se estudiarán los sistemas operativos principales Windows, MAC OS y GNU/Linux.

- **Java:** Java es completamente multiplataforma
- **Flash:** Teóricamente es multiplataforma, pero en algunas de las distribuciones de Linux probadas no funciona bien.
- **Flex:** Teóricamente es multiplataforma, pero en algunas de las distribuciones de Linux probadas no funciona bien.

A.2.4. Conclusiones Subjetivas

A partir del estudio de las características anteriores así como el resultado de los distintos programas de prueba realizados, se han obtenido una serie de conclusiones subjetivas que conducirán a la decisión final adoptada.

- Flash es más sencillo de programar, y el resultado puede ser más espectacular visualmente.
- A priori, Flash, puede hacer toda la funcionalidad requerida por el problema.
- Las ampliaciones futuras están limitadas por el lenguaje Actionscript 3.
- Java es más complejo de programar, pero se adapta mejor a las metodologías de diseño.
- Con trabajo se puede llegar a casi el mismo resultado con Java que con Flash.
- Las animaciones con Java no son excesivamente fluidas, pero las ampliaciones son casi ilimitadas.
- Flex es en lo que respecta a programación igual a Flash.
- Flex no aporta ventajas sobre el mismo si los dos se ejecutan sobre ámbito AIR.

Debido a estos aspectos la tecnología con la que se implementará el proyecto será Adobe Flash, con el lenguaje de programación Actionscript 3 sobre el runtime Adobe AIR.

A.2.5. Conclusiones Objetivas

Para hacer un análisis objetivo de las distintas características de las tecnologías y escoger la mejor, se utiliza un modelo global multicriterio para la toma de decisiones.

En este caso se valoró cada una de las características y se le asignó un peso a cada una de ellas, para posteriormente aplicar las siguientes fórmulas y obtener la valoración de cada lenguaje.

$$I = FC_j * [\alpha * FO_j + (1 - \alpha) * FS_j]$$
$$FC_j = \prod_i^m FC_{ij}$$
$$FO_j = \frac{Medición_j}{Max(Medición_j)}$$
$$FS_j = \frac{\sum_{i=1}^{m} P_{ij}}{MP}$$

Tabla A.3: Fórmulas Modelo Global multicriterio

Los resultados de las ponderaciones son los de la Figura A.1

Factores Críticos

	Drag&Drop	Almacenamiento	Audio	Animación
Java	1	1	1	1
Flash	1	1	1	1
Flex	1	1	1	1

FC=	1
FC=	1
FC=	1

Factores subjetivos

	Metodologías	Estructuración	Web	Velocidad	Ampliación	Integración	Documentos	Codificación	Aprendizaje	Total
Java	1	1	3	3	1	1	3	3	1	17
Flash	2	2	1	1	2	2	1	1	2	14
Flex	3	3	2	2	3	3	2	2	3	23

Figura A.1: Resultados parciales método

Una vez ponderados los distintos factores del estudio hace falta calcular el índice final para cada lenguaje de programación. Para calcular dicho índice se utiliza la fórmula correspondiente de la Tabla A.3 con una alfa estándar 0.5.

Índice

	IL
Java	0,81481481
Flash	0,75925926
Flex	0,92592593

Figura A.2: Índices resultantes

El menor índice resultante es el lenguaje de programación escogido, en este caso Flash + AIR.

Apéndice B

Diseño del Sistema

Partiendo del análisis expuesto en el capítulo anterior, se describe primero el entorno de ejecución de la aplicación, las interacciones entre los distintos usuarios así como los ficheros y elementos utilizados. Posteriormente se expone la arquitectura del sistema, describiendo los tres niveles en los que se ha basado este diseño, para después hacer un análisis más profundo de cada uno de ellos. El siguiente apartado aborda el diseño de la base de datos, y las relaciones existentes entre las distintas entidades, por último se expone el diseño del interfaz de usuario.

B.1. Entorno de la aplicación

La aplicación podrá instalarse y ejecutarse en cualquier computador con los sistemas operativos Windows, MAC OS X o GNU/Linux, que disponga del Runtime Adobe AIR [17]. El diagrama de despliegue del sistema se expone en la Figura B.1.

En el sistema existen dos tipos de usuarios bien definidos en relación con las actividades que pueden realizar: por un lado, los **entrenadores**, que son los encargados de la gestión de los ejercicios, y los **jugadores**, que son los usuarios fineles la información generada con la pizarra.

El entrenador puede crear y modificar los ejercicios a través de la pizarra y posteriormente exportar la parte que desee a un fichero externo, que será el que utilicen los jugadores. El entrenador es también el encargado de la actualización del sistema (elementos, ejercicios o deportes), a partir de los ficheros correspondientes, o de información que él mismo hubiera exportado utilizando la pizarra.

El jugador utiliza la información generada con la aplicación, ya sea viendo la animación o el vídeo generado con la pizarra, o estudiando la jugada a través de un documento pdf obtenido del repositorio de la aplicación.

En la Figura B.1 se puede apreciar un diagrama de despliegue de los distintos sistemas. La aplicación puede ser utilizada tanto por el entrenador, para generar

información o actualizar el sistema, como por los usuarios para consumir esta información. La aplicación es capaz de sonectarse con un sistema remoto para consumir información y enviar los ejercicios al sistema remoto, que a su vez servirá esta información a través del navegador.

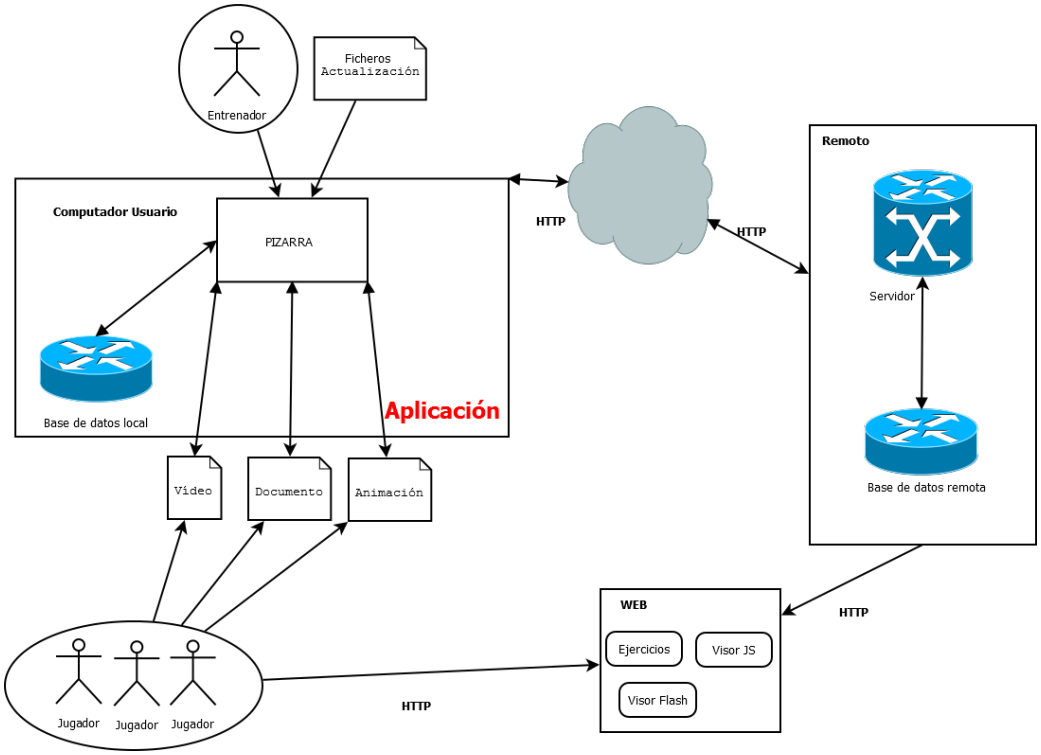


Figura B.1: Diagrama de Despliegue

Para exportar la información a un servidor remoto, y así poder ser visualizada a través de intenet, se utilizan llamadas *Http* con un formato específico. Cuando el servidor recibe esas llamadas, escribe la información correspondiente a los ejercicios en la base de datos remota, y responde con el resultado de la transacción. A partir de esta información remota, se puede acceder vía Web a los ejercicios exportados por un usuario y visualizarlos en el visor html.

La aplicación utiliza para su funcionamiento una base de datos local en SQLite[28], de la que se extrae la información almacenada de los ejercicios, la información de los deportes, los elementos, las preferencias, etc. El funcionamiento y diseño de dicha base de datos se explica en el apartado 3.4.

B.2. Wireframes

En esta sección se muestran los esquemas de alto nivel sobre los que se basa el diseño software del sistema.

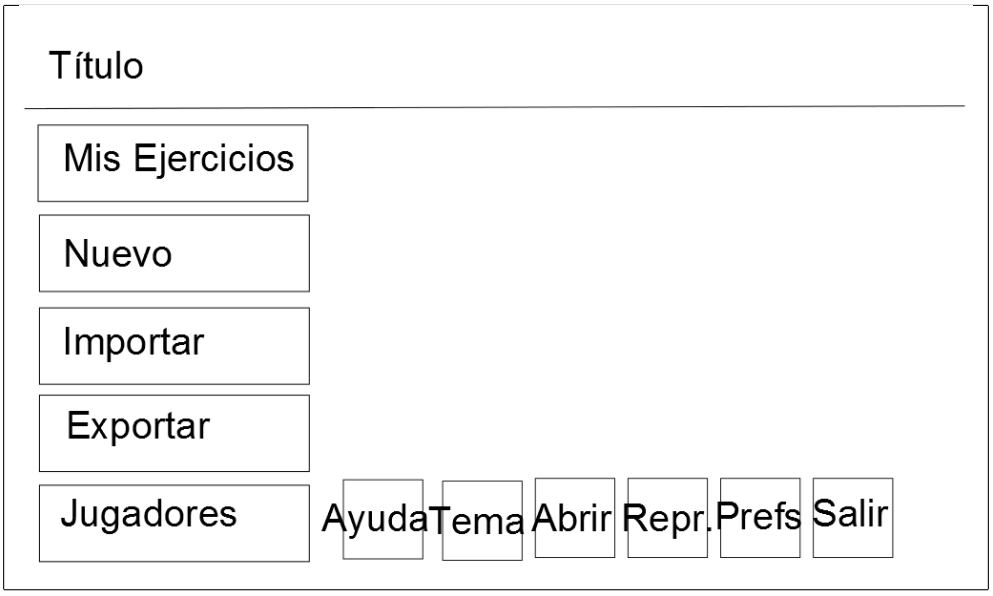


Figura B.2: Menu Principal

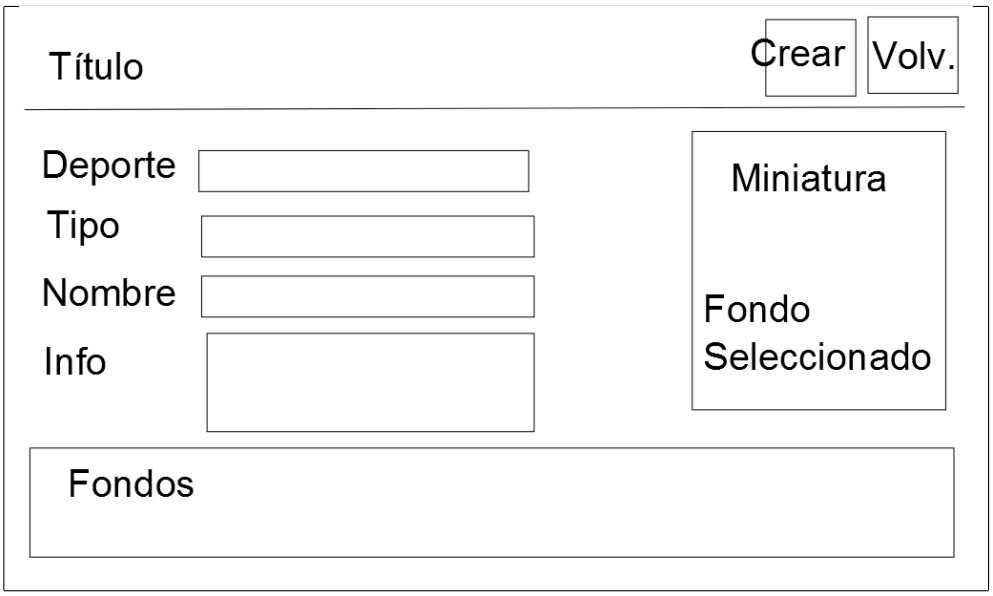


Figura B.3: Nuevo Ejercicio

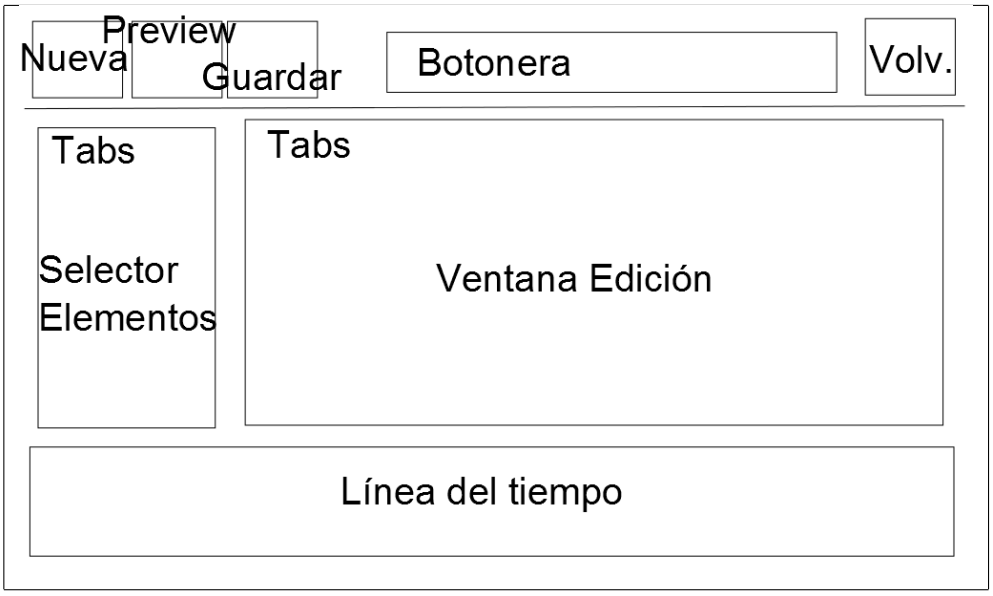


Figura B.4: Editor

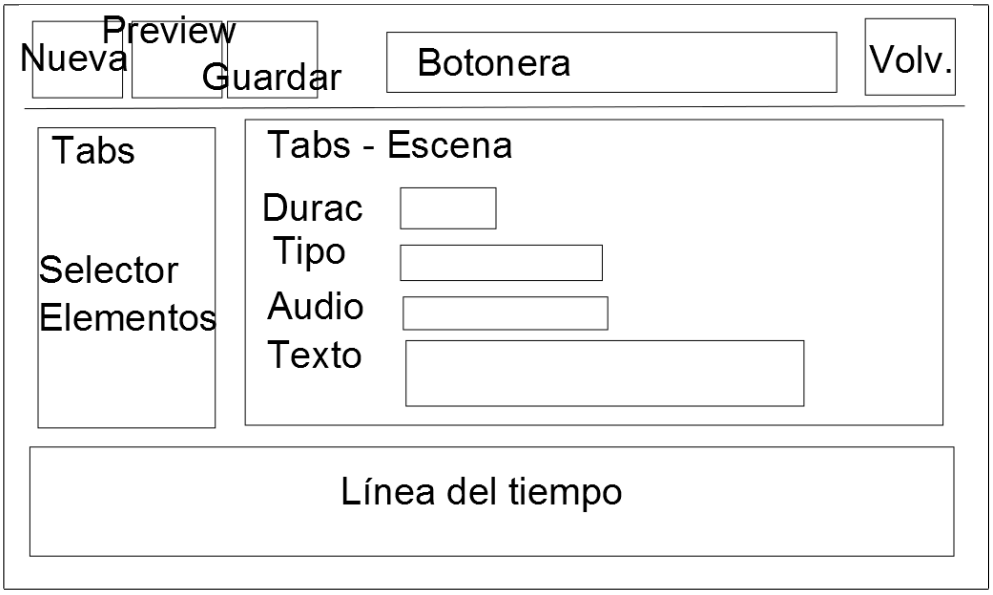


Figura B.5: Editor Escena

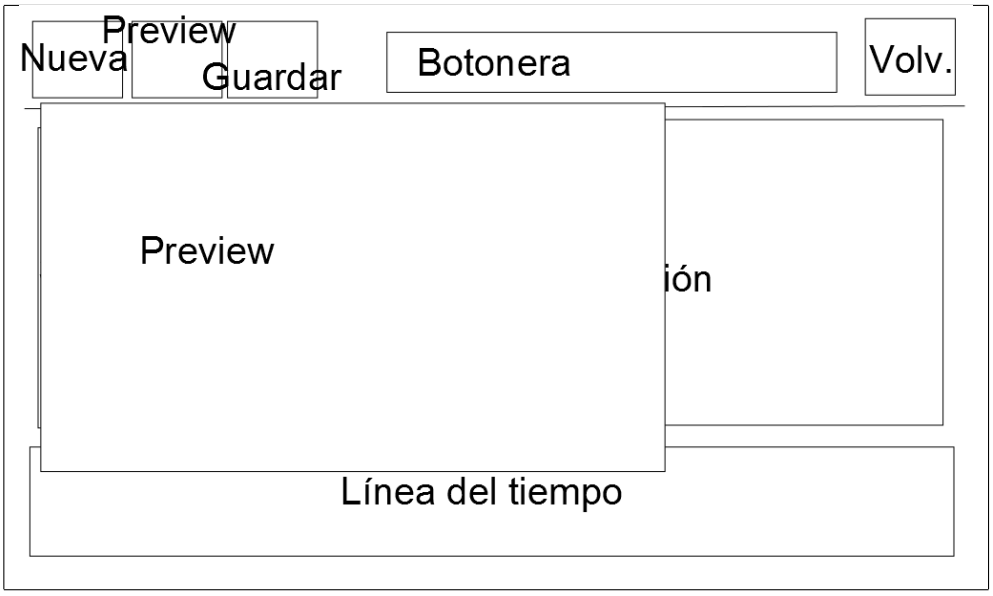


Figura B.6: Editor con Preview

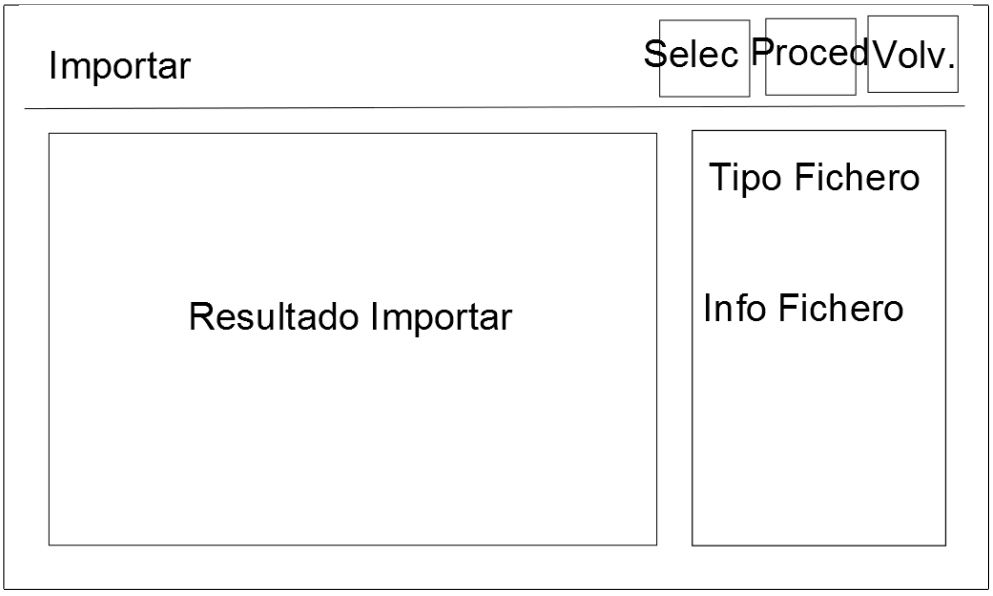


Figura B.7: Importación

B.3. Arquitectura del sistema

En esta sección se procederá a la descripción detallada del diseño de la aplicación de manera funcional y técnica.

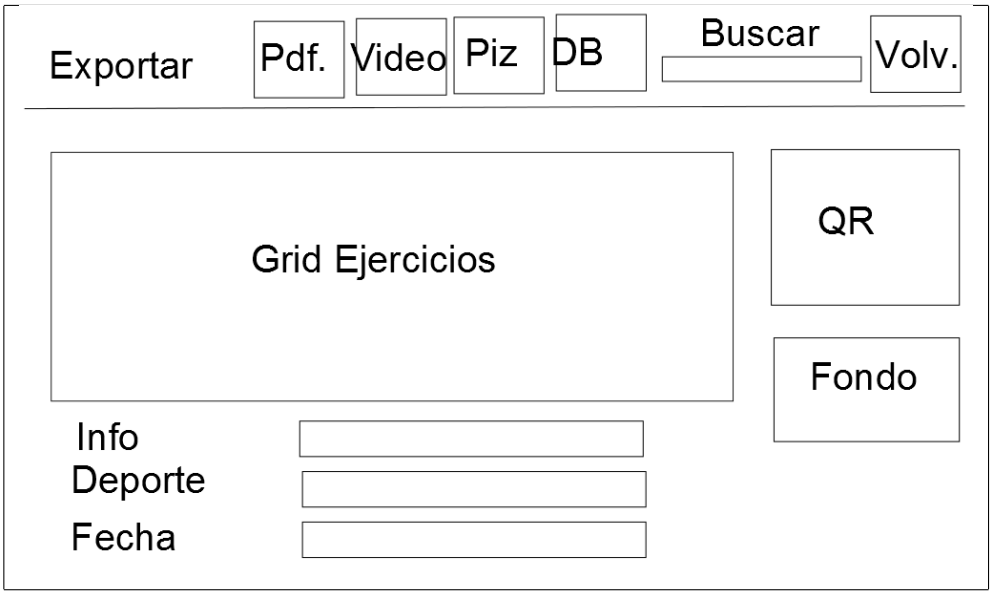


Figura B.8: Exportación

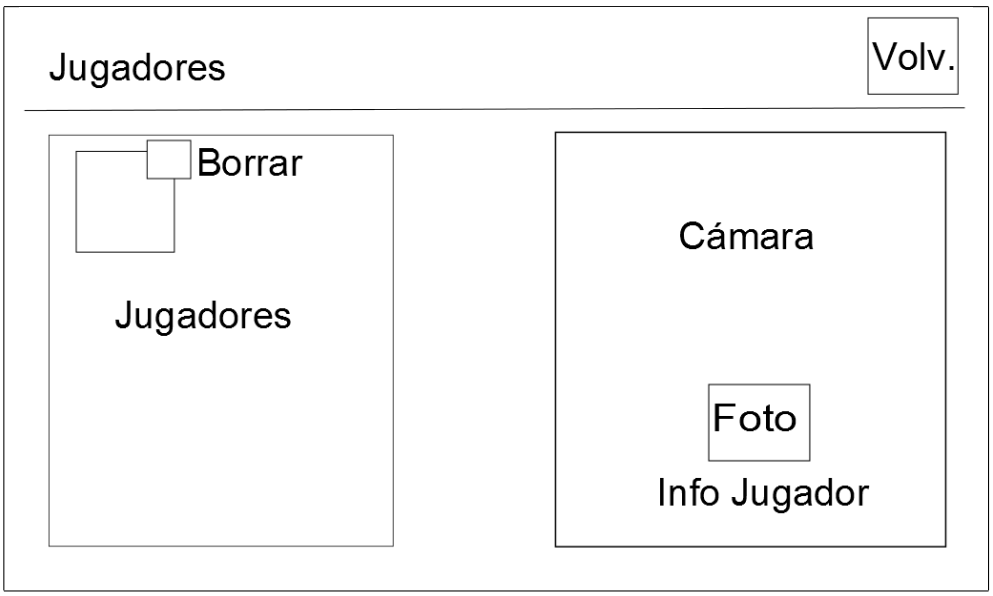


Figura B.9: Jugadores

B.3.1. Diseño funcional

A partir de la toma de requisitos de la aplicación, que se detalla en la seccion A.1, se procedió a ala definición de las distintas máscaras de las que consta la aplicación a alto nivel. Este diseño se realizó en base a wireframes[30], que se muestran en la sección B.2 de los anexos.

Una vez detallada la funcionalidad de la aplicación a nivel de interfaz gráfica de usuario, así como la interacción con el mismo, se comenzó con un diseño funcional

de alto nivel e independiente de la tecnología. Este diseño a alto nivel se realizó siguiento el patrón **Modelo-Vista-Controlador(MVC)** [31]. En este modelo se separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres niveles distintos.

- **Modelo:** Contiene los datos y la lógica para manejar el estado de la aplicación. Es responsable de responder acerca del estado de la aplicación y actualizarse en los cambios de estado.
- **Vista:** Presenta la interfaz de usuario y el estado de la aplicación en pantalla. Es responsable de la comunicación con el usuario.
- **Controlador:** Gestiona las entradas del usuario para cambiar el estado de la aplicación. Es responsable de determinar cómo las vistas responden a la interacción con el usuario.

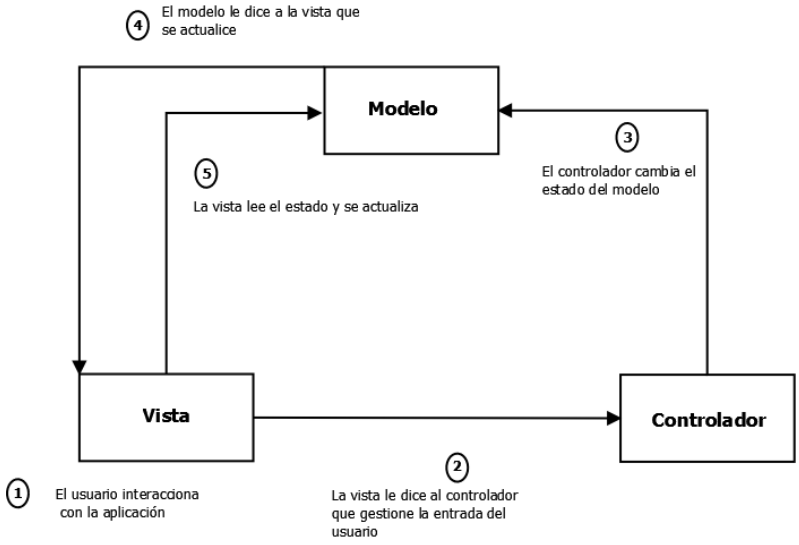


Figura B.10: Funcionamiento MVC + Observer

La ventaja del patrón MVC radica en la separación anteriormente nombrada sin solapar las responsabilidades de cada capa.

A su vez se ha utilizado el patrón **Observer** [32], que define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. Esto permitirá implementar de una manera sencilla un sistema en el que muchos componentes dependen de los eventos de otros componentes.

Con los siguientes fragmentos de código fuente se va a ejemplificar la implementación general del patrón en Actionscript 3, el lenguaje empleado en el proyecto.

Para implementar el **modelo**, se define una interfaz con los getters y setters necesarios para el acceso a cada uno de los atributos.

```
1  package{
2      import flash.events.*;
3
4      public interface IModel extends IEventDispatcher{
5          function setAtt(key: uint):void
6          function getAtt():uint
7      }
8  }
```

Figura B.11: IModel.as

Posteriormente se implementa esta interfaz, que genera un evento de cambio al cambiar el valor de algún atributo

```
1  package{
2
3      import flash.events.*;
4
5      public class Model extends EventDispatcher implements IModel{
6
7          private var attr: uint = 0;
8
9          public function setAtt(key: uint):void{
10             this.attr = key;
11             dispatchEvent(new Event(Event.CHANGE));
12         }
13
14         public function getAtt():uint{
15             return this.attr;
16         }
17     }
18
19 }
```

Figura B.12: Model.as

La implementación del **controlador**, también comienza generando una interfaz con los métodos a los que va a responder el controlador.

```
1  package{
2
3      import flash.events.*;
4
5      public interface IInputHandler{
6          function handler(evento: Event):void
7      }
8  }
```

Figura B.13: IInputHandler.as

Para conseguir la clase controlador, se implementa el anterior interfaz, al que se le debe pasar una referencia al modelo, para que puede mandarle actualizarse.

```actionscript
package{

    import flash.events.*;

    public class Controller implements IInputHandler{

        private var model: IModel;

        public function Controller(aModel: IModel){
            this.model = aModel;
        }

        public function handler(evento: Event):void{
            //  Ejemplo de acción que cambia el modelo
            model.setAtt(event.intCode);
        }
    }
}
```

Figura B.14: Controller.as

Para la implementación de una vista sencilla, que escribe en el terminal.

```actionscript
package{

    import flash.events.*;
    import flash.display.*;

    public class View{

        private var model: IModel;
        private var controller: IInputHandler;

        public function View(aModel: IModel,
                             aController: IInputHandler,
                             target: Stage){
            this.model = aModel;
            this.controller = aController;
            model.addEventListener(Event.CHANGE, this.update);
            target.addEventListener(intEvent, onEvent);
        }

        private function update(event:Event):void{
            trace(model.getAtt());
        }

        public function onEvent(evento: Event):void{
            controller.handler(evento);
        }
    }

}
```

Figura B.15: View.as

Para insertar la estructura del patrón dentro de su escena correspondiente se deben instanciar los objetos de las clases creadas

```actionscript
var model: IModel = new Model();
var controller: IInputHandler = new Controller(model);
var view: View = new View(model, controller, this.stage);
```

Figura B.16: escena.as

En la Figura B.10 se puede observar el funcionamiento del patrón, de la manera en la que se utilizará en el proyecto y la interacción entre las distintos elementos. La relación entre el modelo y la vista realizara a través de las suscripciones del patrón observer anteriormente citado, por lo tanto es una comunicación indirecta.

Esta arquitectura de la aplicación se compone de tres niveles distintos: la **interfaz** gráfica de usuario, los **componentes** software de la aplicación y la **base de datos**.

Para el diseño de la *interfaz de usuario* se han utilizado diagramas de navegación que se convirtieron en prototipos software que se sometieron a distintas pruebas de usabilidad y de validación. Como se puede ver en la parte superior de la figura 3.3, esta parte se sometió a varias iteraciones de pruebas, tanto de cliente como de usuarios, hasta conseguir la navegación y disposición más adecuados. Esta capa se corresponde con las distintas vistas y subvistas del patrón MVC, esta parte se describe con más detalle en la Sección B.6.

El *núcleo del sistema* se diseñó mediante un diagrama de componentes. El objetivo de este diagrama fue descomponer las distintas funcionalidades de la aplicación para cumplir todos sus objetivos. Posteriormente, cada componente fue refinado a nivel de diseño en los correspondientes diagramas de clases mostrados en el Anexo B. Esta parte encapsula las acciones a realizar en los controladores del patrón MVC. La sección B.4 profundiza en este apartado.

La capa inferior del sistema es la *base de datos*, como se puede observar en la figura. Esta parte se diseñó mediante un esquema entidad relación que representa los datos sobre los que funciona la aplicación. Para la implementación de dicha base de datos se transformó dicho esquema en Relaciones, y finalmente se convirtió dichas relaciones en tablas SQL que se insertarán en el SGBD utilizado por la aplicación, en este caso MySQL. En la Sección B.5 se detalla este diseño.

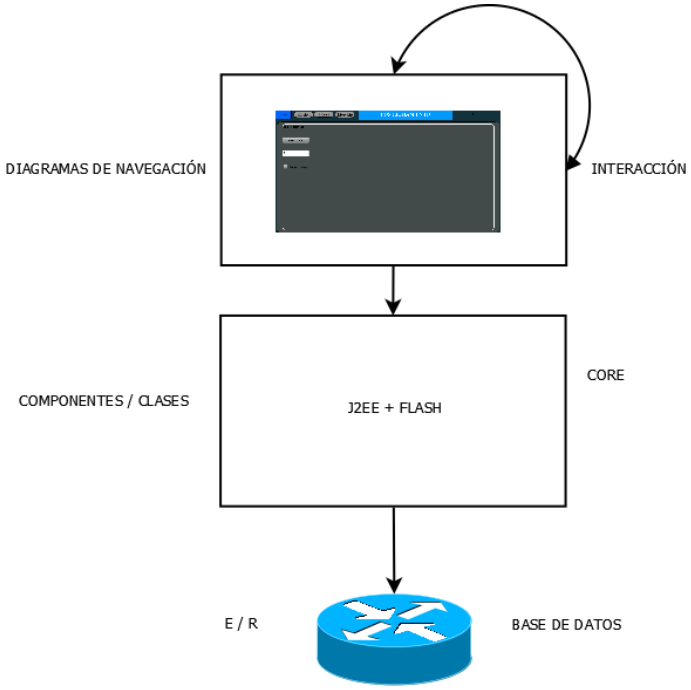


Figura B.17: Esquema de alto nivel del sistema completo

Cada una de estas capas se describe en detalle en los siguientes apartados del capítulo así como en el Anexo B.

B.4. Diseño de componentes de la aplicación

Para describir la estructura funcional a alto nivel de la aplicación se ha utilizado un diagrama de componentes. Este diagrama describe elos componentes de la aplicación y las dependencias existentes entre los mismos.

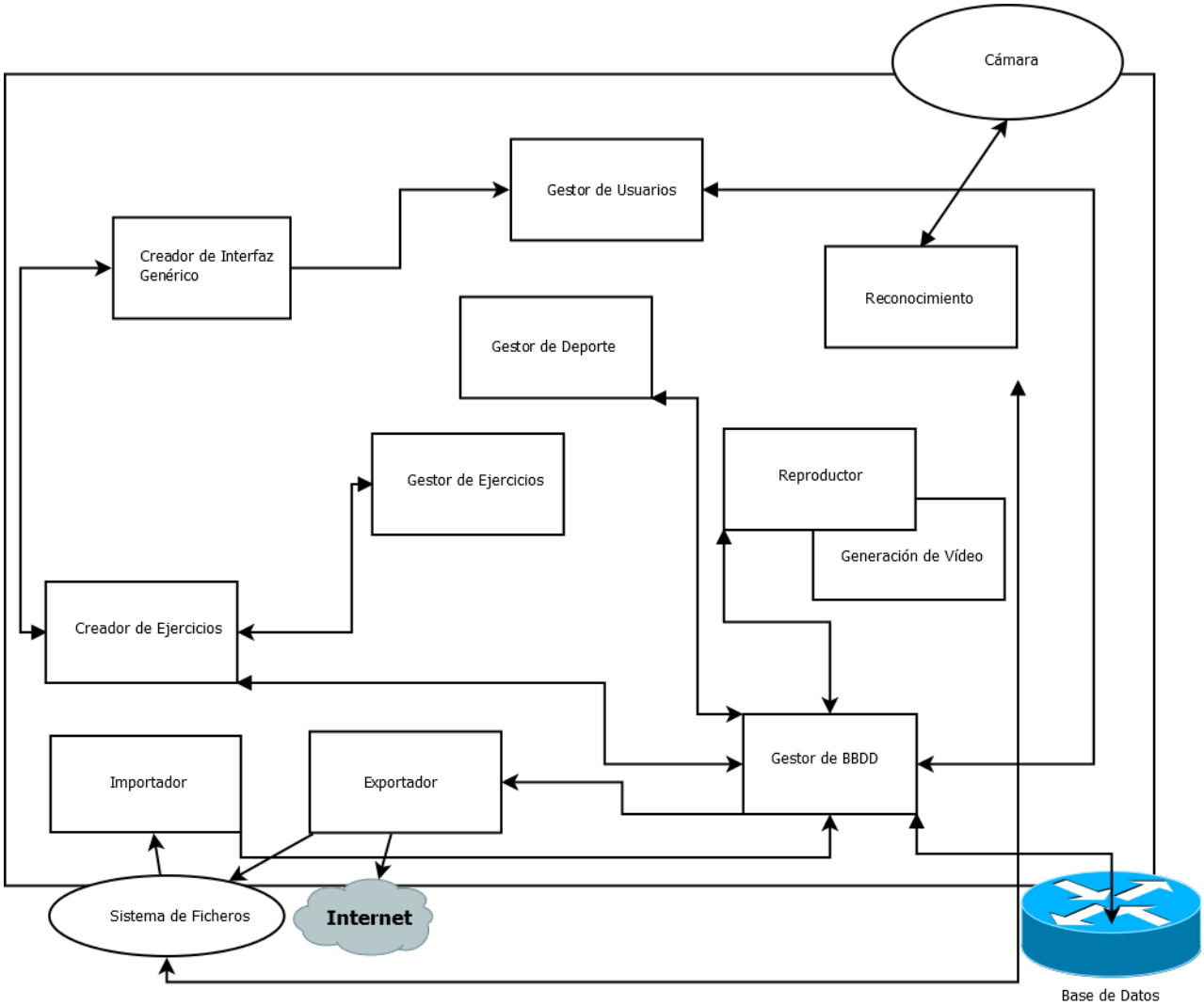


Figura B.18: Diagrama de componentes

En la Figura B.18 se puede observar dicho diagrama, en el que cada caja representa una un componente funcional del sistema completo, mientras que las flechas que unen las cajas son las interacciones entre los componentes. Una representación inicial a este nivel tiene sentido ya que se trata de un sistema complejo con funcionalidades muy distintas entre sí, y a partir de este diagrama, se podrá descomponer cada una de ellas como un subproblema más pequeño y diseñarlas utilizando los diagramas de clases y estados de UML. Estos diagramas UML utilizarán clases comunes a otros componentes, que serán reutilizadas.

En la Figura B.18 se puede ver que los componentes interaccionan con varios sistemas externos: una cámara web, una conexión a Internet, la base de datos y el sistema de ficheros local. Estos elementos están situados son externos al sistema de información desarrollado y han sido representados en óvalos.

En la parte inferior de la Figura B.18 se pueden ver los componentes relacionados con la gestión de datos. Por un lado, se necesitan componentes que sean capaces de *importar* distintos tipos de información desde el sistema de ficheros del computador. También se necesita un componente que *exporte* la información desde la pizarra al computador del usuario (sistema de ficheros) o a algún servidor remoto (Internet). Como fuente y destino de la información de los anteriores componentes se encuentra la *base de datos* de la aplicación. Por lo tanto se necesita un componente que se encargue de la interacción con dicha base de datos. Este gestor de Base de Datos será usado por otros componentes para la interacción con el repositorio de información.

En la parte izquierda de la Figura B.18 se encuentran los componentes de *gestión de ejercicio*s y de *creación de ejercicios*. La gestión se encarga de obtener los ejercicios de la base de datos y ponerlos en el formato adecuado para su manipulación, mientras que la creación de ejercicios se encarga de todo lo relacionado con las herramientas que se proporcionan al usuario para dibujar un nuevo ejercicio.

En la parte derecha de la figura se encuentran los componentes de *reproducción* y *generación de vídeo*. Han sido representados como dos componentes solapados ya que la gestión de vídeo se apoyará en la reproducción para producir el fichero de vídeo resultante.

El componente de *reconocimiento* se encarga de, a partir de la cámara o una imagen, generar un fichero con la cara de un jugador, que posteriormente podrá ser utilizado dentro del editor de ejercicios.

Además de estos componentes, se incluyen algunos más que realizan tareas generales de gestión de los elementos de la aplicación, como el *gestor de usuarios*, que permite ver la información de las personas que utilizan la aplicación, y el *gestor de deportes*, que permite la administración de los distintos paquetes de ejercicio instalados en el sistema.

Por último, en la esquina superior izquierda, se ve el componente *Creador de interfaz genérico*, que se encarga de tareas generales relacionadas con la interfaz y la creación de algunos componentes genéricos(Clases base de Vista, inclusión del título, registro de eventos para la navegación de manera genérica). En este componente se apoyarán las vistas y subvistas que se crearán al utilizar el patrón MVC.

Los ejercicios no son sólo almacenados en base de datos. temporalmente también pueden ser alamacenados en el componente Gestor de Ejercicios, que actua como una "caché" de ejercicios. Consideramos relevante describir cómo este componente almacena estos ejercicios. La Figura B.19 describe la disposición lógica de los elementos que forman un ejercicio.

Los ejercicios forman una jerarquía. Cada ejercicio contiene escenas, que agrupan los movimientos de elementos en un periodo discreto de tiempo. Cada uno de estos movimientos que componen una escena está asociado a un elemento, y describe una trayectoria concreta a través de una seri de puntos.

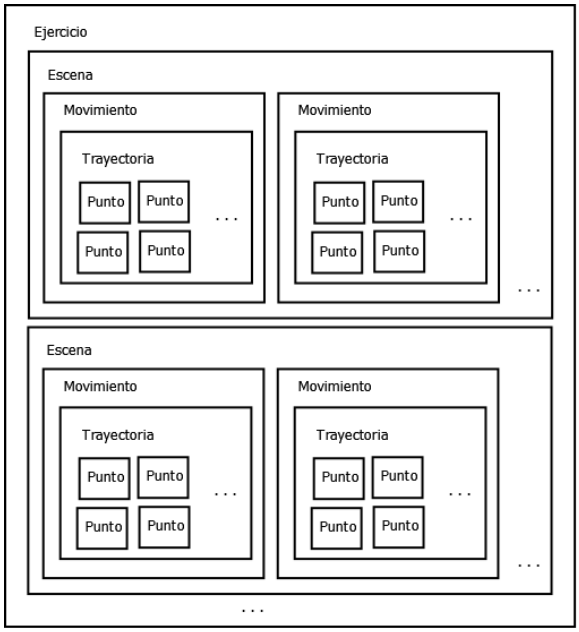


Figura B.19: Disposición lógica de un ejercicio

Utilizando la agregación de UML, el diagrama de clases de esta parte aparece en la Figura B.20.

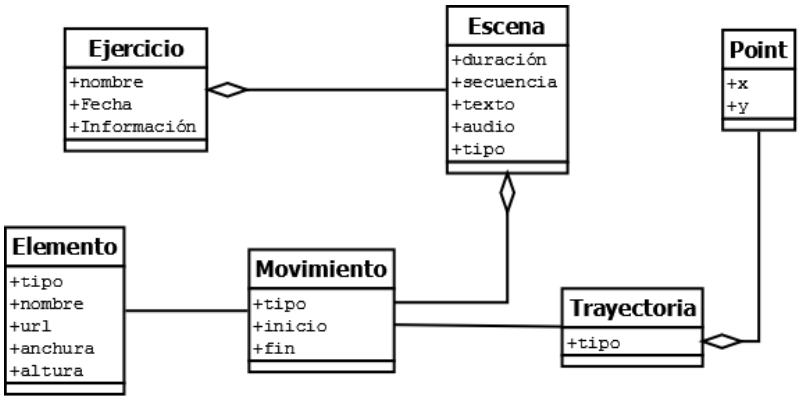


Figura B.20: Diagrama de clases de ejercicio

Las distintas entidades representadas en el diagrama lógico están encapsuladas en clases, y cada una va agregando los elementos que la componen hasta llegar a los puntos que forman una trayectoria en último lugar. Para caracterizar el elemento que realiza el movimiento, se relaciona la clase elemento con la clase movimiento.

B.4.1. Base de Datos

Se ha diseñado una clase que encapsula el comportamiento de la base de datos, se encarga de conectarse y desconectarse, así como leer y escribir la información que sea necesaria.

Figura B.21: Base de Datos

B.4.2. Registrador de Interfaz

En el lenguaje utilizado para la programación del proyecto, los elementos creados dinámicamente no se liberan ni eliminan de la pantalla al hacer un cambio de escena, es un proceso que se debe realizar manualmente. Para hacer este trabajo más sencillo se ha diseñado un elemento software en el que se añaden los elementos que se insertan dinámicamente en la pantalla, para luego simplemente ordenar al objeto de la clase que limpie la pantalla.

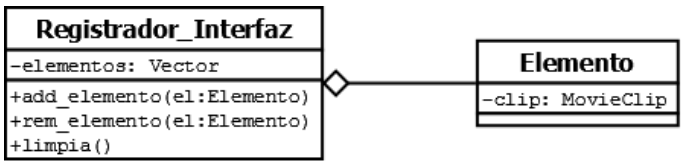


Figura B.22: Registrador de elementos

B.4.3. Usuario y preferencias

Para almacenar la información de los usuarios y las preferencias del sistema se utilizarán dos clases: Usuario y Preferencias.

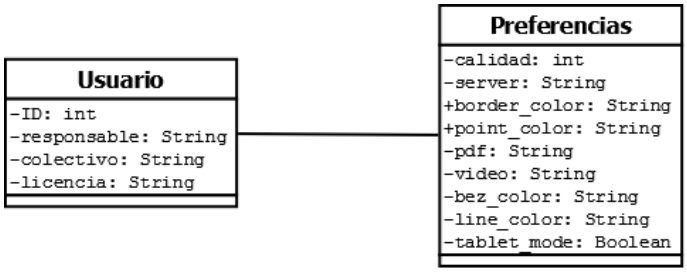


Figura B.23: Preferencias y Usuarios

B.4.4. Deportes y Elementos

Cada deporte dispone de una serie de elementos, que pueden a su vez pertenecer a varios deportes distintos, para encapsular la información de los mismos se utiliza la siguiente jerarquía de clases.

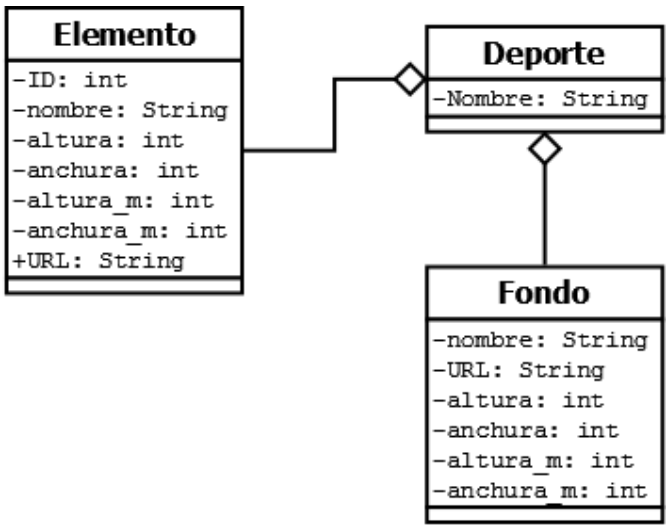


Figura B.24: Deporte, fondo y elemento

En esta estructura se encapsula la información de los elementos, paro a partir de la clase elemento, se obtienen otros elementos, que integran el controlador de los objetos que se visualizan en pantalla como se muestra en la siguiente figura.

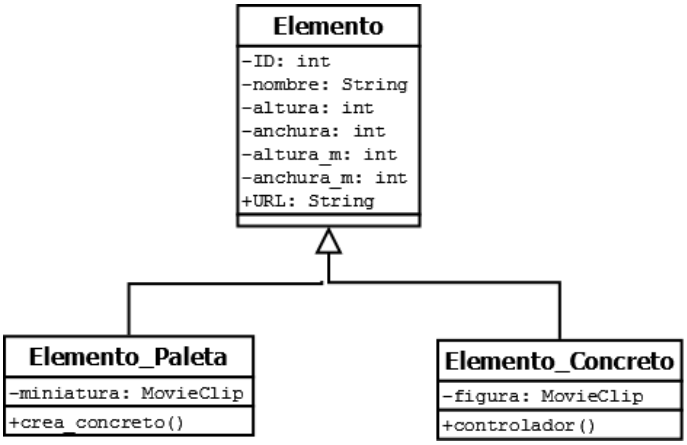


Figura B.25: Elemento concreto y elemento paleta

Elemento_Paleta es el objeto que se inserta en el selector de elementos, el elemento gráfico que permite añadir elementos a la escena, una vez se arrastra un Elemento_Paleta desde el selector hasta la escena, se crea un Elemento_Concreto, que gestiona todas las interacciones con él mismo. Está relacionado con varias clases auxiliares que encapsulan el modelo utilizado por el componente como se muestra en la siguiente figura.

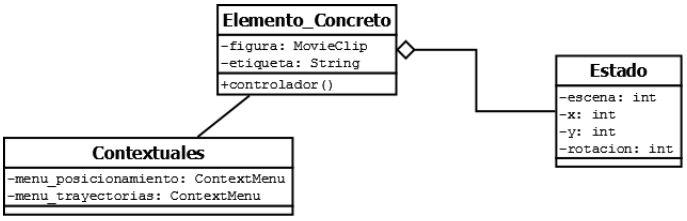


Figura B.26: Elemento concreto

Elemento concreto encapsula tanto las posiciones, como las trayectorias de las distintas escenas en las que interviene.

B.4.5. Mis Ejercicios

Para la gestión de ejercicios se necesita de un componente gráfico que permita seleccionar los ejercicios, que se encapsulará en la clase Selector_Ejercicios, donde se incluirá la vista y el controlador de los ejercicios, así como las funciones de búsqueda sobre el vector de ejercicios.

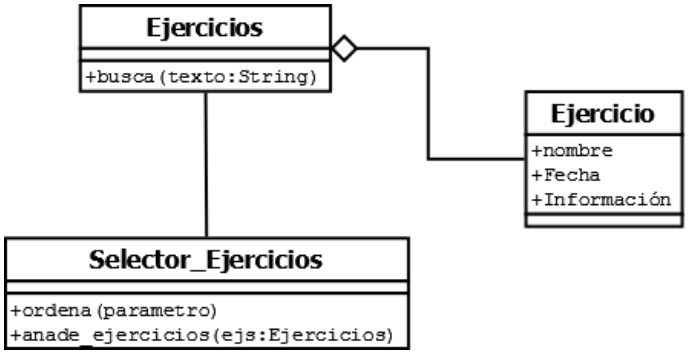


Figura B.27: Mis ejercicios

B.4.6. Reproductor

El reproductor es una de las partes más complejas de la aplicación, se ha diseñado siguiendo un patrón de precálculo. El sistema está limitado por la carga de los audios de los ejercicios, por lo tanto se ha de esperar a los mismos para conocer su duración y por lo tanto adecuar la duración de las transiciones, para ver la manera en la que funciona el reproductor se modelará mediante un diagrama de estados.

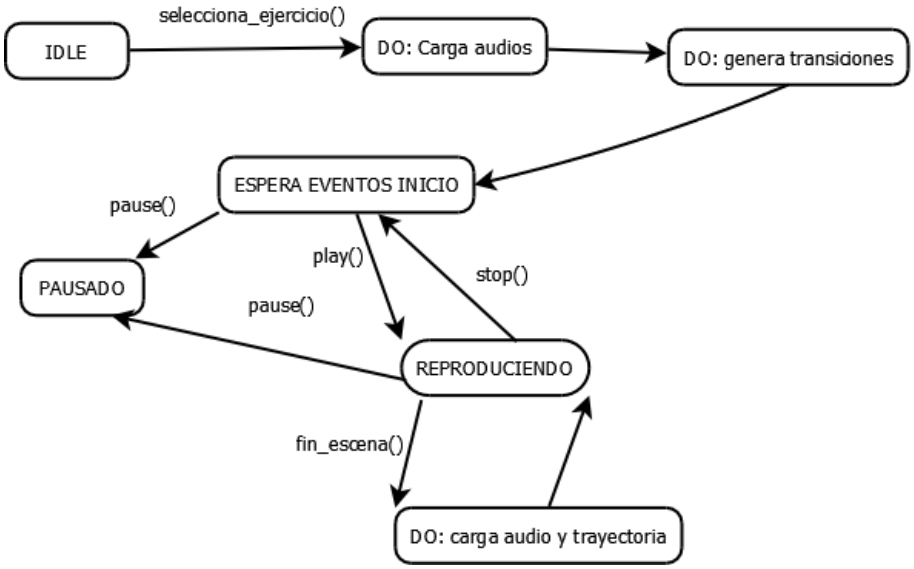


Figura B.28: Reproductor

El sistema de reproducción se basa en la clase Reproductor así como en un mediador que transmite el reproductor las pulsaciones desde la interfaz de usuario y la botonera del reproductor.

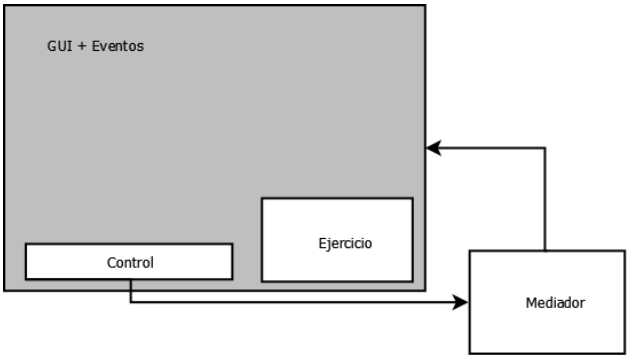


Figura B.29: Estructura eventos

B.4.7. Editor

El editor es un sistema principal en la aplicación y en este caso está compuesto por varios elementos gráficos a medida que se modelan en las siguiente figuras

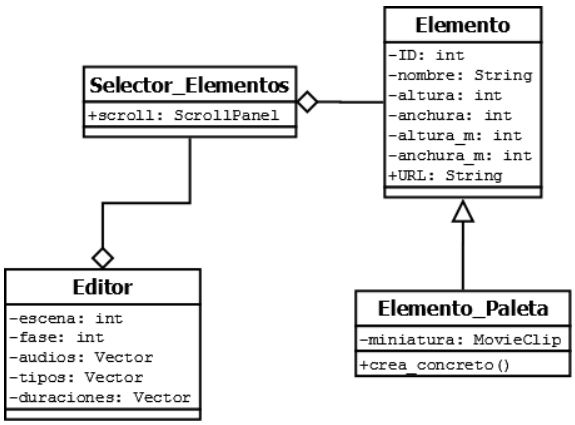


Figura B.30: Selector de elementos

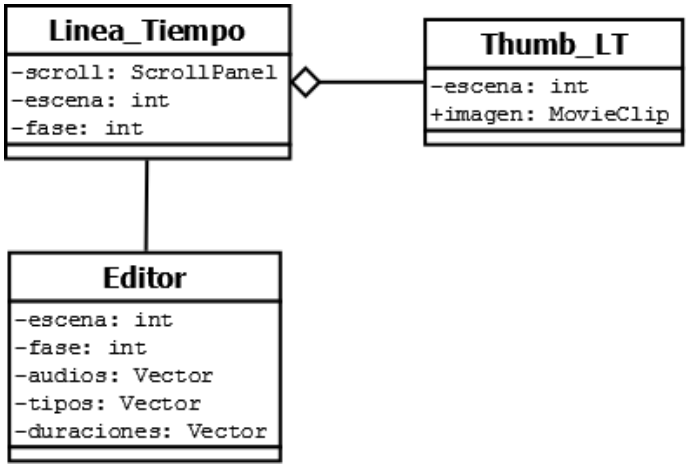


Figura B.31: Línea del tiempo

Para modelar el comportamiento del editor se muestran a continuación una serie de diagramas de estados, en el primero de ellos se observa la interacción de cada elemento concreto con el editor, y en el segundo de ellos se observa los cambios relacionados con la línea del tiempo.

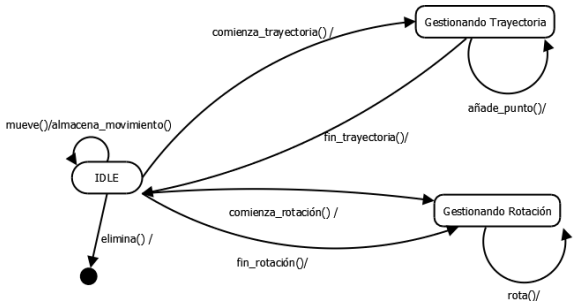


Figura B.32: Diagrama de estados Elemento_Concreto

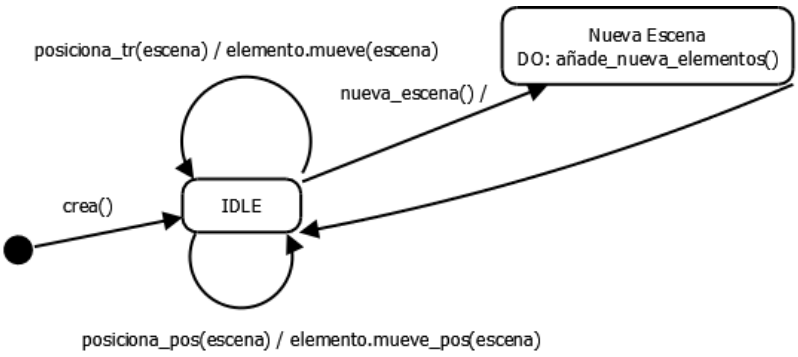


Figura B.33: Diagrama de estados Línea_Tiempo

B.5. Diseño de la base de datos de la aplicación

A continuación, la figura 3.7 muestra el diagrama Entidad -Relación de la aplicación.

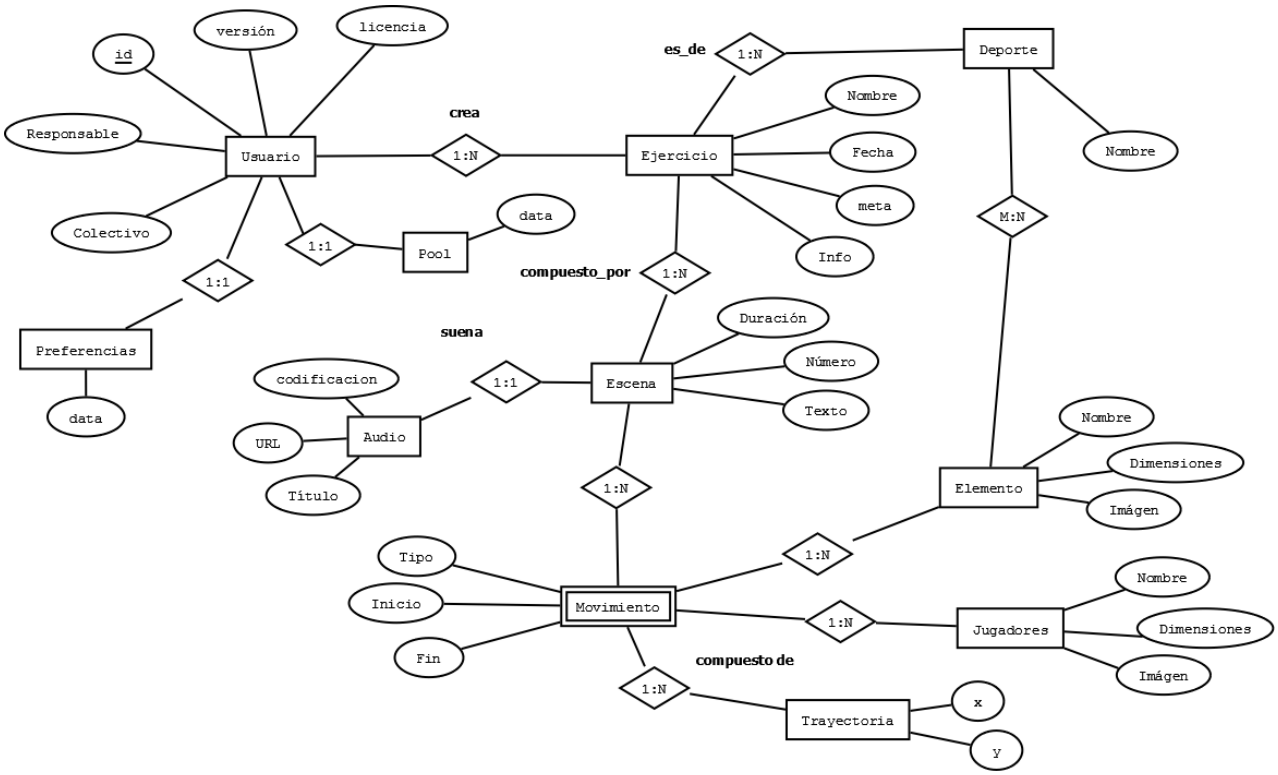


Figura B.34: Esquema Entidad-Relación

En la figura se puede ver como las entidades y las relaciones son muy similares a las clases del diseño explicado en el punto anterior. Se dispone de una entidad *Ejercicio* que contiene la información general y que está compuesto por *escenas*, y éstas, a su vez tienen varios *movimientos* que están formados a su vez por varios *puntos* que representan una Trayectoria circular, lineal o zig-zag.

También se almacenan en la base de datos las preferencias generales del usuario, que incluyen los colores de las trayectorias o el comportamiento de la aplicación ante distintos eventos entre otros. Los jugadores del equipo en sus correspondientes entidades *Usuario* y *Jugadores* también son guardados en el modelo de datos, siendo las urls a las imágenes que posteriormente mostrará la aplicación la parte más importante.

Por último un deporte está compuesto por varios elementos distintos, que pueden ser comunes a varios deportes. Estos *elementos* junto con los jugadores son lo que se representa en movimientos, es decir las transiciones en las escenas o bien son elementos de deportes (jugadores, balón, etc), o bien son las caras de los jugadores.

En las figuras B.27, B.28 y B.29 se puede ver un ejemplo gráfico de como se almacena la información de los ejercicios en la base de datos.

Usuario

ID	Version	Licencia	Responsable	Colectivo
1	0,1	Ilimitada	Alberto Bolsa	STG

Deporte

ID	Nombre
1	Fútbol sala
2	Baloncesto
3	Balonmano

Elemento

ID	Nombre	altura	anchura	Imagen
1	Cono	10	20	cono.jpg
2	Balón	10	10	balon.jpg
3	Jugador	10	30	jugador.jpg

Ejercicio

Nombre	Fecha	Info
EJ1	10/01/2010	test 1

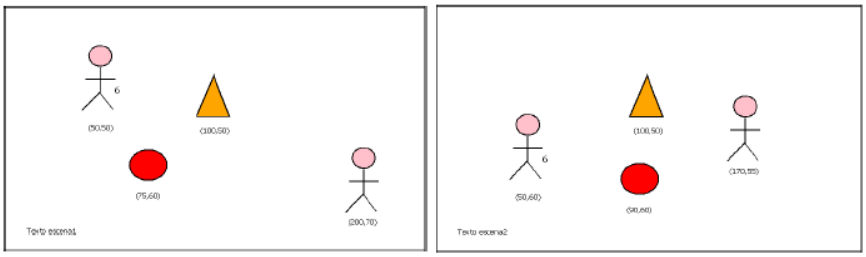


Figura B.35: Almacenamiento en la base de datos(I)

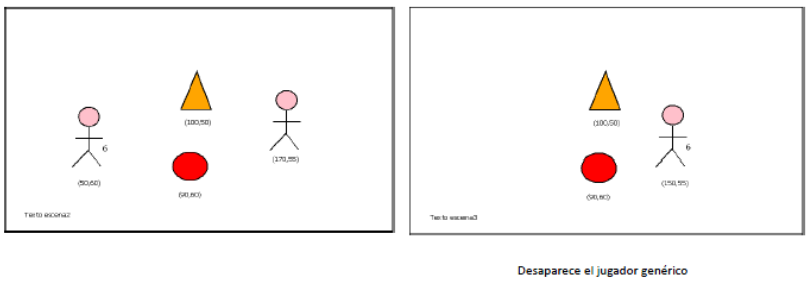
Escena

ID	Duración	Número	Texto
1	100	1	Texto escena 1

Se podría detectar el movimiento del cono como nulo y no programar la animación

Movimiento

Escena	Tipo	x_inicio	y_inicio	x_fin	y_fin	Elemento	Etiqueta
1	lineal	50	50	50	60	3	6
1	estático	100	50	NULL	NULL	1	NULL
1	lineal	75	60	90	60	2	NULL
1	lineal	200	70	170	55	3	NULL



Desaparece el jugador genérico

Figura B.36: Almacenamiento en la base de datos(II)

Escena

ID	Duración	Número	Texto
1	100	1	Texto escena 1
2	100	2	Texto escena 2

Movimiento

Escena	Tipo	x_inicio	y_inicio	x_fin	y_fin	Elemento	Etiqueta
1	lineal	50	50	50	60	3	6
1	estático	100	50	NULL	NULL	1	NULL
1	lineal	75	60	90	60	2	NULL
1	lineal	200	70	170	55	3	NULL
2	lineal	50	60	150	55	3	6
2	estático	100	50	NULL	NULL	1	NULL
2	estático	90	60	NULL	NULL	2	NULL
2	desaparecer	170	55	NULL	NULL	3	NULL

Figura B.37: Almacenamiento en la base de datos(III)

B.6. Diseño del interfaz

El diseño de la interfaz gráfica de usuario del proyecto se abordó desde tres puntos de vista distintos: organización de los elementos del producto y usabilidad, la combinación de colores, y la compatibilidad con tablet PCs.

B.6.1. Organización de los elementos

El elemento más importante es el editor de jugadas. Este elemento supone la mayor parte de las entradas del usuario hacia la aplicación y su correcto y adecuado funcionamiento depende de la interfaz escogida. Este modelo de editor debe contener una serie de elementos básicos para poder alcanzar el objetivo:

- **Selector de elementos:** Elemento que permita añadir objetos a la escena.
- **Línea del Tiempo:** Componente que permite al usuario moverse entre las distintas encenas.
- **Editor de la escena:** Panel principal sobre el que se distribuyan los elementos.
- **Selector de Terrenos:** Paleta con los distintos fondos.
- **Edición de trayectorias:** Sistema de definición de trayectorias.

Para el desarrollo del editor se realizaron varios prototipos sobre los que se fueron haciendo cambios. En la Figura B.38 se muestra la primera versión de este prototipo de editor.

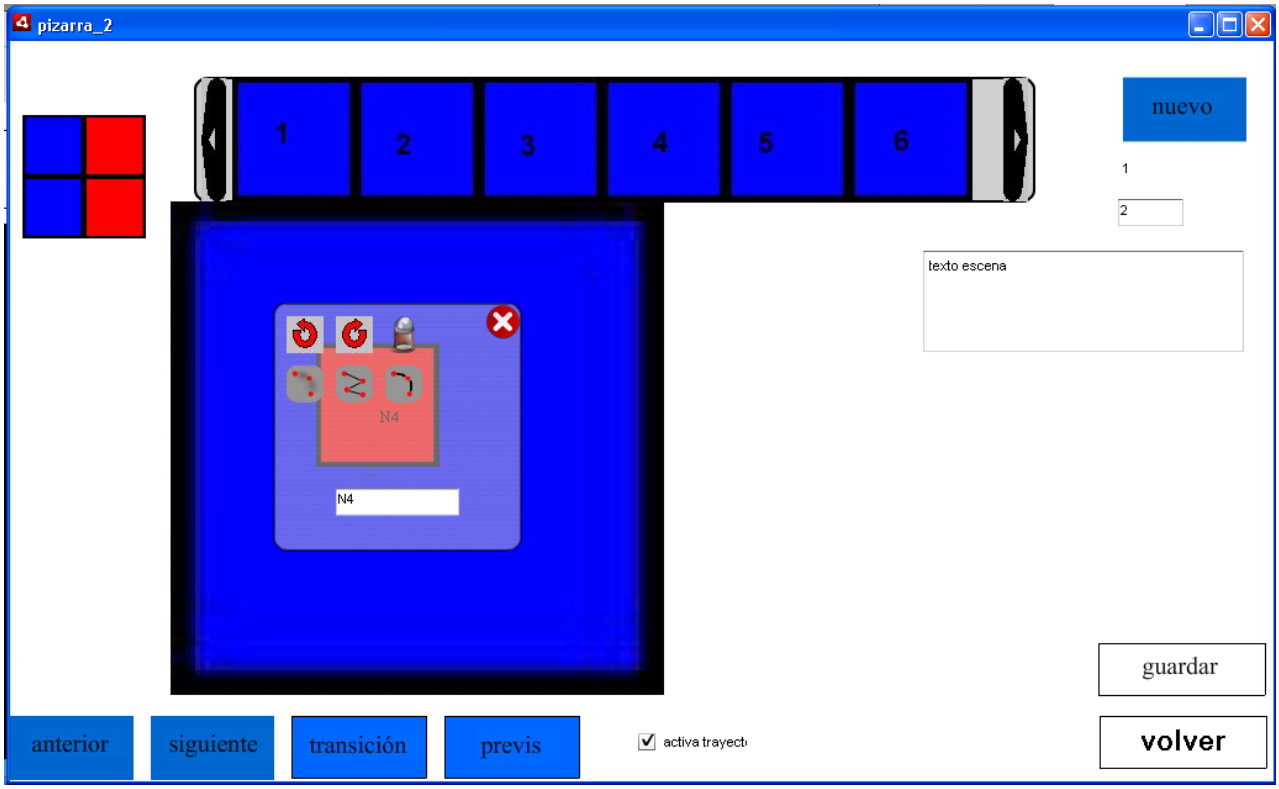


Figura B.38: Primera versión del interfaz

Esta primera versión contaba con el selector de terrenos en la ventana del editor y los elementos se disponían en un solo panel en la parte izquierda de la aplicación(Cuadrados rojos y azules). Los elementos se arrastrában sobre la zona azul que representa el panel de la escena. Para realizar cualquier acción sobre un elemento hay que hacer doble click y un menú contextual aparece sobre el mismo, donde se puede eliminar, crear la trayectoria o cambiar la etiqueta del mismo.

Para cambiar entre escenas se pulsaba sobre el botón nuevo, y para moverse entre las mismas sobre anterior o siguiente.

En este primer prototipo se detectó que un panel encima del elemento podría impedir el acceso a otros elementos y fue una solución descartada a la hora de gestionar las opciones de los elementos. También hizo falta diseñar una mejor manera de moverse entre escenas, ya que los botones de anterior y siguiente no resultaban intuitivos.

A partir de estas pautas obtenidas con el prototipo se procedió a especificar un segundo prototipo que se puede ver en la Figura B.39. La especificación completa está disponible en el Anexo B.

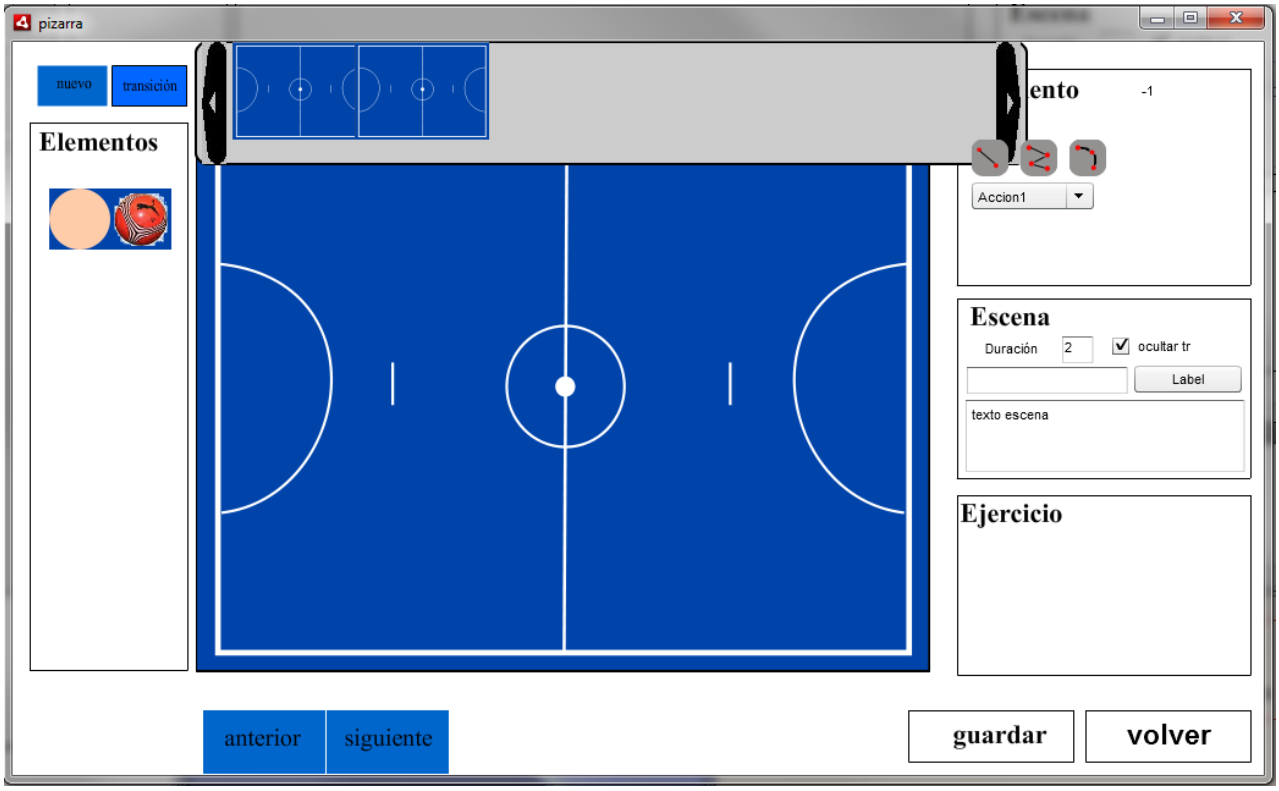


Figura B.39: Segundo Prototipo

Sustituyendo el panel modal que aparecía sobre el elemento, en esta versión se experimentó con varios paneles a la derecha para configurar los elementos y también se hizo el selector de terrenos desplegable desde la parte de arriba.

Se observó que no es necesario tener el selector de terrenos presente en todo momento en la edición y se puede desplazar a la pantalla previa, donde se introduce la información del ejercicio, y que quede fijo durante la edición gráfica del mismo. La información que aparece en los paneles de la derecha, ocupa un 20 % de la pantalla, y no se utiliza con mucha frecuencia, por lo tanto un sistema alternativo que emplee ese espacio en panel de dibujo resulta mas interesante.

La siguiente versión fue la versión final del editor de ejercicios en la Figura B.40.

Figura B.40: Versión final editor

Esta versión se complementa con un línea del tiempo con las miniaturas de las posiciones de las escenas en la parte inferior de la pantalla, y un sistema de pestañas para realizar la edición de los parámetros, encima del panel de edición. La selección de los terrenos se realiza desde la pantalla anterior, así se puede utilizar el espacio de este elemento para el resto de componentes.

B.6.2. Combinación de colores

Para la selección de las paletas de colores utilizadas en la aplicación se tuvo en cuenta en contexto en el que la aplicación se podría utilizar, en este caso, la mayoría del tiempo de uso es dentro de la edición o la reproducción de ejercicios, por lo tanto con el terreno presente. Los terrenos más comunes son, el azul de goma de fútbol sala y balonmano, el verde césped de fútbol y el naranja parquet de baloncesto. La paleta escogida tiene que contrastar con estos colores para que el terreno no se integre por completo en la aplicación y no se vea bien.

Para la elección de la paleta se ha utilizado la aplicación Adobe kuler[37] en la Figura B.41, que permite la elección de colores partiendo de combinaciones base. Además ofrece toda la información de los colores en las distintas codificaciones para luego utilizarla en el código fuente.

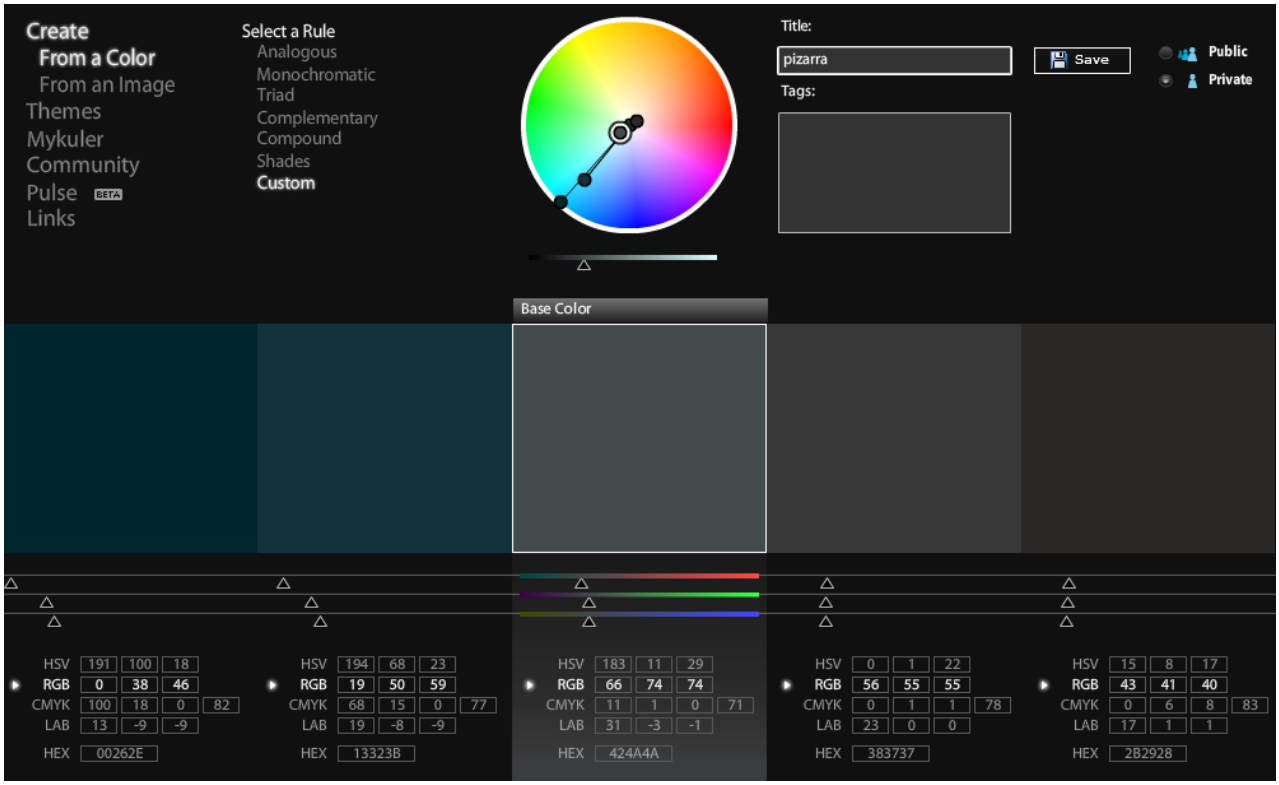


Figura B.41: Paleta de colores

B.6.3. Compatibilidad con tablet PC

La aplicación hace uso para su funcionamiento normal del botón derecho y de los eventos de MouseOver (pasar el ratón por encima de elementos para cambiar su estado), como por ejemplo en la aparición de las barras de scroll al pasar por encima el ratón. Estas características ofrecen una gestión del espacio mejor ya que aparecen sólo cuando son necesarias si se está usando un ratón para manejar la aplicación, pero no están disponibles para usuarios de tablet PC. Como éstos son usuarios potenciales, se ha habilitado una opción de configuración para establecer cuando se está utilizando la aplicación desde un tablet PC.

Cuando esta opción está activada, las acciones que se activaban con clicks derechos de ratón ahora se activan con dobles clicks, mientras que los elementos que aparecían al hacer MouseOver, se fijan a la interfaz, como las barras de scroll.

B.7. Diseño Técnico

Una vez definidos los componentes funcionales de la aplicación se debe definir a nivel técnico la aarquitectura de la aplicación en la tecnología sobre la que se

implementará la solución.

Como se describió en la Sección B.4, la implementará basándose en los patrones MVC y Observer. A esto hay que añadir una implementación con arquitectura n-capas [39], que se describe a continuación.

El patrón de diseño en n-capas fue definido por Microsoft para su utilización en proyectos software basados en el framework .net.

La idea de la arquitectura es la separación de la aplicación en distintas partes poco acopladas, y que permiten la abstracción de distintos componentes de manera genérica, lo cual mejora la escalabiliadad del sistema. La pila simplificada sobre la cual se trabajará en este proyecto se puede ver en la Figura B.42

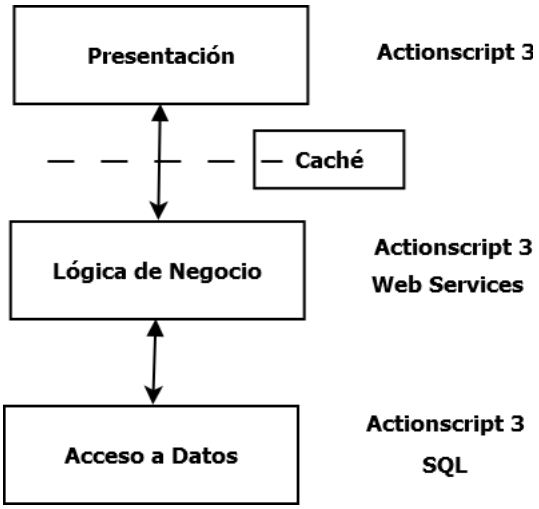


Figura B.42: Modelo n-capas

La capa de presentación define como se visualiza la aplicación. La capa de caché almacena información que se envía de manera frecuente entre la capa de Presentación y la de lógica de negocio. Ésta última es la que sabe que debe hacer la aplicación, y se encarga de comunicar las interacciones del usuario con la capa de acceso a datos, que es la que da persistencia al sistema.

Para hacer funcionar ambos patrones de diseño a la vez, se ha realizado la siguiente implementación.

Las Vistas engloban todos los componentes gráficos de la aplicación y sólamente la parte gráfica. De esta manera, si se desea cambiar el front-end o realizar un port a otro lenguaje compatible, como pudiera ser en este caso FLEX, sólo se sebería cambiar esta parte del sistema.

Los Controladores de cada pantalla capturan y tratan los eventos de los controles en la vista, y se encargan de llamar a las funciones de la lógica de negocio, la parte de la aplicación que sabe que acciones realizar. Es esta capa la que accede a los datos a través de los modelos.

Debido a que los datos que se deben tratar en la aplicación no son de un tamaño excesivo, y que esta información se intercambia de manera frecuente entre las distintas partes, se ha optado por un objeto *modelo* común, ya que el impacto en el rendimiento de la aplicación es inperceptible, y simplifica las labores de programación de manera considerable.

Este objeto modelo es el encargado de llamar a las funciones de la capa de acceso a datos. Cada entidad de la base de datos tiene su clase de acceso a datos, por lo tanto cuando se necesita información acerca de una de ellas se debe llamar a la función correspondiente de su Data Access Object. estos objetos suponen una abstracción de alto nivel sobre las acciones comúnes de lectura, actualización, borrado e inserción. Cuando sea necesario se pueden extender estas clases para obtener funcionalidad más específica. La mayor ventaja del sistema es que si se desea migrar a otro sistema gestor de base de datos, o fuente de datos, se debería añadir una implementación concreta, pero el proxy de la capa de acceso a datos seguiría siendo el mismo, por lo tanto no se deberían hacer cambios en la aplicación.

B.7.1. Arquitectura de Componentes

Los elementos gráficos de los que se dispone en el framework por defecto son muy limitados, y el contexto de la aplicación hizo necesaria la implementación de un gran número de controles específicos. Para mantener la coherencia entre los distintos elementos se diseñó una interfaz común para los mismos. La especificación completa se encuentra en el anexo B.

Apéndice C

Pruebas del sistema

En el presente capítulo se describen las pruebas generales que se han efectuado sobre el sistema para comprobar su correcto funcionamiento.

Para la realización de las mismas se han utilizado casos de uso para describir las acciones que realizaría el usuario, incluyendo dentro de estas acciones las que pudieran producir un error para ver si la respuesta es la adecuada.

Cada una de las pruebas se compone de tres partes, una descripción de la prueba a realizar, así como del objetivo principal de la misma, el caso de uso que se seguirá en la prueba, y por último el resultado de la prueba.

C.1. Acceso a un ejercicio guardado

Descripción: Prueba de acceso a un ejercicio guardado dentro de la aplicación.

1. Acceder al menú ejercicios
2. Buscar el ejercicio del listado
3. Hacer doble click sobre el mismo

Resultado: Satisfactorio, se accede a la animación del ejercicio.

C.2. Buscar en ejercicios

Descripción: Prueba del correcto funcionamiento de la búsqueda de ejercicios.

1. Acceder al menú ejercicios

2. Escribir un parámetro el parámetro de búsqueda "ejercicio" (existe en el campo nombre de varios ejercicios)

Resultado: Satisfactorio, se filtran los ejercicios que contienen la palabra que se busca.

C.3. Buscar en ejercicios (ERROR)

Descripción: Prueba del correcto funcionamiento de la búsqueda de ejercicios cuando no hay resultados.

1. Acceder al menú ejercicios
2. Escribir un parámetro el parámetro de búsqueda "noexiste" (no hay coincidencias)

Resultado: Satisfactorio, la lista de ejercicios es vacía.

C.4. Eliminación de un ejercicio

Descripción: Prueba de la eliminación de un ejercicio de los disponibles en la aplicación.

1. Acceder al menú ejercicios
2. Seleccionar un ejercicio
3. Pulsar sobre borrar

Resultado: Satisfactorio, el elemento se elimina.

C.5. Reproducción de un vídeo flv desde la aplicación

Descripción: Prueba del correcto funcionamiento de la reproducción de vídeos flv desde la aplicación

1. Pulsar sobre el botón de reproducción de vídeos
2. Seleccionar el vídeo del navegador de ficheros
3. Pulsar el botón abrir

Resultado: Satisfactorio, la reproducción del vídeo comienza.

C.6. Cambiar las preferencias del usuario

Descripción: Prueba del correcto funcionamiento de las preferencias del usuario

1. Pulsar el botón de preferencias
2. Cambiar los colores de las líneas y los puntos
3. Pulsar el botón guardar
4. Pulsar el botón volver
5. Pulsar el botón preferencias

Resultado: Satisfactorio, los valores se han guardado correctamente

C.7. Creación de un ejercicio

Descripción: Prueba de la creación de un ejercicio desde la aplicación.

1. Pulsar sobre el botón nuevo
2. Seleccionar el deporte "Fútbol Sala"
3. Seleccionar el tipo "Ejercicio"
4. Escribir el nombre y la información del ejercicio
5. Seleccionar el primer terreno disponible
6. Pulsar sobre crear
7. Arrastrar un jugador a la escena
8. Pulsar sobre el botón "+"
9. Hacer click derecho en el jugador
10. Seleccionar trayectoria zig-zag
11. Pulsar en la pantalla sobre los puntos del esquema
12. Pulsar sobre el botón guardar

Resultado: Satisfactorio, el ejercicio se ha almacenado, y su reproducción se corresponde con el ejercicio creado.

C.8. Previsualización de un ejercicio

Descripción: Prueba de la creación de un ejercicio desde la aplicación.

1. Pulsar sobre el botón nuevo
2. Seleccionar el deporte "Fútbol Sala"
3. Seleccionar el tipo "Ejercicio"
4. Escribir el nombre y la información del ejercicio
5. Seleccionar el primer terreno disponible
6. Pulsar sobre crear
7. Arrastrar un jugador a la escena
8. Pulsar sobre el botón "+"
9. Hacer click derecho en el jugador
10. Seleccionar trayectoria zig-zag
11. Pulsar en la pantalla sobre los puntos del esquema
12. Pulsar sobre el botón "play"

Resultado: Satisfactorio, el ejercicio se previsualiza en la pantalla de edición.

C.9. Ficheros inexistentes

Descripción: Prueba sobre la sustitución automática de ficheros de imágenes en ausencia de alguno de ellos.

1. Borrar elementos existentes en un vídeo
2. Pasos hasta reproducir el vídeo

Resultado: El vídeo se reproduce con el sustituto por defecto del elemento borrado.

Apéndice D

Evaluaciones por parte de los usuarios

Las pruebas de usuario fueron realizadas a 5 usuarios distintos entre el 9 y el 18 de Mayo de 2010.

A continuación se presenta una breve descripción de cada uno de los usuarios, así como sus conocimientos técnicos, deportivos y posibles utilidades para ellos de dicha aplicación.

1. Usuario jugador de fútbol sala y fútbol 11, conocimientos informáticos de nivel usuario.

   El usuario utilizaría la apliciación para sus actividades deportivas.

2. Usuario no relacionado con ningún deporte, con conocimientos informáticos de nivel usuario.

3. Usuario jugador de paintball, con conocimientos informáticos nivel usuario básico.

   El usuario utilizaría la aplicación para el apoyo táctico de su equipo.

4. Usuario jugador y árbitro de fútbol sala, con conocimientos avazados de informática.

   Utilizaría la aplicación para su equipo.

5. Usuario no relacionado con ningún deporte, con conocimientos muy avanzados de informática.

Las conclusiones principales que se esperaban obtener de estos tests son: valores para una serie de parámetros de la percepción del vídeo, y evaluar cómo realiza el usuario una serie de tareas críticas, y si es capaz de realizar ciertas acciones. Las

conclusiones esperadas no son acerca de la usabilidad general de la aplicación para afectar drásticamente a la confección de la misma y a su interfaz gráfica de usuario.

Las conclusiones extraidas a partir del estudio realizado a los usuarios sobre el **realismo y utilidad del vídeo** son las siguientes:

	CONCURRENTE	
	ELEMENTOS	TIEMPO
ANIM 1	8	2
	8	4
	8	6
	8	8
ANIM 2	5	2
	5	4
	5	6
	5	8
ANIM 3	3	2
	3	4
	3	6
	3	8
ANIM 4	1	2
	1	4
	1	6
	1	8

USUARIO					
1	2	3	4	5	Media
2	2	2	1	2	1,8
2	3	3	2	3	2,6
2	3	3	2	3	2,6
2	2	3	2	3	2,4
2	2	2	2	2	2
3	3	3	2	3	2,8
3	3	3	2	3	2,8
3	3	3	2	3	2,8
2	3	3	3	3	2,8
3	4	4	4	4	3,8
4	4	4	4	4	4
3	4	4	3	4	3,6
2	2	3	2	3	2,4
3	4	4	4	4	3,8
4	4	4	4	3	3,8
3	3	3	3	3	3

	SOLAPADO	
	ELEMENTOS	TIEMPO
ANIM 1	8	2
	8	4
	8	6
	8	8
ANIM 2	5	2
	5	4
	5	6
	5	8
ANIM 3	3	2
	3	4
	3	6
	3	8
ANIM 4	1	2
	1	4
	1	6
	1	8

USUARIO					
1	2	3	4	5	Media
3	3	3	3	4	3,2
3	4	3	3	3	3,2
3	3	3	3	3	3
2	3	2	2	3	2,4
4	4	3	3	4	3,6
4	4	3	3	4	3,6
3	3	3	3	4	3,2
3	3	3	3	3	3
3	3	4	3	4	3,4
4	4	3	4	4	3,8
4	4	4	3	4	3,8
3	3	4	3	4	3,4

Figure D.1: Resultados de los tests

A partir de los resultados obtenidos en las simulaciones con 1, 3, 5 y 8 elementos, para configuraciones concurrentes y solapadas de duraciones 2, 4 6 y 8 segundos, se puede deducir que:

- Cuanto mayor es el número de elementos, mayor es la confusión y la dificultad para entender lo representado, si a eso se añade un movimiento concurrente, los usuarios determinaron que el vídeo no conseguia explicar lo que pretendía.
- En términos generales, los movimientos de 8 segundos se hacen largos para el usuario, en este caso, representando trayectorias relativamente simples, pero de longitud considerable.

- En casi todas las situaciones los movimientos en modo solapado con tiempos largos obtienen peores puntiaciones
- Los rangos con mejor puntuación son los que se encuentran en el centro de las muestras alrededor de 4 elementos y alrededor de 5 segundos.
- La mejor puntuación promedio es para 3 elementos y 6 segundos de duración en un movimiento concurrente.
- Los resultados son muy igualados y muy dispersos, y el estudio es completamente dependiendo del ejercicio escogido.

A partir de estos hechos obtenidos a partir de los números se deduce que:

- Optar por una configuración fija sería correcto para muchos casos, pero sería errónea para muchos otros
- Estos resultados nos sirven para obtener un valor aproximado de cuales podrían ser los valores por defecto de la aplicación a la hora de tratar el tipo de movimiento y la duración del mismo.

Finalmente las decisiones tomadas a partir de los resultados del test, y que se implementarán en la aplicación son:

- Se establecerán los valores por defecto a movimiento solapado y el tiempo a 5 segundos.
- Se permitirá el cambio de estos parámetros para las escenas de la animación.
- Se permitirá que se pueda variar de tipo de escena y duración para todas las escenas sin relación entre ellas en estos parámetros(concurrente-solapada-concurrente).

Análisis de los resultados obtenidos de los **tests generales** de la aplicación:

1. Instalación de la Aplicación a partir del pen-drive suministrado.

   Ninguno de los usuarios tuvo problemas para instalar el programa(con el framework Adobe AIR previamente instalado).

   La instalación de dicho framework es una descarga de la página oficial de adobe, similar a cuando alguien instala Acrobat reader ó el reproductor de Flash.

2. Ejecución independiente del grabador de audio Java

   Esta acción no supuso problemas para los usuarios previa explicación y/o lectura de la parte correspondiente del manual de usuario.

3. Creación de un ejercicio básico con dos escenas.

   Tras la lectura del manual o una pequeña demo parecida al vídeo que se incluirá con la aplicación ningún usuario tuvo problemas graves para realizar la tarea propuesta.

   El tiempo de diferencia entre los distintos usuarios fue bastante grande, sobre todo en usuarios con distintos conocimientos informáticos, con un timpo total de entre 1 y 5 minutos para la primera ejecución de la aplicación por parte del usuario.

4. Exportación del ejercicio creado.

   No supuso problema la exportación del ejercicio en ninguno de los formatos por parte del usuario.

   Se descubrión un bug que se produción con el modo de utilización del los usuarios "avanzados", por el cual estos usuarios al guardar el fichero, en el nombre escribían la extensión del mismo, y el resultado producía un fichero con doble extensión(bug solucionado).

5. Reproducción del fichero exportado en formato interno(.piz)

   Al contrario de lo que pudeira parecer, esta fue una de las actividades que más problemas produjo en el usuario, ya que algunos de los usuarios con conocimientos básicos de informática no sabían como hacerlo, ya que buscaban una opción abrir dentro del propio programa.

6. Sincronización de audio con en la edición de un ejercicio.

   Previa lectura o explicación del funcionamiento del editor, no se produjeron incidencias destacadas en esta actividad y los usuarios fueron capaces de llevarla a cabo sin problemas

7. Edición de una escena compleja

   Esta parte fau otra de las que más problemas conllevó, y hasta que los usuarios no se habituaron a la utilización de la línea del tiempo como medio para desplazarse por las distintas escenas no se pudo llevar a cabo por todos los usuarios

8. Importación de un nuevo ejercicio con los ficheros e instrucciones proporcionados.

   Con un pequeño anexo explicando la importación de nuevos deportes, ningún problema para ningún usuario.

Apéndice E

Ejemplo Documento Exportable

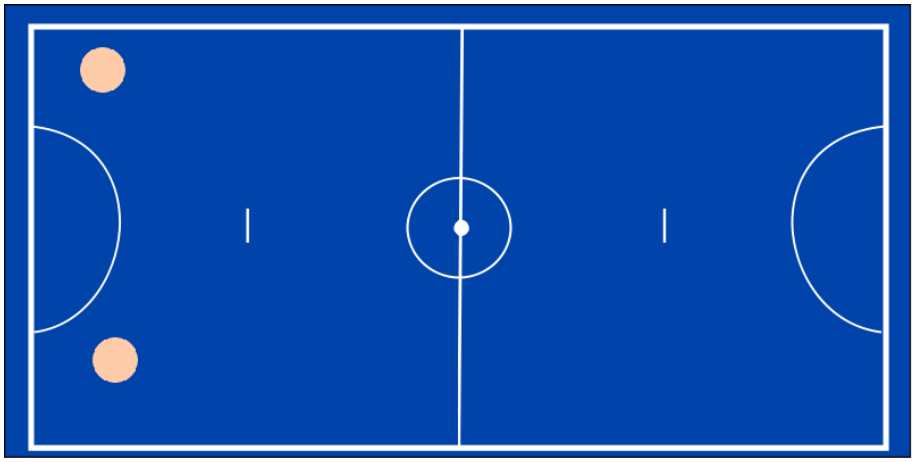
Pizarra Digital

nombre del ejercicio

Fecha Creación	6 / 5 / 2012
Fecha Modificación	6 / 5 / 2012
Fecha Actual	Sat Jun 9 10:12:55 GMT+0200 2012

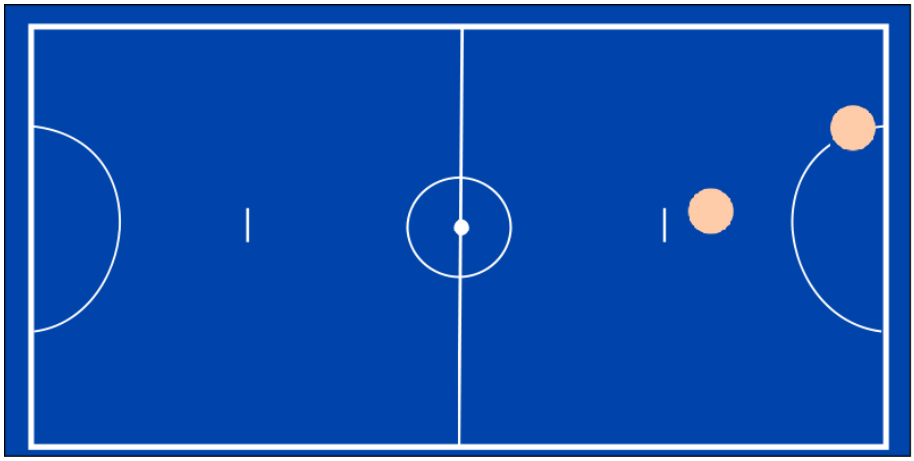
Índice

nombre del ejercicio



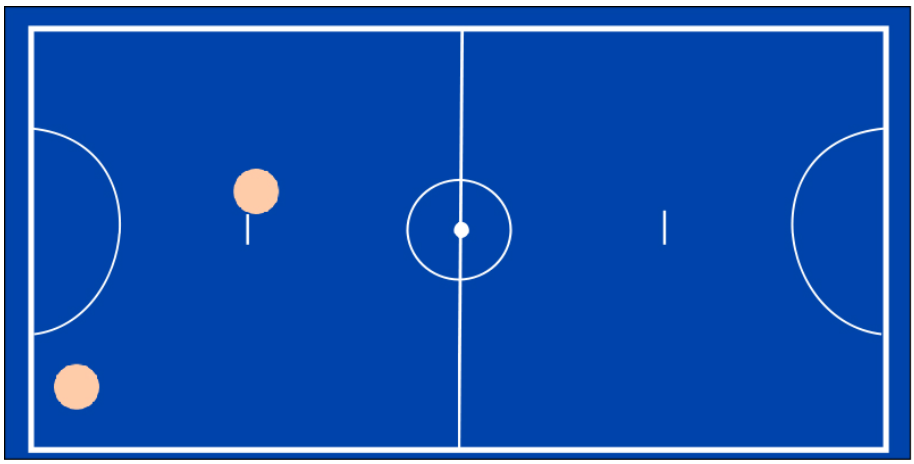
Texto de la escena 1

Duración: 5 segundos Tipo: Serie



Texto de la escena 2

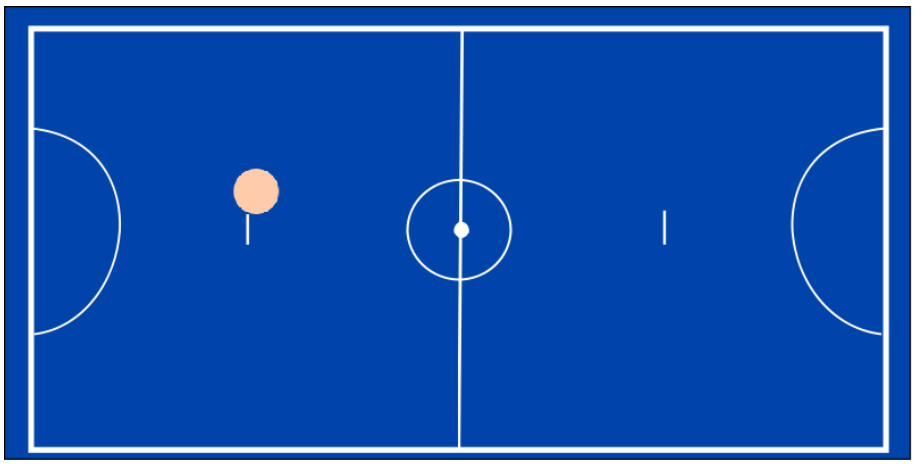
Duración: 5 segundos Tipo: Serie



Texto de la escena 3

Duración: 5 segundos Tipo: Serie

nombre del ejercicio



Texto de la escena 4

Duración: 5 segundos Tipo: Serie

Bibliografía

[1] LaTeX - http://www.latex-project.org/. Última consulta (09/06/2012) 7.1

[2] MiKTeX - http://miktex.org/. Última consulta (09/06/2012) 7.1

[3] LyX - http://www.lyx.org/. Última consulta (09/06/2012) 7.1

[4] pdf - http://es.wikipedia.org/wiki/PDF. Última consulta (09/06/2012) 1.2

[5] Easy Animation http://www.es.easy-animation.com/. Última consulta (09/06/2012) 2.2.1

[6] Jes Soft -http://www.jes-soft.com/. Última consulta (09/06/2012) 2.2.2

[7] Skaut Notebook- http://www.skaut.es/es/Productos-de-software/skaut-notebook. Última consulta (09/06/2012) 2.2

[8] Dartfish - http://www.dartfish.com/en/software/dartfish-teampro/index.htm. Última consulta (09/06/2012) 2.2

[9] Sport Office - http://www.sportcode.co.rs/. Última consulta (09/06/2012) 2.2

[10] flv - http://es.wikipedia.org/wiki/Flash_Video. Última consulta (09/06/2012) 2.3, 4.1

[11] SprotCode - http://www.sportcode.co.rs/. Última consulta (09/06/2012)

[12] HP IT Project Risk Management he542s a.01 5.2.2

[13] NuSoap - http://sourceforge.net/projects/nusoap/. Última consulta (09/06/2012) 2.3

[14] Flash - http://www.adobe.com/es/products/flashplayer.html. Última consulta (09/06/2012) 2.4, 2.4.2

[15] Flex - http://www.adobe.com/es/products/flex.html. Última consulta (09/06/2012) 2.4

[16] Java - http://www.java.com/es/. Última consulta (09/06/2012) 2.4

[17] Air - http://www.adobe.com/es/products/air.html. Última consulta (09/06/2012) 3.1, B.1

[18] Modelo Multicriterio - http://www.ccee.edu.uy/ensenian/catmetad/material/MdA-Scoring-AHP.pdf. Última consulta (09/06/2012)

[19] R.I.A. - http://es.wikipedia.org/wiki/Rich_Internet_Applications. Última consulta (09/06/2012) 2.4.1

[20] NetBeans - http://netbeans.org. Última consulta (09/06/2012) 2.4.2

[21] Microsoft Word - http://office.microsoft.com/es-es/word/FX100487983082.aspx. Última consulta (09/06/2012) 5.4

[22] Microsoft Excel - http://office.microsoft.com/es-es/excel/fx100487623082.aspx. Última consulta: 29/05/2010

[23] Dia Diagrammer - http://dia-installer.de/index_en.html. Última consulta (09/06/2012) 5.4

[24] Inkscape - http://www.inkscape.org/?lang=es. Última consulta (09/06/2012) 5.4

[25] Gantt Project - http://www.ganttproject.biz/. Última consulta (09/06/2012) 5.4

[26] XAMPP - http://www.apachefriends.org/es/xampp.html. Última consulta (09/06/2012) 2.4.2

[27] phpMyAdmin - http://www.phpmyadmin.net/home_page/index.php. Última consulta (09/06/2012) 2.4.2

[28] SQLite Browser - http://sqlitebrowser.sourceforge.net/. Última consulta (09/06/2012) 2.4.2, 3.1, B.1

[29] SQLite - http://www.sqlite.org/. Última consulta (09/06/2012)

[30] Wireframes - http://en.wikipedia.org/wiki/Website_wireframe/. Última consulta (09/06/2012) 3.2.1, B.3.1

[31] Modelo-Vista-Controlador - http://es.wikipedia.org/wiki/Modelo_Vista_Controlador. Última consulta (09/06/2012) 3.2.1, B.3.1

[32] Observer - http://www.sqlite.org/. Última consulta (09/06/2012) 3.2.1, B.3.1

[33] Notepad++ - http://notepad-plus.sourceforge.net/es/site.htm. Última consulta (09/06/2012) 2.4.2

[34] Adobe Flex - http://www.adobe.com/es/products/flex/?promoid=BPBJI. Última consulta (09/06/2012) 2.4.2

[35] Maaash - http://maaash.jp/. Última consulta (09/06/2012) 4.1

[36] W3C - http://www.w3c.es/. Última consulta (09/06/2012) 6.1

[37] Adobe Kuler - http://kuler.adobe.com. Última consulta (09/06/2012) 3.5.2, B.6.2

[38] Desarrollo en cascada - http://es.wikipedia.org/wiki/Desarrollo_en_cascada. Última consulta (09/06/2012) 5.1

[39] N-Tier - http://www.webopedia.com/quick_ref/app.arch.asp. Última consulta (09/06/2012)

3.6, B.7