UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

DESARROLLO DE UNA APLICACIÓN PARA FACILITAR EL DISEÑO DE VIDEOJUEGOS EDUCATIVOS

Autor: Álvaro Rodríguez Ruiz

Tutor: Mario Rafael Ruiz Vargas

Julio, 2011

DEDICATORIA

Le dedico este proyecto fin de carrera a:

Tamara, sin la que no habría encontrado la manera de finalizar la carrera de esta forma.

Mi tutor del proyecto, Mario Rafael, por estar siempre disponible cuando lo necesitaba.

Meli, por todos los buenos momentos.

Y por último pero más que a nadie a mis padres, sin los que nada de esto hubiera sido posible.

TRIBUNAL DEL PROYECTO FIN DE CARRERA

HOJA PARA EL TRIBUNAL QUE CORRIGE EL PROYECTO FIN DE CARRERA	
Título	Desarrollo de una aplicación para facilitar el diseño de videojuegos educativos
Autor	Álvaro Rodríguez Ruiz
Tutor	Mario Rafael Ruiz Vargas
El tribunal	
Presidente	Telmo Zarraonandia
Vocal	Sergio Pastrana
Secretario	Juan Manuel Tirado
Realizado el acto de defensa y lectura del pfc el día 27 de Julio del 2011 en la Escuela Politécnica Superior de la UC3M, se acuerda otorgarle la calificación de:	

Espacio de firma		
Vocal	Secretario	Presidente

TABLA 1- HOJA PARA EL TRIBUNAL QUE CORRIGE EL PROYECTO FIN DE CARRERA

ÍNDICE

Tabla de ilustraciones

Índice de Tablas

Definiciones

Concepto	Definición
Aplicación standalone	Aplicación que no depende de una conexión a red para su correcto funcionamiento.
Bug	Es el resultado de un fallo o deficiencia durante el proceso de creación de software.
Drag & Drop	Drag and drop (Arrastrar y soltar) es una expresión informática que se refiere a la acción de mover con el ratón objetos de una ventana a otra o entre partes de una misma ventana
Feedback	Feedback o retroalimentación es el proceso en el cual un programa recibe una señal por parte del usuario y cambia su estado de alguna forma para hacerle saber al usuario que ha recibido su señal y la está procesando o la ha procesado.
Product backlog	Lista de historias de usuario en la que se especifica su prioridad, su estimación con una desviación tanto pesimista como optimista y el estado en el que se encuentra dicha historia de usuario.
Programación por parejas	La programación por parejas consiste en que dos personas programan en la misma computadora. Mientras uno va escribiendo el código, el otro está constantemente fijando en qué escribe el otro, corrigiéndole de posibles errores y pensando en mejores alternativas.
Refactorizar	Técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.
Spike solution	En la metodología adaptativa XP, una "spike solution" es un breve programa en el que se pondrán a prueba escenarios que se van a dar en el sistema final, que nos ayudará a realizar estimaciones de historias de usuario (y por tanto de tareas) con más confianza. A modo de metáfora, podríamos ver esto como si el sistema fuese excavar un gran túnel y la "spike solution" sea usar un taladro que llegue hasta donde queremos llegar con el túnel. Esto no nos servirá como túnel, pero sí sabremos cómo de profundo es y por tanto podremos estimar recursos mejor que antes.

Acrónimos

Acrónimo	Significado
RIA	**R**ich **I**nternet **A**pplication
VoIP	**V**oice **o**ver **IP**

1. INTRODUCCIÓN

En este proyecto fin de carrera se ha querido realizar una aplicación que ayude a educadores a diseñar videojuegos educativos con el propósito de que mediante dicho videojuego se pueda instruir o enseñar más fácilmente al alumno, ya que ellos podrán jugar con ellos y así aprovechar las ventajas de los videojuegos para facilitar su aprendizaje. De esta manera se integraría el videojuego como un recurso más para la enseñanza.

Pese a que el educador pueda tener una idea aproximada de cómo quiere que sea su videojuego, diseñarlo para su posterior implementación es una tarea muy complicada. Realizar un diseño de este tipo requeriría los conocimientos y la capacidad de abstracción de un ingeniero informático, por lo que probablemente los educadores que van a usar la aplicación no vayan a estar preparados para ello.

Es por ello que la aplicación resultante de este proyecto fin de carrera debe guiar y orientar al educador sobre cómo debe completar la concepción de su diseño.

Para ello, la aplicación es capaz de estructurar la información que compone un videojuego, de manera que el educador perciba toda la información que se puede definir en un videojuego de la manera más estructurada posible y la tarea de realizar el diseño sea lo más rápida y sencilla posible manteniendo la condición de que pueda indicar cualquier requisito para el videojuego que deseen.

A partir de esta modelización, la aplicación tiene varios módulos en los que se le presentará al educador lo que se quiere conseguir con cada uno, por lo que la aplicación también proporcionar la ayuda necesaria al educador para que pueda especificar los requisitos que el desee en cada módulo.

La aplicación también permite abrir y guardar diseños de manera que se pueda consultar el juego diseñado o realizar modificaciones en él.

Además, la aplicación también genera un informe con la información introducida por el educador. Esta información está lo más estructurada posible, de manera que a partir de este informe el equipo de desarrollo entienda fácilmente lo que el educado quiso diseñar y de estar manera minimizar o incluso evitar realizar reuniones para aclarar dicho diseño.

Por último, pese a que el propósito principal de la aplicación fuera facilitar el diseño de videojuegos educativos realizados por educadores, debe ser capaz de permitir el diseño de otro tipo de juegos, de manera que la aplicación se pueda utilizar en otro contexto sin tener que ser modificada. Sin embargo, no se espera que con esta aplicación se permitan realizar diseños de videojuegos muy complejos, como por ejemplo aquellos que requieran mundos virtuales con efectos tridimensionales, ya que su realización hubiera sido demasiado complicada.

1.1 Planteamiento del problema

El planteamiento del problema para este proyecto fin de carrera fue el siguiente:

- Realizar un estudio del arte, para tomar consciencia de las aplicaciones o estudios que se hayan realizado y de esta manera no repetir el trabajo que ya está realizado o caer en errores que ya se han cometido.

- Valorar y estudiar cuál sería la tecnología más adecuada para el proyecto.

- Valorar y estudiar cuál sería la metodología que más garantías aportara de que el proyecto iba a cumplir con las expectativas en el tiempo planificado.

- Desarrollar la aplicación, siguiendo los pasos que la metodología seleccionada marcase y documentando todo el proceso en esta memoria. Durante el desarrollo, realizar todas las reuniones posibles con el tutor de este proyecto fin de carrera para verificar que el desarrollo estuviera llevando el rumbo correcto y corregir las posibles dudas que se tuvieran.

- Realizar una evaluación de la aplicación para comprobar que cumple con todos los requisitos establecidos. Se debió seleccionar el método más adecuado para la evaluación de esta aplicación. Una vez seleccionado, se debió realizar dicha evaluación para que si no fuera satisfactoria, se debiera volver hacia atrás para realizar los cambios necesarios para que se cumplieran dichos requisitos.

1.2 Objetivos

El **objetivo general** de este proyecto fin de carrera fue:

- Implementar una aplicación que permitiera facilitar a educadores el diseño de videojuegos educativos. Esta debería permitir diseñar videojuegos con los que dichos educadores consigan que los jugadores aumenten el nivel de aprendizaje respecto a otro método tradicional de educación.

Mientras que los **objetivos específicos** fueron:

1. Facilitar el diseño de otro tipo de videojuegos, cuyo propósito no sea el aprendizaje del jugador sino otro, como el entretenimiento del mismo, por ejemplo. Pese a que el objetivo principal era el de diseñar videojuegos educativos, la aplicación debía ser capaz de diseñar otro tipo de videojuegos, si bien no se esperaba que con ella se pudieran diseñar videojuegos muy complejos con detallados mundos tridimensionales, por ejemplo.

2. Proporcionar una funcionalidad de generar informes con toda la información que el educador hubiera introducido y lo más clara y estructurada posible, para que el equipo de desarrollo pudiera saber perfectamente cómo implementar el videojuego a partir de dicho informe. Estos informes crean un lenguaje común entre el educador que realiza el diseño y el equipo de desarrollo, de manera que se limitan las ambigüedades y las partes del diseño que no estén claras.

3. Proporcionar ayuda constantemente al educador mientras realice el diseño, para que sepa en todo momento la información que puede introducir. Evitar que la ayuda agobie al educador, ya que podría resultar molesto y complicaría el diseño del videojuego.

4. Realizar una aplicación lo suficientemente modulable y flexible para que resulte muy sencillo realizar mejoras futuras a dicha aplicación.

5. Realizar una evaluación de la aplicación. De esta forma podremos asegurarnos de que esta cumple totalmente con los requisitos planteados inicialmente.

1.3 Estructura del documento

El documento tiene la siguiente estructura, a partir de este apartado:

- **Planificación:** En este apartado se detalla la planificación realizada al principio del proyecto, estructurando la misma en fases con un propósito individual para cada fase.

- **Estudio previo:** En él se explican los estudios realizados sobre los videojuegos educativos y las aplicaciones similares a la que se quiso conseguir en este proyecto fin de carrera. Además, se valoran las posibles tecnologías que se pudieron usar para implementar la aplicación y se elije una de ellas, de manera justificada.

- **Estudio de metodologías:** En este apartado se exponen los tipos de metodologías existentes y sus filosofías. Se realiza un análisis y una comparación de cuál es la más adecuada para este tipo de proyecto y se elije una de ellas de manera justificada.

- **Normativa de código:** En él se explica cuál es la normativa de código que se siguió para implementar la aplicación.

- **Metáfora del sistema:** En este apartado se explica la metáfora del sistema. Este apartado es un paso de la metodología seleccionada y buscaba definir el sistema con una metáfora de manera que sea más sencillo diseñar e implementar el mismo.

- **Spike solution:** En este apartado se explica cómo se diseñó un boceto de lo que iba a ser la aplicación final. Este es también un paso de la metodología seleccionada y buscaba elaborar un pequeño ejemplo con el que se tomara el contacto con la tecnología seleccionada y ayudara a estimar la duración del mismo y los posibles errores o problemas que pudieran surgir a lo largo del proyecto.

- **Implementación de la solución:** En este apartado se explica la modelización de los videojuegos elaborada, las consideraciones iniciales que se tomaron para implementar la aplicación, la estructura de la misma en paquetes y módulos y por último se explica cómo se implementó cada módulo de la aplicación.

- **Evaluación:** En este apartado se explica cuál fue el método de evaluación escogido y se justifica su elección. Después se explica cómo se realizó dicha evaluación.

- **Conclusiones:** En las conclusiones se explican los beneficios que proporciona la aplicación implementada, la modularidad y flexibilidad del mismo, el trabajo futuro y el coste que ha supuesto la realización de este proyecto.

- **Anexos:** En este apartado se incluye cualquier tipo de información adicional, como la información relacionada con la metodología seguida y la estructura de archivos del CD adjuntado con este proyecto.

- **Bibliografía y referencias:** En este último apartado se detalla toda la bibliografía y las referencias utilizadas para elaborar este proyecto fin de carrera.

1.4 Planificación

Dado que el proyecto iba a tener unas dimensiones considerablemente grandes, se realizó una planificación a priori, con el fin de cumplir con el plazo de entrega y la calidad exigida. Las conversaciones con el tutor comenzaron el día **22 de Septiembre de 2010**, por lo que tomaremos esa fecha como la inicial.

La planificación a seguir fue la siguiente:

1. Realizar un estudio previo de las tecnologías existentes relacionadas con la temática de este proyecto, seleccionando aquella que se considere como la más adecuada para el proyecto. Fecha prevista de entrega al tutor: **29 de Octubre de 2010**.

2. Realizar las metodologías existentes para la realización de proyectos. De todas las metodologías que se estudien, seleccionar una, con la que se realizará el proyecto: **5 de Noviembre de 2010**.

3. Realizar una versión de demostración de la aplicación, que sirva para habituarse a la tecnología elegida y además ayude a acotar los requisitos establecidos para este proyecto: **22 de Diciembre de 2010.**

4. Realizar la implementación de la totalidad de la aplicación, realizando las pruebas que sean oportunas. El objetivo de esta fase es tener una aplicación standalone operativa. Fecha prevista de entrega: **25 de Mayo de 2011**.

5. Realizar la presentación que se deberá exponer ante el jurado. Fecha prevista de entrega al tutor: **11 de Julio de 2011**.

En todas las fases están implícitas reuniones con el tutor que fueron imprescindibles para el desarrollo del proyecto.

2. ESTUDIO PREVIO

Antes de comenzar a realizar el proyecto fue necesario realizar un estado del arte para tomar consciencia de las aplicaciones o estudios que se hubieran realizado y de esta manera no repetir el trabajo que ya estaba realizado o caer en errores que ya se habían cometido.

Una vez realizado el estudio del arte, era necesario un estudio previo de las tecnologías utilizadas para la elaboración de este tipo de aplicaciones para escoger la o las tecnologías con las que se desarrollaría el proyecto. Esta fase era por tanto crucial, ya que una mala selección de una tecnología se arrastraría a lo largo del proyecto. En concreto, se estudiaron las siguientes tecnologías:

- Javascript
- PHP
- Flash CS5
- Actionscript
- MXML
- Adobe Flex Builder
- Java

2.1 Estado del arte

Este estudio del estado del arte nos guió tanto a la hora de confeccionar una lista de funcionalidades que debía tener nuestra aplicación como a la hora de evitar desarrollar funcionalidades que no eran necesarias. Era muy importante realizar un correcto estudio del arte, ya que este afectaría en la siguiente fase del proyecto que consistía en seleccionar la tecnología, la cual era crucial para el devenir del proyecto.

Para este proyecto, se comprobó el estado del arte de los estudios realizados sobre juegos educativos y de aplicaciones de creación de diagramas, ya que se esperaba que en la aplicación de este proyecto el educador pudiera crear diagramas que le facilitasen la tarea de diseñar el juego educativo.

2.1.1. Estudios realizados sobre juegos educativos

Esta parte era muy importante, ya que no se podía realizar una aplicación que permitiese diseñar juegos educativos sin comprender primero estos: cuáles deben ser sus características, qué ventajas proporcionan al aprendizaje, etc.

En la actualidad, hay ya muchos estudios realizados [1][2][3] en los que se habla acerca de los juegos educativos. Una consideración crucial que se debe tener en cuenta en este tipo de juegos es que *"si conseguimos estar continuamente motivando a nuestro jugador, éste exprimirá todas las facetas del juego, consiguiendo mejorar la experiencia del jugador"* [3]. Es por tanto necesario tener en cuenta o incluso jugar con las sensaciones y emociones del jugador para que este encuentre la motivación de seguir jugando.

Sin embargo, no solo es necesario que un jugador tenga la motivación suficiente para querer seguir jugando, sino que este debe estar *"inmerso dentro de la dinámica del juego, lo que puede provocar que asimile mejor todo lo que el videojuego muestra y, por lo tanto, la experiencia del juego sea mayor"* [3].

En ese momento en el que el jugador está inmerso en el juego, es cuando "fluye" en él, según Csikszentmihalyi. En este momento el jugador estará totalmente absorto y concentrado en el videojuego.

Para conseguir esto, se debe conseguir un equilibrio entre la dificultad y las habilidades del jugador. Una complejidad demasiado elevada supondrá al jugador una frustración y ansiedad que no van a permitir el aprendizaje del jugador de la manera óptima. En el lado opuesto, retos asequibles o repetitivos no van a suponer otra cosa que hastío o aburrimiento al jugador, con lo que la motivación del mismo bajará considerablemente y el aprendizaje cesará [3].

Este compromiso entre los retos y las habilidades de los jugadores se conoce como el "flujo", según Csikszentmihalyi. En este equilibrio, el jugador encontrará el nivel de aprendizaje óptimo con el juego, ya que al estar plenamente concentrado en el juego el aprendizaje es automático e inherente, adquiriendo así los conocimientos en menos tiempo de lo habitual.

Este estado de "flujo" se presenta cuando se le muestra al jugador un reto que pone sus capacidades al máximo, lo cual genera cierta sensación de ansiedad. Una vez el jugador utiliza sus habilidades para dominar la situación esta ansiedad deja sitio a este estado de fluidez [3].

Esta teoría de Csikszentmihalyi se denomina la **Teoría de Flujo** y se resume en la ilustración "Teoría de flujo":

ILUSTRACIÓN 1 - TEORÍA DE FLUJO

Mediante esta teoría de flujo se puede observar que para que el juego educativo cumpla su cometido debe comprometer las habilidades del jugador sin llegar a un nivel demasiado alto, pero evitando en todo momento retos que sean fáciles para él ya que de esta manera se perderá el interés del jugador y por tanto no se conseguirán los objetivos perseguidos, ya que el nivel de aprendizaje obtenido será menor del esperado. Una vez se encuentre este equilibrio entre dificultad y habilidades, el nivel de aprendizaje es máximo y el jugador podrá aprender de una manera casi inapreciable para él, en contraposición a un método más clásico de educación.

Esto revela que la aplicación que se quería implementar en este proyecto fin de carrera debía facilitar al educador el diseño de videojuegos educativos que comprometieran al máximo las capacidades de los alumnos y que fuera él quien encuentre el equilibrio en el que no se generase demasiada ansiedad a ellos.

Si la aplicación no permite diseñar juegos complicados, el educador solo podrá diseñar juegos sencillos que resultarán aburridos a los alumnos, por lo que estos perderán el interés hacia ellos y no se podrá realizar ningún aprendizaje.

2.1.2. Aplicaciones de creación de diagramas

En la actualidad hay numerosos editores de diagramas de múltiples propósitos: diagramas de flujo, diagramas de secuencia, diagramas de clases o cualquier tipo de diagramas ad-hoc, como los que se pretende editar en este proyecto. Por ello se seleccionaron algunos de los editores que se consideraron más populares: ArgoUML [4], Gliffy [5] y Microsoft Visio [6].

No se estudió en detalle una comparación entre las aplicaciones o cómo o con qué símbolos resuelven sus diagramas, porque los diagramas que se pretendía resolver son ad-hoc al problema y por tanto no va a haber ninguna relación entre ellos. Lo que se pretendía mirar es cómo se resuelven las interacciones básicas con el usuario: cómo se pone un componente, cómo se relacionan varios componentes, si se pueden redimensionar y en ese caso cómo se lleva a cabo, etcétera.

Las funcionalidades básicas son las siguientes:

Funcionalidad
Poner un componente en el "lienzo"
Mover un componente mediante "Drag & Drop"
Enlazar componentes
Redimensionar componentes
Poner texto en un componente

TABLA 2- FUNCIONALIDADES BÁSICAS DE APLICACIONES DE DISEÑO DE DIAGRAMAS

Y esta es la manera en la que se realizan estas acciones en las aplicaciones anteriormente explicadas:

- Poner un componente en el "lienzo". En Visio y Gliffy haciendo Drag & Drop desde el botón hasta dicho "lienzo", mientras que en ArgoUML hay que pulsar primero en el botón y luego en el "lienzo". Para el proyecto se va a utilizar una solución alternativa: con pulsar el botón ya aparece el componente en el "lienzo". De esta manera el "feedback" es inmediato y no supone ninguna interactuación adicional del usuario con el ratón, ya que simplemente debe poner el componente donde desea.

- En los tres editores para mover un componente se hace Drag & Drop. Es una interacción muy típica e intuitiva, por lo que no parece que tenga sentido utilizar otra forma de hacerlo.

- Para enlazar componentes, los tres editores siguen el mismo procedimiento: pinchar en el botón correspondiente al conector y hacer Drag & Drop desde uno de los "anclas" del componente origen hasta uno de los "anclas" del componente destino. Entre el "Drag" y el "Drop" se puede ver de manera temporal como aparece el conector entre el componente origen y donde está el ratón. En esta aplicación se va a seguir esta solución con una salvedad: en lugar de hacer Drag & Drop la conexión se realizará con dos "clicks". Entre un "click" y otro se mostrará análogamente el conector temporal entre el componente origen y donde esté el ratón. De esta forma no se pierde "feedback" y es más cómodo para el usuario.

- En los tres editores para redimensionar un componente se hace Drag & Drop desde uno de los anclas de dicho componente. De nuevo no hay ninguna razón aparente para realizarlo de otra forma, así que así se hará en la aplicación.

- Los tres editores realizan de manera muy similar la acción de poner texto en un componente: se hace un "click" en el centro del componente para acto seguido escribir el texto. También se piensa que se podría hacer mediante "doble-click", pero se ha decidido finalmente usar el mecanismo de un único "click" tal y como funcionan la mayoría.

- Hay algunas funcionalidades adicionales como cambiar de color un componente, copiar y pegar un componente mediante los atajos Ctrl+C y Ctrl+V o la existencia de una regla delimitando los bordes del "lienzo". Como este proyecto está enfocado para un usuario que no tiene muchos conocimientos sobre informática ni sobre la notación en la que se realizarán los diagramas la aplicación debe ser lo más sencilla y concisa posible: una funcionalidad superflua solo puede conseguir desviar la atención por parte del usuario.

2.2 Javascript

Javascript es un lenguaje de **scripting orientado a objetos débilmente tipado**. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de páginas dinámicas. Al estar integrado en el navegador web, el código **Javascript se ejecuta en el lado del cliente**. Uno de los aspectos más atractivos de Javascript es que solo se necesite un navegador compatible para poder ejecutar código Javascript. Y la gran mayoría de navegadores del mercado aceptan código Javascript. [7]

Javascript surgió, como otros lenguajes, para extender las capacidades de HTML: hacer a la página dinámica. Y este es su principal uso: tanto en un fichero externo, especificando su ruta en la página HTML como embebido directamente entre el código HTML.

Sin embargo JavaScript también se puede usar fuera de los navegadores Web, como por ejemplo en widgets, gadgets, ficheros PDF o como lenguaje de scripting en numerosos framework [8].

Para que este lenguaje pudiera ser un candidato válido para ser escogido, debe cumplir dos cosas: que tuviera una parte gráfica para poder dibujar correctamente los componentes necesarios en 2-D y que se pudieran capturar los eventos creados por el usuario.

Respecto a la parte gráfica hay varias soluciones: desde librerías como jsDraw2D [9] hasta recientemente mediante el uso de canvas en HTML5 [10].

En lo que a los manejadores de eventos respecta, hay varias formas de hacerlo usando JavaScript [11]. De la forma antigua empleando atributos HTML como por ejemplo "onclick" o de la nueva forma independiente de HTML mediante la función "addEventListener" de la especificación DOM. De esta última forma, JavaScript da soporte a numerosos eventos: mouseover, mouseout, mousedown, mouseup, click, doubleclick, etcétera [12]. Sin embargo, esta última capacidad de JavaScript no está soportada por la mayoría de navegadores o al menos lo más populares.

En este momento del proyecto aún no se sabía si la aplicación será online o de escritorio aún y pese a que como hemos visto hay aplicaciones de escritorio para JavaScript su uso no está muy expandido y parecía que había otras opciones mejores que proporcionarn ayudas al desarrollador como entornos de desarrollo.

Como hemos visto ni en la opción standalone ni en la aplicación de escritorio JavaScript nos proporcionaban la suficiente confianza de que fuera la mejor de las tecnologías posibles, por lo que quedó descartado su uso para la realización del proyecto. Si finalmente se hubiera elaborado una solución online, quizás hubiera que apoyarse en el uso de JavaScript pero de cualquier forma no iba a participar en la lógica de la aplicación.

2.3 PHP

PHP es un lenguaje de **scripting orientado a objetos sin tipado** que fue originalmente diseñado para, al igual que Javascript, desarrollar páginas web dinámicas.

La principal diferencia con Javascript es que mientras este se ejecuta en el lado del cliente, **PHP se ejecuta en el lado del servidor**. Para ejecutar un programa PHP necesitamos un servidor web con módulo procesador PHP (Apache Tomcat, por ejemplo), que será el que envíe la página web procesada en HTML al cliente, por lo que no debería haber ningún problema de compatibilidad con los navegadores web como sí podía pasar con JavaScript, sobre todo con elementos gráficos y manejadores de eventos. Además al tratarse el código en el lado del servidor hay una mayor seguridad sobre código malicioso de la que hay usando JavaScript. No obstante, para el propósito que quiere alcanzar este proyecto, la seguridad no es una de las propiedades.

En este aspecto de gráficos y eventos, pese a que hay varias librerías de PHP que realicen esta parte gráfica (muchas orientadas a la creación de gráficas), el soporte del manejo de eventos en PHP no es el adecuado: hay algunas implementaciones realizadas [13], pero si se requieren implementaciones para realizarlo significa que el propio lenguaje nativo no está preparado para esta funcionalidad y por tanto no va a ser tan fácil encontrar documentación en el momento de que nos encontremos con algún problema.

Por tanto se descartó la opción de utilizar PHP porque se pensó que podía haber mejores opciones en el mercado.

2.4 ActionScript

ActionScript es un **lenguaje de script orientado a objetos fuertemente tipado**, que se usa especialmente en animaciones Web realizadas con **Adobe Flash**. Es un dialecto de ECMAScript al igual que JavaScript, de ahí que ActionScript tenga muchas similitudes con dicho lenguaje (como por ejemplo el tipado).

La principal diferencia entre ambos lenguajes es que Actionscript está orientado a la creación de animaciones y la integración de archivos de audio y vídeo. Ese fue el propósito de su creación y en esa línea ha seguido su desarrollo a lo largo de sus versiones (la actual es la 3.0).

Con Actionscript 3.0, la estructura del código generalmente es la siguiente:

- Un documento Flash donde se va a generar el entorno.
- Una o varias clases en Actionscript en las que se determina la lógica de los objetos que participarán en el entorno gráfico.

De esta manera, podemos cambiar fácilmente el comportamiento (cambiando una de las clases) o la apariencia (cambiando el documento Flash). Usando esta estructuración encapsulada del código, la aplicación es mucho más modificable y escalable.

Si bien en este proyecto no iban a ser necesarias animaciones muy elaboradas, la parte gráfica es esencial, por lo que ActionScript debería ser sin duda alguna un lenguaje a considerar.

No obstante vamos a ver con más detalle ActionScript en las dos herramientas de Adobe en las que se utiliza: Adobe Flash Professional CS5 y Adobe Flash Builder 4.

2.5 Adobe Flash Professional CS5

Adobe Flash Professional CS5 es una **aplicación de escritorio** diseñada por Adobe para elaborar **aplicaciones con animaciones en Flash**, ya sea para aportar dinamicidad a una página web o para elaborar una animación que se vaya a reproducir independientemente por un reproductor Flash.

Esta aplicación está en un nivel de madurez muy alto, por lo que no resulta difícil explotar las capacidades de ActionScript con esta aplicación. Sin embargo, su principal propósito es el de elaborar complicadas animaciones que como hemos dicho en el apartado anterior, a priori no van a ser necesarias.

La aplicación que se pretendía elaborar necesita un soporte de interfaces gráficas para el que Adobe Flash Professional CS5 no está diseñado, por lo que probablemente sea la aplicación menos relacionada con el proyecto de las que habíamos visto hasta el momento. Por tanto se descartó su uso para este proyecto.

2.6 MXML

MXML (*Macromedia eXtensible Markup Language*) **es un lenguaje descriptivo basado en XML** (usa unos namespace definidos por Adobe) que describe interfaces de usuario, crea modelos de datos y tiene acceso a los recursos del servidor. Se usa generalmente en combinación con ActionScript para desarrollar aplicaciones RIA (*Rich Internet Application*).

MXML se usa principalmente para la declaración de interfaces de aplicaciones, pero puede usarse también para implementar la lógica de negocio y comportamiento de aplicaciones distribuidas. Puede contener trozos de código ActionScript en su interior.

Fue desarrollado inicialmente por Macromedia para la plataforma FLEX de Adobe. Como vamos a ver esta plataforma en el apartado "Adobe Flash Builder 4", veremos cómo es un fichero MXML con más detalle en dicho apartado.

2.7 Adobe Flash Builder 4

Al igual que Flash CS5, **Adobe Flash Builder 4** es una **aplicación de escritorio** diseñada por Adobe que implementa el SDK de Adobe Flex. Esta aplicación permite a los desarrolladores de aplicaciones web construir rápida y fácilmente RIA's (*Rich Internet Applications*) .

La principal característica de estas aplicaciones es que se permite la interacción del usuario **sin que este tenga que refrescar la página** ya que desde el principio se carga toda la aplicación, y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una Base de Datos o de otros ficheros externos [14].

Para conseguir esto, Adobe Flex Builder usa MXML para mostar la interfaz de usuario y Actionscript para definir la lógica de la aplicación.

Al igual que con JavaScript y PHP, se examinó la aplicación respecto a dos ámbitos: la capacidad gráfica y la de manejar eventos. Respecto a la parte gráfica ActionScript está precisamente diseñado con este fin, mientras que la lista de eventos que se proporciona es muy grande [15], dando soporte a numerosos eventos de ratón y teclado, que a priori son los que va a necesitar el proyecto. Además la documentación en ambas partes parecía generosa o al menos mayor que en JavaScript y PHP. Esto era algo muy importante, ya que la capacidad de resolver problemas mediante el estudio de la documentación existente ayuda a garantizar un progreso constante del proyecto.

Además, Adobe flash Builder 4 es un entorno de desarrollo basado en Eclipse, con el que sin duda nos sentiremos más cómodos a la hora de implementar la lógica de la aplicación.

Como no parece que haya ninguna funcionalidad que no se pueda implementar en Flex y sí en las otras alternativas y la combinación de la capacidad gráfica con la facilidad de crear interfaces de usuario que proporciona Flex cumple todas las necesidades requeridas en el proyecto, era la principal candidata para desarrollar el proyecto.

2.8 Java

Java es un **lenguaje de programación orientado a objetos y fuertemente tipado**. Abstrae al programador de la gestión de la memoria, lo que le ha hecho muy popular respecto a C y C++. Es una tecnología muy madura, que posee una amplia documentación y entornos de programación como Eclipse o NetBeans que ayudan al programador a la hora de implementar un programa. Además Java es teóricamente portable de manera total (en la práctica la portabilidad no es total, pero casi) ya que el código en Java es compilado a un bytecode general antes de ser compilado a código máquina.

Pese a todas las posibilidades que proporciona Java, no era la tecnología idónea para este proyecto, ya que la parte gráfica de Java se implementa mediante la librería Swing que es demasiado compleja e incómoda de utilizar.

Otra alternativa hubiera ssido utilizar la familia JavaFX, que es la alternativa de Sun para desarrollar RIAs. Esta tecnología tiene dos desventajas importantes frente a Adoble Flex: el código generado es mucho más pesado (en términos de memoria de almacenaje) y que JavaFX, por ser una tecnología aún temprana, no tiene el soporte que sí tiene Adobe Flex en la web con los navegadores [16].

Esto hace que directamente no se tuviera en cuenta esta opción a favor del SDK de Adobe Flex.

2.9 Selección de las tecnologías

Una vez estudiadas todas las tecnologías, se realizó una tabla comparativa para poder escoger una entre todas ellas. Para elegir la teconología, se valoró positivamente que tuvieran estas características:

- **Creación de gráficos:** que permitiera la creación de elementos gráficos con los que se reprensentarían los elementos y formas de los diagramas.
- **Gestión de interfaces de usuario:** que permitiera crear aplicaciones que posean interfaces de usuario.
- **Gestión de eventos:** que tuviera gestión de eventos para poder capturar las acciones que quiere realizar el usuario sobre los elementos y formas de los diagramas.

- **Herramientas de apoyo:** que tuviera suficientes herramientas de apoyo para que la realización del proyecto sea lo más sencilla posible, tales como un entorno de desarrollo o documentación oficial.
- **Standalone:** que tuviera la posibilidad de implementar aplicaciones es escritorio.
- **Online:** que tuviera la posibilidad de implementar aplicaciones a las que se pueda acceder online.
- **Madurez:** que fuera lo suficientemente madura como para que se pueda confiar en que no contenga errores y que lleve el suficiente tiempo en el mercado como para que le haya dado a la gente a explotar todo su potencial.

	JavaScript	PHP	Adoble Flash Professional CS5	Adobe Flash Builder 4	JavaFX
Creación de gráficos	Sí	Regular	Sí	Sí	Sí
Gestión de interfaces de usuario	Sí	Sí	No	Sí	Sí
Gestión de eventos	No	Regular	Sí	Sí	Sí
Herramientas de apoyo	No	No	Sí	Sí	Sí
Standalone	Regular	Regular	Sí	Sí	Sí
Online	Sí	Sí	Sí	Sí	Regular
Madurez	Sí	Sí	Sí	Sí	No

TABLA 3 – COMPARACIÓN ENTRE LAS DIFERENTES TECNOLOGÍAS

Por tanto, ya que cumple solventemente con todos los requisitos, proporciona herramientas de apoyo como el entorno de desarrollo (por tanto facilitará el desarrollo del proyecto), la madurez y documentación de la misma, la tecnología elegida fue: **Adobe Flash Builder 4**

3. ESTUDIO DE METODOLOGÍAS

Para elaborar un buen proyecto es necesario seguir una metodología. En este apartado se van a exponer las metodologías existentes para el desarrollo de sistemas de información y se explicará cuál se eligió, razonando los motivos de por qué se eligió sobre las demás.

3.1 Metodologías predictivas

Las metodologías predictivas fueron la primera solución a los problemas derivados de los primeros programas: se codificaba código directamente sin realizar ningún tipo de análisis ni planificación, lo que derivaba en mucha inconformidad por parte del cliente: no obtenía lo que él exactamente quería, además de otro tipo de problemas, como la imposibilidad de saber el estado del proyecto en un momento dado o estimar los recursos necesarios para realizarlo.

Estas metodologías se denominan predictivas porque se lleva a cabo desde el principio una **gran planificación** que incluye la totalidad del proyecto, con unas **fases muy bien delimitadas que se deben realizar de manera secuencial**. Esta planificación se sigue de manera rigurosa, con la predictibilidad y la eficiencia como metas.

En estas metodologías, primero se debe elaborar una planificación del proyecto, donde se valorarán los recursos que se requieren y se establecerá un calendario con la realización de todas las fases del proyecto.

Después, se deben valorar las alternativas tecnológicas que se van a usar para elaborar el proyecto y se debe redactar una lista de los requisitos de usuario que van a contener toda la funcionalidad que el cliente espera de la aplicación.

Una vez hecho esto, se comienza la **fase de análisis** en la que se elabora una lista de los requisitos de software provenientes de los requisitos de usuario, se realizan los modelos de datos, los diagramas de clases (en caso de que el sistema siga el paradigma de orientación a objetos), de secuencia, los casos de uso, etcétera.

Tras el análisis, se comienza la **fase de diseño**. Esta fase define totalmente la arquitectura del sistema, del entorno tecnológico que le va a dar soporte y la especificación detallada de todos los componentes que compondrán el sistema. El **objetivo del diseño es que a partir de él, cualquier programador pueda generar el código**, como si este se generase de forma pseudo-automática (de hecho ya hay investigaciones para generar código automáticamente a partir del diseño). El programador en sí no es importante, ya que este no puede plantear alternativas o mejoras a la solución extraída del diseño. Es por esto que se dice que "las metodologías predictivas están orientadas al proceso" [17].

Para realizar todas estas fases, es necesaria una gran cantidad de documentación.

Llegados a este punto, **cualquier pequeño cambio en la funcionalidad requiere mucho trabajo**, ya que se debe volver al principio del desarrollo del proyecto, cambiar aquellos requisitos de usuario afectados y todo aquello que derive de estos requisitos de usuario (requisitos de software, modelos de datos, diagramas de clases, de secuencia, de casos de uso,…). Por tanto estas metodologías se "resisten al cambio" [18].

La metodología predictiva más conocida en España es **Métrica V3**, la metodología promovida por el Ministerio de Administraciones Públicas del Gobierno de España. Métrica V3 se divide en estas

fases [19], que como se puede observar están muy relacionadas con las fases comentadas anteriormente y que estarán representados por un documento individual:

- Planificación de Sistemas de Información (PSI).
- Estudio de Viabilidad del Sistema (EVS).
- Análisis del Sistema de Información (ASI).
- Diseño del Sistema de Información (DSI).
- Construcción del Sistema de Información (CSI).
- Implantación y Aceptación del Sistema (IAS).
- Mantenimiento de Sistemas de Información (MSI).

La **ventaja** de esta metodología es que **es muy eficiente si el diseño es perfecto y no hay que realizar cambios**.

La **desventaja** principalmente es que **comprobar que un diseño está bien hecho es muy complejo**. En otras disciplinas de la ingeniería el diseño puede ser comprobado mediante la física o las matemáticas. En la informática es **prácticamente imposible utilizar medidas empíricas para valorar un diseño** [20]. No es hasta la implementación cuando nos damos cuenta de los errores cometidos en el diseño. Y llegados a ese momento los cambios requieren mucho tiempo para llevarse a cabo con este tipo de metodologías.

Por tanto, numerosos proyectos que han utilizado estas metodologías han resultado fallidos y hoy en día hay bastante desconfianza en este tipo de metodologías.

Este tipo de metodologías **son las adecuadas cuando se va a desarrollar un software destinado a elemento tangible**, como puede ser el airbag de un coche o un componente de un avión, ya que la entrada y la salida del sistema son muy estables y están fijas de antemano.

Cuando el éxito del proyecto depende de la satisfacción del cliente seguramente debamos pensar en otro tipo de metodologías, ya que la probabilidad de que sucedan cambios es demasiado alta como para que el desarrollo del proyecto sea eficiente. **Por tanto, para la realización de este proyecto, se desestimó el uso de este tipo de metodologías**.

3.2 Metodologías adaptativas

Como una alternativa a las metodologías predictivas, surgen las adaptativas o ágiles. Estas presentan varias diferencias: **el cambio es lo habitual** en el desarrollo del proyecto, así que "el cambio es bienvenido" [21] en el desarrollo del proyecto porque ya no suponen un grave retraso y además nos acercan al producto que el cliente desea realmente.

Además, en las metodologías predictivas el código podría implementarlo cualquier persona ya como se explicó anteriormente el proceso de elaborar código es prácticamente una mera traducción: del diseño al código. **En las adaptables el papel de las personas a la hora de generar código es imprescindible**. Es necesaria la retroalimentación de los mismos para que el proyecto sea satisfactorio. Por eso se dice que **estas metodologías son "orientadas a la gente"** [22] y no al proceso, como las predictivas.

Mientras en las metodologías el cliente no ve un software ejecutable que pueda ver y opinar hasta una fase muy avanzada del proyecto, **en este tipo de metodologías es principal tener demostraciones de la aplicación desde el principio**: esto nos ayudará a saber qué quiere realmente el cliente. Si el cliente desea realizar modificaciones se cambiarán los requisitos. Como esto no sucede al final del proyecto como sí sucedía en las metodologías predictivas, estos cambios apenas suponen un retraso.

La última de las principales diferencias es que mientras en las metodologías predictivas se realizaba una planificación que alcanzaba la totalidad del proyecto, en las adaptables entra el concepto de **iteración**. **En las adaptables también se establece una rigurosa planificación, pero esta solo alcanza una iteración, que es una fracción del proyecto.** El tiempo que dura cada iteración se determina para cada tipo de metodología adaptable o ágil, de las cuales explicaré algunas breves características más adelante.

La manera de planificar en las metodologías predictivas se muestra en las ilustraciones "Planificación en metodologías predictivas" y "Planificación en metodologías predictivas (II)":

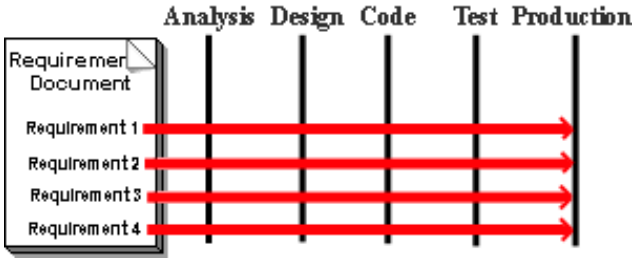

ILUSTRACIÓN 2 – PLANIFICACIÓN EN METODOLOGÍAS PREDICTIVAS

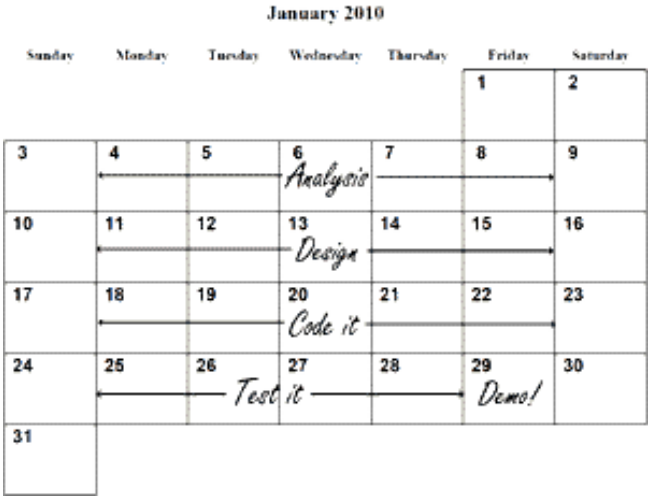

ILUSTRACIÓN 3 - PLANIFICACIÓN EN METODOLOGÍAS PREDICTIVAS (II)

Podemos ver como los requisitos se mantienen constantes a la largo de la planificación, en la cual se sabe de antemano que el diseño comenzará cuando acabe el análisis y la codificación cuando acabe el diseño. Mientras tanto **en metodologías adaptativas las características de la aplicación van a estar compuestas por historias de usuario**, cada una de las cuales atraviese todas las fases, de manera que hay una demostración de cada una que se puede entregar prematuramente al cliente para que lo pruebe.

Con esto se consigue que el cliente vea posibles fallos en la implementación que habrá que corregir (por lo que aumentará la satisfacción del cliente) y tengamos más información sobre lo que quiere el cliente, de manera que lo próximo que se implemente tenga más probabilidades de ser lo que el cliente verdaderamente desea.

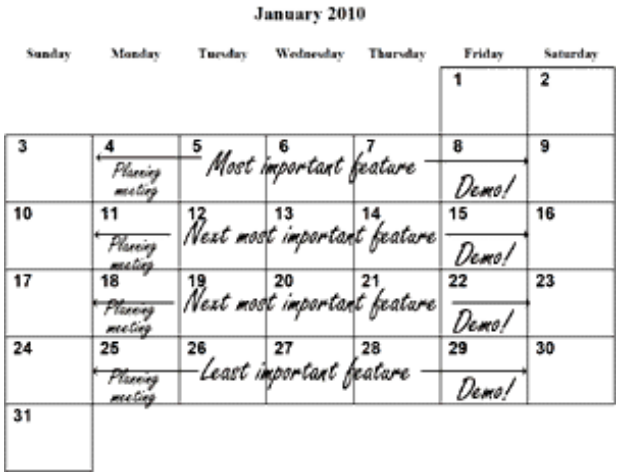

ILUSTRACIÓN 4 – PLANIFICACIÓN EN METODOLOGÍAS ADAPTATIVAS

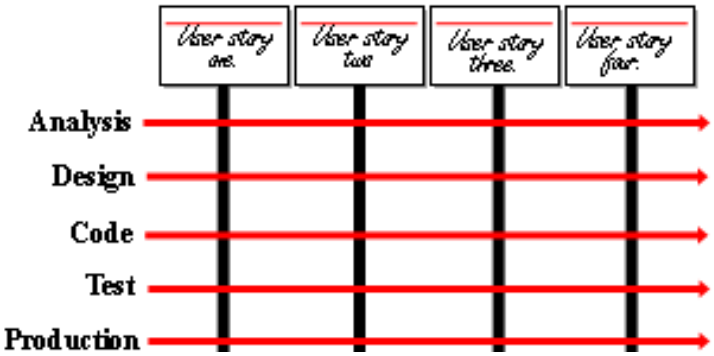

ILUSTRACIÓN 5 - PLANIFICACIÓN EN METODOLOGÍAS ADAPTATIVAS (II)

En estas metodologías, se detecta el riesgo antes, ya que atravesamos todas las fases en cada iteración.

Las historias de usuario tienen el mismo propósito que los casos de uso pero no son lo mismo. Se usan para crear estimaciones de tiempo para el comunicado de la reunión de planificación y sustituyen a los largos documentos de requisitos de usuario.

Desde el punto de vista del cliente, las historias de usuario son lo que se necesita hacer para ellos. Son como los escenarios de uso, excepto por el hecho de que no se limitan a describir interfaces de usuario.

Se deben escribir en no más de tres líneas en una terminología que el cliente conozca.

Un error muy común es confundir el nivel de detalle de los requisitos con el de las historias de usuario. Las historias de usuario solo deberían proveer suficiente detalle como para realizar una estimación razonable de cuánto se va a tardar esta estimación en implementarse [23].

A la hora de elaborar historias de usuario, se deberían evitar detalles como la algoritimia o la disposición de la base de datos. Las historias se deben concentrar en las necesidades y beneficios del usuario en lugar de especificar diseños de la interfaz de usuario.

Estas diferencias, además de suponer un número considerablemente menor de los documentos que encontrábamos en las metodologías predictivas, suponen una adaptabilidad al cambio muchísimo mayor. Y por tanto, mayor facilidad para satisfacer las necesidades del cliente.

Todas estas características quedan resumidas en doce principios [24] enumerados en el "Manifiesto Ágil" [25] redactado en 2001 por varios de los representantes de las metodologías ágiles del momento [26].

A partir de este manifiesto y estos principios que se pueden consultar en el apartado "Bibliografía y referencias", el procedimiento generalmente en estas metodologías es el siguiente:

1. Se generan las historias de usuario del proyecto (no olvidemos que podrán estar sujetas a cambios).
2. Se le da una prioridad a cada historia de usuario.
3. Se estima el tiempo que llevará realizar estas historias.
4. Se determina el alcance de la iteración.
5. Se detallan las historias de usuario de esa iteración.
6. Se especifican pruebas para la historia de usuario.
7. Se identifican las tareas de cada historia de usuario y su responsable.
8. Se re-estiman las historias de usuario si fuera necesario.
9. Se realizan las tareas de la iteración y al terminar se vuelve al paso 4.

Hay varias metodologías adaptables, pero como todas guardan los principios básicos de las metodologías adaptables enumerados anteriormente, vamos a ver ahora las diferencias entre tres de las metodologías adaptativas más populares: XP, SCRUM y Crystal.

3.2.1. Diferencias entre metodologías adaptativas

De igual forma que con la elección de la tecnología a usar, se realizó una tabla comparativa entre las metodologías adaptativas más populares:

	XP	SCRUM	Crystal
Enfoque	En proyectos que posean requisitos vagos o con alta probabilidad de cambio.	Énfasis en la gestión del proyecto.	Tiene varias metodologías en función del número de personas y el nivel de formalismo.
Definir visión o historia a utilizar	Crear una metáfora del sistema.	Se requiere una descripción general del sistema. Se dice que se puede crear como la caja de un juego.	-
Redacción de historias	El cliente debería redactar las historias de usuario.	El propietario del producto mantiene un product backlog.	Se realizan workshop(s) para identificar los requisitos de alto nivel. Se componen en un documento (Documento de Requisitos / Business Area

			Definition) que contiene los casos de uso y los requisitos no funcionales.
Quién hace las estimaciones iniciales	Desarrolladores.	El propietario del producto.	Jefe de proyecto y los desarrolladores.
Duración de las iteraciones	Entre 1 y 3 semanas.	30 días.	Lo decide el Jefe de proyecto y los desarrolladores.
Diseño	Simple.	Simple.	Mucho más formal.
Codificación	Normativa de código obligatoria. Pruebas antes de la codificación. Programación en parejas.	-	Normativa de código opcional.
Seguimiento	Reunión diaria. Diagramas grandes visibles.	Reunión diaria. Gráfico del backlog.	Diseminadores de información.

TABLA 4 - DIFERENCIAS ENTRE METODOLOGÍAS ADAPTATIVAS

3.3 Elección de la metodología

Como vimos en el apartado de las metodologías predictivas, **no se iban a usar este tipo de metodologías para el proyecto, ya que con total seguridad iban a suceder multitud de cambios a lo largo del mismo**. Además, el objetivo del sistema no era un producto tangible como una parte de un avión con unas necesidades claras y específicas, sino que el objetivo de este proyecto fin de carrera era satisfacer las necesidades del proyecto "Modeling educational experience for training on emergency situations for children" del departamento DEI de la Universidad Carlos III de Madrid [27] en el que se contextualiza este proyecto fin de carrera. Por tanto la metodología elegida era una metodología adaptativa.

De las tres posibles metodologías **se seleccionó la metodología XP**, porque parecía la más "liviana" de las tres: por el empleo del diseño simple frente al diseño formal de Crystal y porque el énfasis del seguimiento que se realiza en SCRUM carecía de sentido en este proyecto ya que el equipo de trabajo se componía por un solo miembro, el cual debe saber cómo va el proyecto sin la necesidad de realizar informes o gráficas del backlog.

3.4 Especificación de la metodología a seguir

El aspecto principal de XP son sus simples reglas. Estas reglas de trabajo se representan mediante el siguiente diagrama de flujo de la ilustración "Diagrama de flujo de XP" [28]:

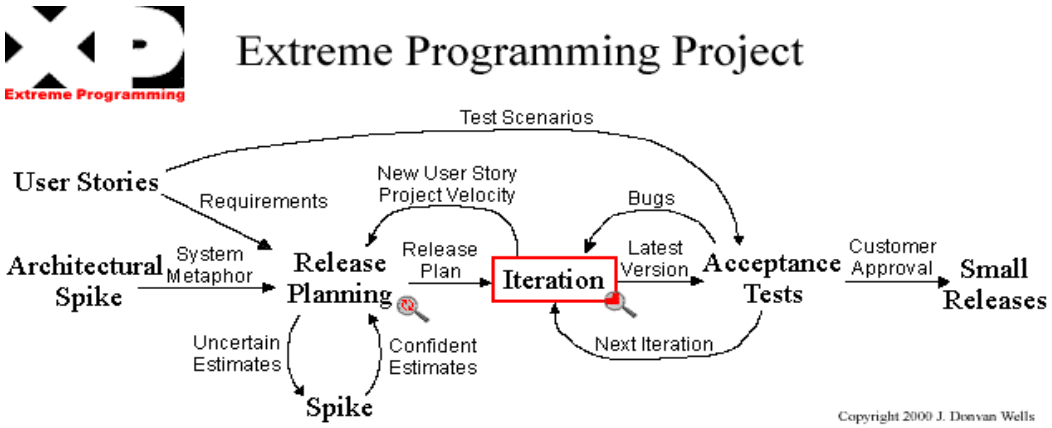

ILUSTRACIÓN 6 - DIAGRAMA DE FLUJO DE XP

Los pasos a seguir son:

1. Si la empresa no tiene una normativa de código o no se trabaja para ninguna empresa, crearla.

2. Elegir una metáfora del sistema (por ejemplo una línea de producción o una lista de materiales) y crear una "spike solution" para averiguar respuestas a problemas de diseño. Construirlo solo para abordar el problema en cuestión y evitar otros temas. El objetivo es aumentar la fiabilidad de la estimación de historias de usuario.

3. Se elaboran las historias de usuario y se estima cuánto tiempo va a suponer implementarlas, **entre 1 y 3 semanas, que es el tiempo que establece XP como iteración**. Si una historia ocupa menos de una semana es que se ha utilizado demasiado nivel de detalle y si ocupa más de tres es que debemos dividirla en varias.

4. Se especifica al menos una prueba por cada historia de usuario para comprobar que se ha realizado correctamente.

5. Se realiza una reunión para la planificación de la que se produce **la publicación del plan**. Este plan especifica qué historia de usuario va a ser implementado para cada versión del sistema y las fechas de esos lanzamientos.

6. Para cada iteración, se le da un conjunto de historias al **cliente**, el cual **indicará el orden en el que se elaborarán las historias de usuario** durante la reunión de la planificación de la iteración. Hay que tener en cuenta que en la iteración no solo se elaborarán las historias de usuario, si no que se realizarán las pruebas de aceptación para cada historia de usuario. Además, se deben volver a comprobar aquellas historias de usuario de la anterior iteración cuyas pruebas fallaron. En este paso obtendremos las tareas a realizar.

7. Las historias de usuario y pruebas falladas se desglosan en tareas.

8. Se elaboran fichas para cada tarea a realizar.

9. Estas fichas, a diferencia de las historias de usuario, sí deben estar escritas en un lenguaje del diseñador.

10. Después, quién vaya a realizar esa tarea estima cuánto le va a suponer realizar sus tareas. El tiempo ideal para realizar una tarea es de 1 a 3 días. Si se requiere menos, se debería agrupar y si se requiere más, se debería dividir en otras tareas.

11. Se comprueba que el tiempo total que se necesitará en realizar las tareas no exceda el tiempo planificado para finalizar la iteración. Si se excede, el cliente debe elegir historias de usuario que se realizarán en posteriores iteraciones. Si hay margen para incluir más historias de usuario, el cliente podrá añadir la que desee (*"New User Story, Project Velocity"*).

12. Se elabora el **diseño simple** de las tareas. Diseño simple es aquel diseño que maximiza la cantidad de trabajo que hay que hacer y que no se ha hecho y se elaboran las fichas CRC (Clase, Responsabilidades y Colaboraciones) para cada elemento que aparezca en el diseño. XP prohíbe añadir funcionalidad no planificada previamente. También recomienda refactorizar todo lo posible.

13. Se **codifican** las tareas. XP dice que hay que seguir la programación por parejas. Es obligatorio **elaborar las pruebas unitarias primero**, que representarán a las pruebas de aceptación especificadas previamente y seguir la normativa de código. También se aconseja que cada programador vaya integrando su código periódicamente con el de los demás, para evitar usar fragmentos de código de otra persona obsoletos.

14. Una vez que se ha terminado de diseñarlo se ejecutan las pruebas. Cuando se encuentra un bug en el código hay que elaborar una prueba de aceptación para solventar este problema.

15. Si aún quedan historias de usuario para elaborar, se vuelve al paso 5.

4. NORMATIVA DE CÓDIGO

En este apartado se define la normativa de código con la que se realizó este proyecto. El motivo de Implementar el código siguiendo una normativa es aumentar la consistencia y legibilidad de dicho código.

Se decidió seguir la normativa que propone Adobe para Flex [29].

5. METÁFORA DEL SISTEMA

La metáfora del sistema fue la de una pizarra en la que un educador escribe un diagrama. Como es una pizarra, el profesor podrá borrar o modificar elementos del diagrama en la pizarra, ya sea para moverlos de sitio o modificar su tamaño.

6. SPIKE SOLUTION

En este apartado se explica la "spike solution" realizada. Como el sistema iba a consistir en su mayor parte en realizar diagramas en los que se tiene que desplegar un componente y unirlo con otros, se realizó una pequeña aplicación en el que un número indeterminado de cuadrados y círculos se puedan poner en el diagrama y puedan relacionarse mediante el uso de conectores.

La primera tarea necesaria será elaborar el diseño. Por simplicidad, se han omitido los getters y setters de los métodos y todos los manejadores de eventos de la clase "ManejadorEventos". El diseño de la "spike solution" es el siguiente:

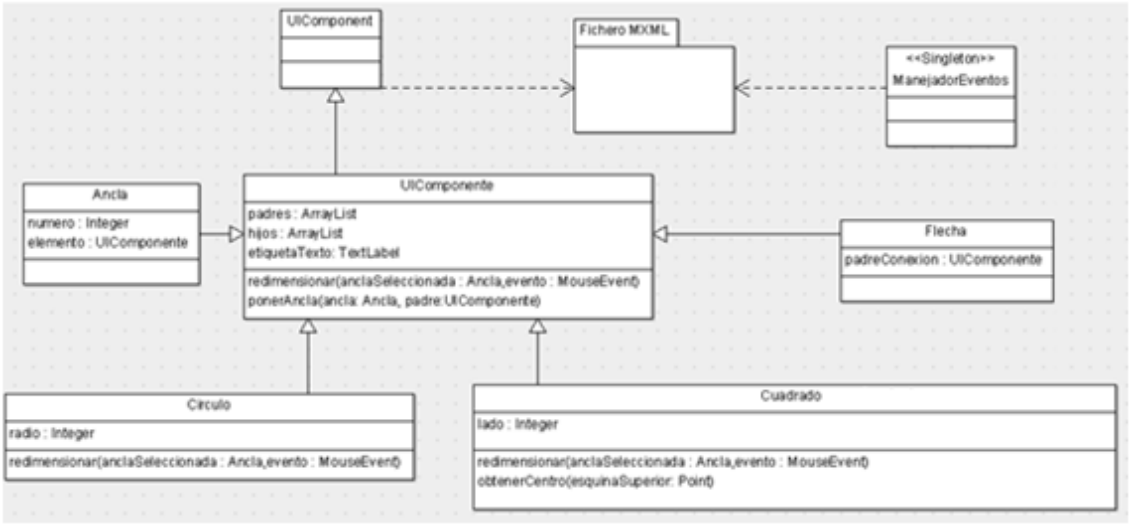

ILUSTRACIÓN 7 - DIAGRAMA DE CLASES DE SPIKE SOLUTION

Sin embargo, antes de codificar la "spike solution" se debían hacer las pruebas tal y como indica la metodología XP.

En este caso, se intentó emplear FlexUnit [30] para elaborar pruebas unitarias que prueben la "spike solution", pero el funcionamiento no es el idóneo a la hora de simular los eventos necesarios para probar el programa. Se pensó en usar un bot [31] que realizasen esta simulación, pero el coste de tiempo en realizar las pruebas con esta solución iba a ser demasiado elevado por lo que se desestimó su uso en este proyecto.

Por tanto se optó por la prueba manual de la aplicación. Sin embargo, para ceñirnos a la metodología lo máximo posible, se redactaron una serie de pruebas a modo de listado previamente a la implementación del código que tanto el autor del proyecto como el tutor debían seguir a la hora de realizar las pruebas manuales, de manera que si los efectos no eran los esperados, no se podría dar por válidas las pruebas y habría que revisar de dónde procedía ese error.

1. Se pulsará el botón "Generar Componentes". Efecto: Aparecerán un cuadrado y un círculo en el punto (0,0) de la escena.
2. Poner el cursor sobre el círculo. Aparecerán las anclas de dicho círculo.
3. Hacer "click" en el círculo y mantener pulsado ("Drag"). Mover el ratón. Efecto: el círculo se mueve al mismo ritmo que el ratón.
4. Dejar de mantener pulsado el botón izquierdo del ratón. Mover el ratón. Efecto: el cursor se ha movido y el círculo se ha quedado parado en el punto donde se dejó de pulsar el ratón.
5. Repetir pasos 2-4 para el cuadrado.
6. Hacer "click" en un ancla del círculo y mantener pulsado ("Drag"). Mover el ratón. Efecto: el círculo se redimensiona acorde al movimiento del ratón.
7. Repetir el paso 6 para el resto de anclas del círculo.
8. Repetir los pasos 6-7 para el cuadrado.
9. Pulsar el botón "Enlazar".
10. Pinchar en un ancla del cuadrado.
11. Pinchar en un ancla del círculo. Efecto: aparece una flecha que va desde el ancla seleccionada del cuadrado al ancla seleccionada del círculo.
12. Repetir los pasos 9-11 tres veces, una para cada ancla restante del cuadrado. Las anclas del círculo pueden ser aleatorias, incluso repetirse. Efecto: aparece tres flechas más que van desde las anclas seleccionadas del cuadrado a las anclas seleccionadas del círculo.
13. Repetir los pasos 10-12 cambiando cuadrado por círculo. Efecto: aparecen cuatro flechas más que van desde las cuatro anclas del círculo hasta las anclas seleccionadas del cuadrado.
14. Mover el cuadrado. Efecto: todas las flechas se adaptan al movimiento del cuadrado. El círculo sigue en la misma posición.
15. Mover el círculo. Efecto: todas las flechas se adaptan al movimiento del círculo. El cuadrado sigue en la misma posición.
16. Redimensionar el cuadrado. Efecto: todas las flechas se adaptan al movimiento de redimensionado del cuadrado. El círculo sigue en la misma posición.
17. Redimensionar el círculo. Efecto: todas las flechas se adaptan al movimiento de redimensionado del círculo. El cuadrado sigue en la misma posición.
18. Crear 10 componentes más pulsando los botones "Poner cuadrado" y "Poner círculo". Mover estos componentes alrededor del primer círculo creado.
19. Realizar los pasos 9-11 entre cada componente y dicho círculo, sin importar las anclas seleccionadas. Efecto: aparecen 10 flechas que van desde los componentes creados al primer círculo creado.
20. Realizar los pasos 9-11 entre dicho círculo y cada componente, sin importar las anclas seleccionadas. Efecto: aparecen 10 flechas que van desde el primer círculo creado a los componentes creados.
21. Mover el primer círculo. Efecto: todas las flechas se adaptan correctamente al movimiento de este círculo.
22. Mover uno de los diez componentes creados. Efecto: las flechas creadas entre ese componente y el primer círculo creado se adaptan al movimiento. El resto de componentes y flechas no se deben ver afectados.

7. IMPLEMENTACIÓN DE LA SOLUCIÓN

7.1 Modelización de los videojuegos diseñada

Como se mencionó en la introducción, para que la aplicación pudiera guiar al educador lo máximo posible, era necesaria que dicha aplicación presentara toda la información que pudiera afectar a un juego de la forma más esquemática posible. Para ello, se elaboró una modelización de los posibles videojuegos que se pueden diseñar con esta aplicación.

Este modelado es fruto de muchas horas de investigación ya que esta modelización era totalmente crítica, ya que el resto de la aplicación se desarrollaba a partir de dicha modelización.

Una incorrecta modelización supondría implementar una aplicación que no cumpliría con los requisitos y por tanto habría que volver hacia atrás en el desarrollo malgastando mucho tiempo. El modelado es el siguiente:

Se ha dividido el juego en dos fases: la ***"Definición de las reglas del juego"*** y la ***"Definición del escenario"***. La primera fase es la de la "Definición de las reglas del juego", que tiene los siguientes módulos que se pueden ver en la ilustración "Diagrama de la definición de las reglas del juego":

ILUSTRACIÓN 8 - DIAGRAMA DE LA DEFINICIÓN DE LAS REGLAS DEL JUEGO

Esta es la información que se trata en cada módulo de la definición de las reglas del juego:

- **LÓGICA**: En este apartado se definen las entidades u objetos que hay en el juego con los que interactúan los jugadores. Se definen los distintos estados que tienen estas entidades, las precondiciones y poscondiciones de cada estado, si hay límites de tiempo en un estado, etc.
- **METAS**: En este apartado se definen las metas que se deben cumplir para superar el juego. Se da la posibilidad de definir metas generales para todos los jugadores o metas más concretas en función del avatar. Además, se permite la definición de umbrales por si al educador le interesa que si un jugador no cumple todas las metas al menos cumpla un 70% de ellas.
- **GUIÓN**: En este apartado se define el guión del juego, como si de una película se tratara.
- **FEEDBACK**: En este apartado se definen aquellos mensajes que se le van a mostrar a los jugadores a lo largo del juego, ya sean mensajes textuales, vídeos, animaciones, etc. Además se pueden definir precondiciones para que se muestren estos mensajes.
- **SOCIALIZACIÓN**: En la socialización se determina cómo se van a poder comunicar los jugadores entre ellos, si el juego va a ser por turnos o en tiempo real y se dará la posibilidad de determinar metas grupales, bien sean competitivas o colaborativas.

- **JUEGOS O RETOS**: En este apartado se da la posibilidad de definir algunos juegos o retos que pueden ir apareciendo a lo largo del juego para que los jugadores los vayan superando, como un cuestionario por ejemplo.
- **PERSISTENCIA**: En el apartado de persistencia se pueden definir los puntos de salvaguarda que habrá en el juego para que los jugadores puedan guardar su partida a lo largo del juego. Además se pueden definir precondiciones que se pueden dar para que un jugador pueda guardar el juego.
- **PREMIOS**: En este apartado se pueden definir los premios que se le irán dando a los jugadores en forma de puntos y las condiciones que se tienen que dar para que los jugadores puedan ganar dichos puntos.

Mientras que los módulos que componen la definición del escenario se pueden ver en la ilustración "Diagrama de la definición del escenario":

ILUSTRACIÓN 9 - DIAGRAMA DE LA DEFINICIÓN DEL ESCENARIO

Esta es la información que se trata en cada módulo de la definición de las reglas del juego:

- **ESCENARIO**: En este apartado se definen las distintas escenas que van a componer el juego y cómo están relacionadas entre ellas. Además, se puede determinar en qué escenas van a estar ubicadas las entidades creadas en la parte de "Lógica".
- **CARACTERIZACIÓN**: En este apartado se van a definir los distintos avatares que se podrán elegir en el juego. Se permitirá dar una definición de la apariencia del mismo, si lo podrán usar los estudiantes o el educador, si será visible por el resto de jugadores, si está automáticamente controlado por ordenador, etc.
- **INTERACCIÓN**: En el apartado de interacción se van a definir los controles que existirán en el juego y los dispositivos con los que se podrá realizar estos controles.
- **CONTEXTO**: En este apartado se puede añadir cualquier información adicional que ayude a contextualizar la situación en la que se va a desarrollar el juego (por ejemplo, un bosque).
- **INTERFAZ**: En este apartado se puede definir la interfaz que verán todos los jugadores del juego.
- **SERVICIOS**: Por último, en el apartado de servicios se pueden determinar todos los servicios que va a proporcionar el juego, como la información a almacenar, la información que podrán compartir los jugadores, si el juego tendrá chat, VoIP, etc.

Todos estos apartados y fases son los que componen la aplicación, como se puede ver la ilustración "Menú lateral de la aplicación":

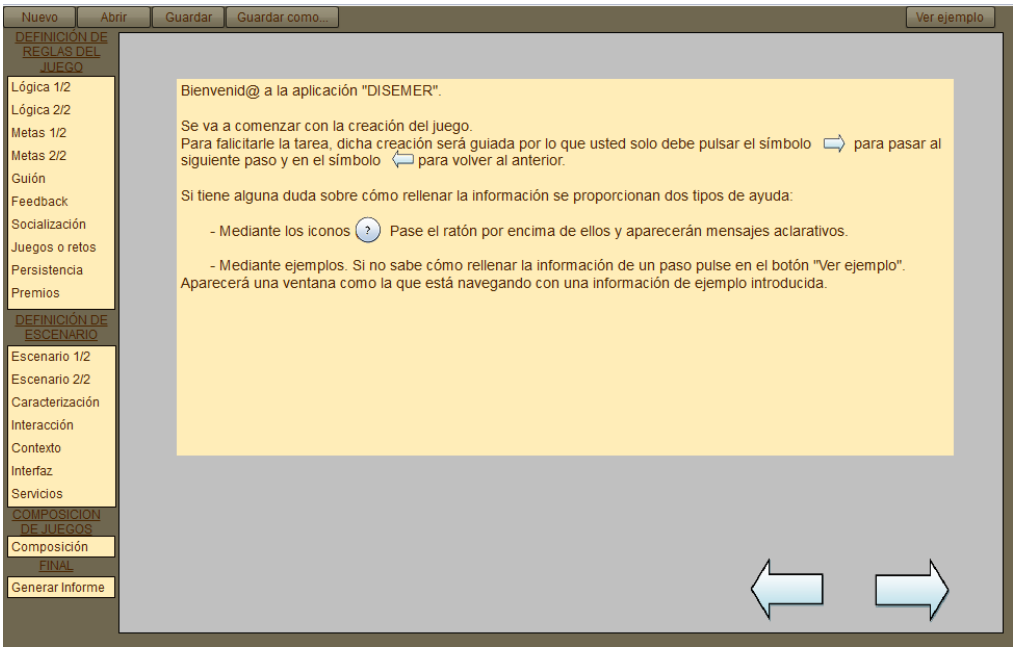

ILUSTRACIÓN 10 - MENÚ LATERAL DE LA APLICACIÓN

Se decidió que la fase de la "Definición de las reglas del juego" fuera antes que la "Definición del escenario" para que a partir de una definición de las reglas del juego se pudieran crear varios juegos con la misma lógica o motor, pero en escenarios distintos. Por ejemplo: un juego de identificar objetos peligrosos en una casa y en un bosque. Las reglas serán las mismas, pero son las escenas las que cambian de un juego a otro.

7.2 Historias de usuario iniciales requeridas

En este apartado se va a elaborar un listado de todas las historias de usuario de las que se va a componer el sistema. Es necesario recordar que, como ya se mostró en el apartado de las metodologías adaptativas, las historias de usuario son aquellas funcionalidades que el usuario quiere que el sistema haga para ellos.

ID	Historia de usuario	Duración (semanas)
HU-01	Elaborar un boceto de la aplicación	1
HU-02	Implementar el ámbito “REWARD” de la aplicación	1
HU-03	Implementar el ámbito “SOCIAL” de la aplicación	1
HU-04	Implementar el ámbito “FEEDBACK” de la aplicación	1
HU-05	Implementar el ámbito “GOALS” de la aplicación	1
HU-06	Implementar el ámbito “LOGIC” de la aplicación	3
HU-07	Implementar el ámbito “STORY-TELLING” de la aplicación	1
HU-08	Implementar el ámbito “DEBRIEFING” de la aplicación	1
HU-09	Implementar el ámbito “PERSISTENCE” de la aplicación	1
HU-10	Implementar el ámbito “SERVICES” de la aplicación	1
HU-11	Implementar el ámbito “INTERACTION” de la aplicación	1
HU-12	Implementar el ámbito “CHARACTERIZATION” de la aplicación	1
HU-13	Implementar el ámbito “SCENARIO” de la aplicación	2
HU-14	Implementar el ámbito “CONTEXT” de la aplicación	1
HU-15	Realización e interpretación de pruebas con usuarios finales	1

TABLA 5 - HISTORIAS DE USUARIO

Como se puede ver, no hay historias de usuario con el fin de generar informes de los juegos que hayan diseñado los educadors o con el fin de almacenar persistentemente estos diseños para poder abrirlos posteriormente y cambiarlos.

Lo que se decidió es implementar estas dos funcionalidades de manera transversal: en cada historia de usuario hay una tarea que resolverán estas dos funcionalidades para su ámbito.

Esta es una manera más "ágil" de hacerlo, ya que desde las primeras demostraciones que se le proporcionen al cliente este va a poder guardar y abrir posteriormente los archivos que vaya editando, además de generar informes de los mismos. Esto nos proporcionaban información de los gustos del cliente sobre ambas funcionalidades en punto prematuro del proyecto, previniendo así posibles fallos y retrasos.

Es necesario aclarar que estas son las historias de usuario iniciales, a las que posteriormente hubo que añadir algunas, como por ejemplo la necesidad de añadir el módulo de la definición de la interfaz. Toda la información relativa a la metodología como las historias de usuario, división en tareas, iteraciones se explica en los Anexos.

7.3 Priorización inicial de historias de usuario

Tal y como determina la metodología XP, es el cliente el que elige la prioridad de las historias de usuario. Estas son las prioridades que el tutor asignó a las historias de usuario:

ID	Historia de usuario	Prioridad
HU-01	Elaborar un boceto de la aplicación	1
HU-02	Implementar el ámbito "REWARD" de la aplicación	9
HU-03	Implementar el ámbito "SOCIAL" de la aplicación	12
HU-04	Implementar el ámbito "FEEDBACK" de la aplicación	11
HU-05	Implementar el ámbito "GOALS" de la aplicación	10
HU-06	Implementar el ámbito "LOGIC" de la aplicación	8
HU-07	Implementar el ámbito "STORY-TELLING" de la aplicación	7
HU-08	Implementar el ámbito "DEBRIEFING" de la aplicación	14
HU-09	Implementar el ámbito "PERSISTENCE" de la aplicación	13
HU-10	Implementar el ámbito "SERVICES" de la aplicación	6
HU-11	Implementar el ámbito "INTERACTION" de la aplicación	4
HU-12	Implementar el ámbito "CHARACTERIZATION" de la aplicación	3
HU-13	Implementar el ámbito "SCENARIO" de la aplicación	2
HU-14	Implementar el ámbito "CONTEXT" de la aplicación	5
HU-15	Realización e interpretación de pruebas con usuarios finales	15

TABLA 6 - PRIORIZACIÓN DE HISTORIAS DE USUARIO

Como en el apartado anterior, es necesario aclarar que estas son las historias de usuario iniciales, a las que posteriormente hubo que añadir algunas, como por ejemplo la necesidad de añadir el módulo de la definición de la interfaz. Toda la información relativa a la metodología como las historias de usuario, división en tareas, iteraciones se explica en los Anexos.

7.4 Consideraciones iniciales

A lo largo de todos los módulos de la aplicación va a haber una serie de servicios que se le proporcionan al educador:

- Ayuda
- Abrir/Guardar diseños
- Generar informe

Como hemos dicho anteriormente, estos servicios se iban a desarrollar de manera transversal a lo largo de todas las historias de usuario, por lo que el funcionamiento de dichos servicios será el mismo para toda la aplicación.

7.4.1. Ayuda al educador

El caso de la ayuda es crítico, ya que se debía partir con la consideración de que el educador no tiene los conocimientos en informática suficientes ni tiene la capacidad de abstracción necesaria para poder diseñar un videojuego por sí solo. Por tanto se le debía brindar la máxima ayuda posible pero esta debe ser a su vez lo menos intrusiva posible para hacer lo más cómoda posible la tarea de diseñar el juego al educador.

Por tanto se pensó en tres mecanismos para proporcionar ayuda al cliente:

- Iconos de ayuda

  En el caso de los mensajes de ayuda, para que sean lo menos intrusivos posible, se muestran solo cuando el educador pose el cursor encima del icono. El mensaje aparecerá justo debajo del icono, de la forma que se detalla en la ilustración "Mensajes de ayuda":

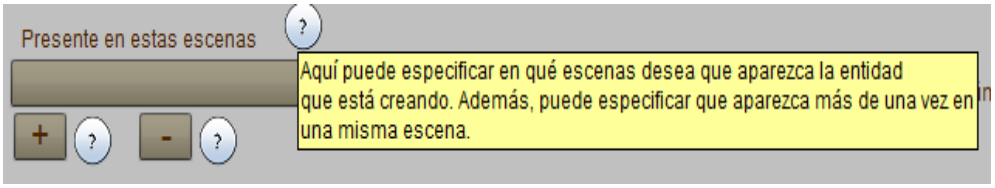

ILUSTRACIÓN 11 – MENSAJES DE AYUDA

  Cuando el educador mueva el cursor fuera del icono, el mensaje desaparecerá. De esta manera, solo se verá cuando el educador desee.

- Ventanas de transición

Las ventanas auxiliares tienen el aspecto que se muestra en la ilustración "Ventanas auxiliares":

ILUSTRACIÓN 12 – VENTANAS AUXILIARES

En estas ventanas se le muestra un diagrama con la fase en la que se encuentra, con la siguiente fase en transparente y con un texto lo más breve posible que le introduce en el objetivo que se quiere conseguir con la siguiente ventana. Al igual que con los mensajes de ayuda, para que sean lo menos intrusivas posible, solo aparecen cuando el educador pulse las flechas de la esquina inferior derecha para desplazarse por la aplicación.

Si el educador tuviera cierta experiencia con la aplicación y no quisiera ver estas ventanas, le bastaría con pulsar en el menú a la izquierda e iría directamente al paso que deseara.

- Ejemplo accesible en cualquier momento

En cualquier ventana de la aplicación, el educador puede pulsar el botón "Ver ejemplo" (en la esquina superior derecha siempre). Al pulsarlo, la aplicación guardará automáticamente toda la información que tenía en ese momento y se le cargará la información correspondiente a un ejemplo predefinido en todas las ventanas. De esta forma puede ver como se debe cumplimentar la información de cada módulo de una forma rápida y cómoda.

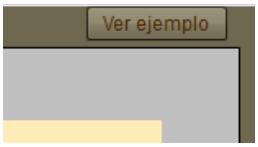

ILUSTRACIÓN 13 - BOTÓN "VER EJEMPLO"

Una vez pulse este botón, se le cambiará por un botón "Volver al diseño", que si pulsa la aplicación cargará toda la información del diseño que estaba creando antes de ver el ejemplo.

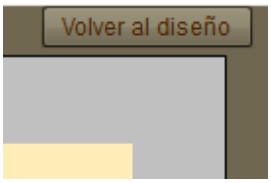

ILUSTRACIÓN 14 - BOTÓN "VOLVER AL DISEÑO"

Por último, como en la aplicación hay muchos desplegables que podrían tener muchas opciones, se decidió limitar el número de estas e incluir una opción "Otra@" para que el educador introduzca lo que él quiera.

De esta manera se evitan desplegables con multitud de opciones, que pudieran llegar a agobiar al educador e incluso a cansar una vez hubiera visto varios de ellos, ya que probablemente tuviera que leer la gran mayoría de las opciones para marcar la que quiera.

7.4.2. Abrir/Guardar diseños

La funcionalidad implementada para gestionar el almacenamiento de la información se realiza con los botones que hay en la parte superior de la aplicación: “Nuevo”, “Abrir”, “Guardar” y “Guardar como…”.

Si se pulsa el botón “Nuevo” se borrará toda la información almacenada hasta este momento y se volverá a la página de inicio. Si no se hubiera guardado la información, se le mostrará un mensaje al educador como el que se muestra en la ilustración "Mensaje de aviso al crear nuevo juego":

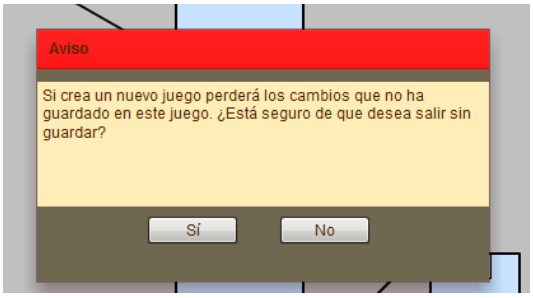

ILUSTRACIÓN 15 – MENSAJE DE AVISO AL CREAR NUEVO JUEGO

Si se pulsa el botón “Abrir” se abrirá un explorador de archivos para que el educador seleccione el archivo que hubiera guardado previamente. Si no se hubiera guardado la información, se le mostrará un mensaje al educador de la misma forma. Si la información se ha cargado correctamente se muestra un mensaje como se muestra en la ilustración "Mensaje de aviso al cargar la información correctamente":

ILUSTRACIÓN 16 – MENSAJE DE AVISO AL CARGAR LA INFORMACIÓN CORRECTAMENTE

Para realizar la funcionalidad del botón “Guardar” hay un problema inherente con Flex: como una RIA se puede realizar distribuídamente, esta no puede almacenar información sin que el usuario quiera, puesto que se podría utilizar con fines maliciosos. Por tanto, no se puede guardar sin que el usuario tenga que seleccionar el archivo en el explorador de archivos. La única diferencia con la funcionalidad de “Guardar como…” es que pulsando el botón “Guardar” aparece ya el nombre del archivo que se usó previamente para guardar.

Una posible solución para este problema era usar Adobe AIR, pero en ese caso no se podría usar en el futuro de manera distribuida y además hay clases utilizadas (el DragManager, por ejemplo) que no se pueden usar en Adobe AIR.

Como este problema con el guardado de archivos solo va a afectar en esta funcionalidad de la aplicación, se decidió conjuntamente con el tutor dejarlo de esta manera.

7.4.3. Generar informe

Finalmente, si se quiere generar informes habrá que ir al final de la aplicación hasta llegar a la ventana que se muestra en la ilustración "Ventana Generar Informes":

ILUSTRACIÓN 17 – VENTANA GENERAR INFORMES

El educador solo tendrá que pulsar el botón “Generar Informes” y se generará un PDF con toda la información introducida e imágenes con los distintos diagrama. Este PDF se genera gracias a la librería AlivePDF [32]

7.5 Distribución de los paquetes y módulos de la aplicación

Para favorecer la modularidad y reutilización de la aplicación, esta se dividió en paquetes y módulos. Estos paquetes y módulos se pueden ver en la ilustración "Diagrama de paquetes y módulos". En ella aparecen los paquetes de la aplicación representados por polígonos que representan a carpetas, mientras que los módulos están representados por rectángulos ubicados dentro de los paquetes.

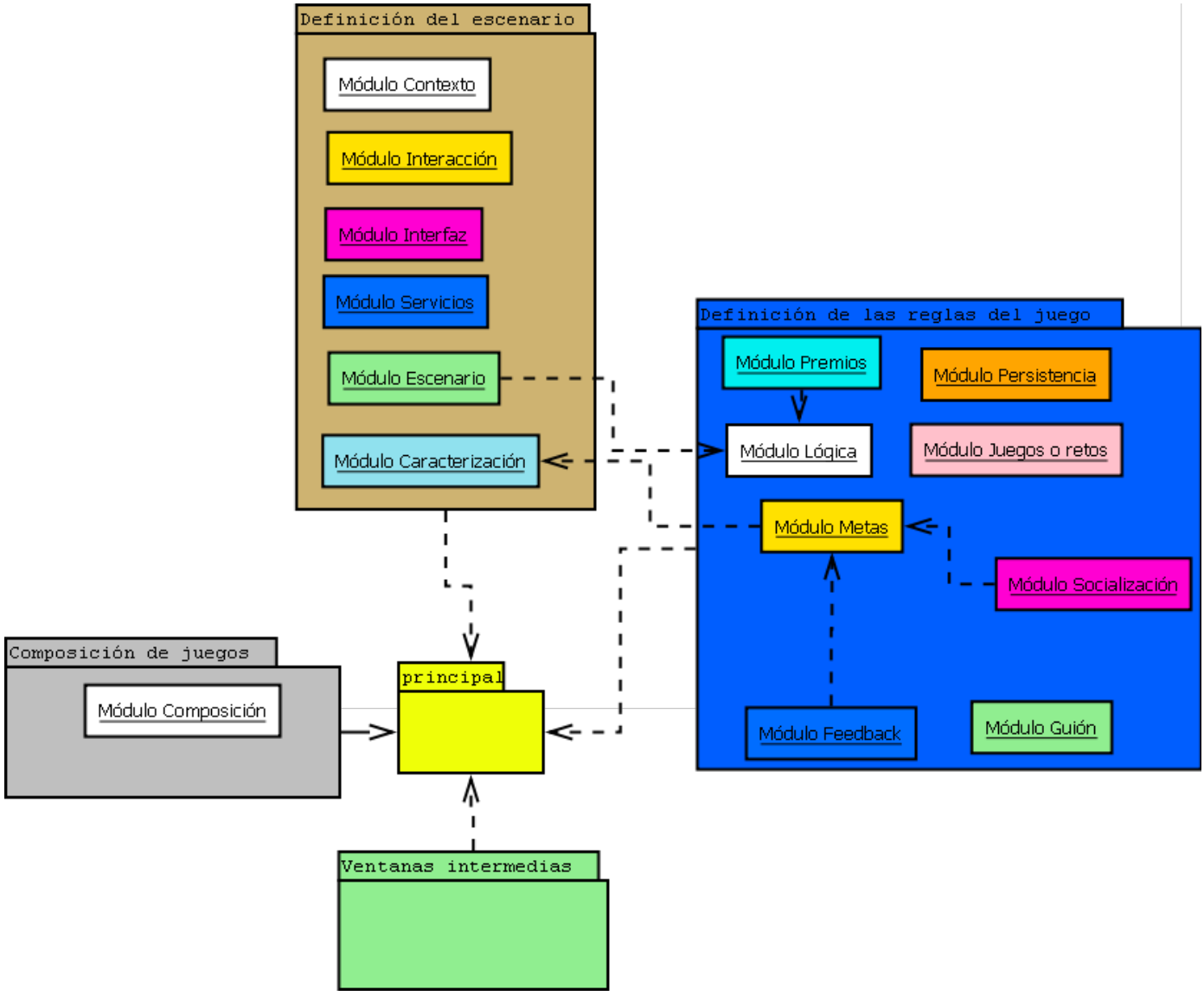

ILUSTRACIÓN 18 - DIAGRAMA DE PAQUETES Y MÓDULOS

7.6 Módulos de la "Definición de las reglas del juego"

7.6.1. Módulo "Lógica"

Diseño del módulo "Lógica"

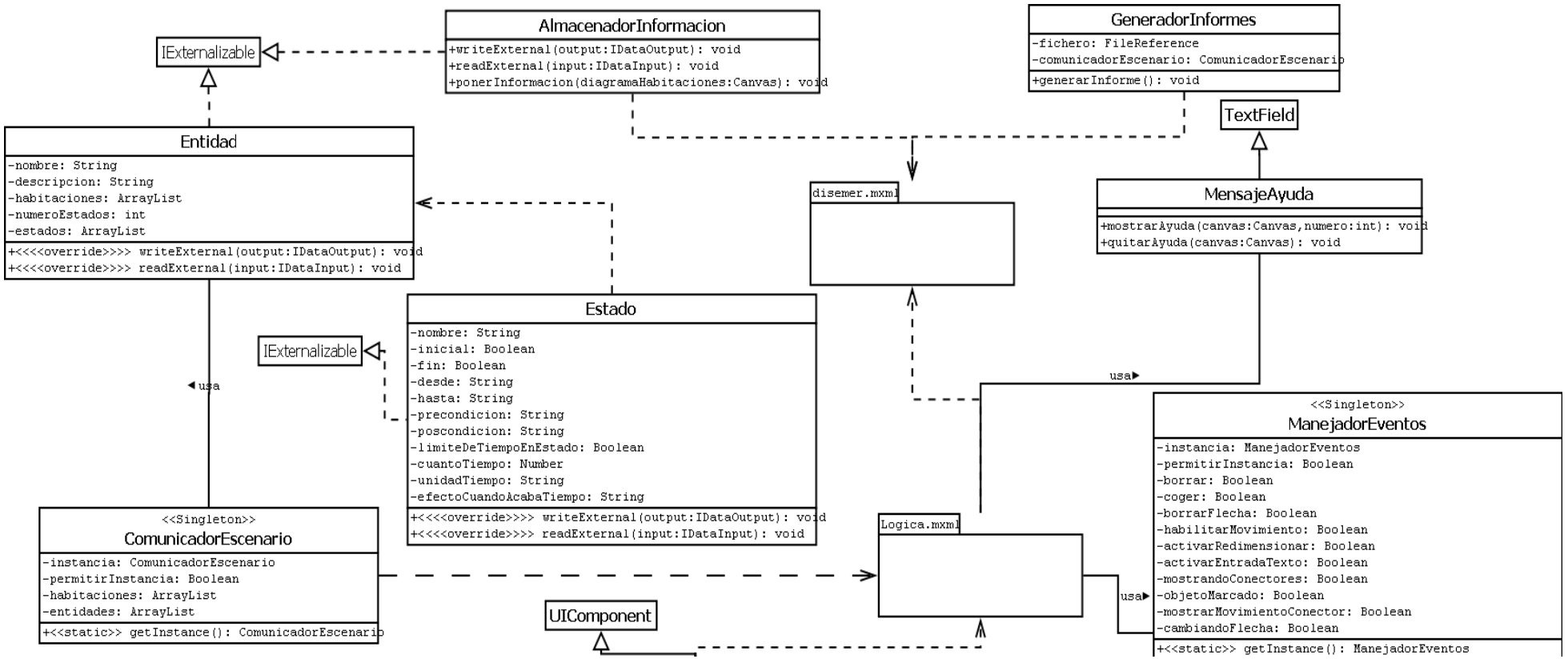

ILUSTRACIÓN 19 - DIAGRAMA DE CLASES DE HU-06 (1/2)

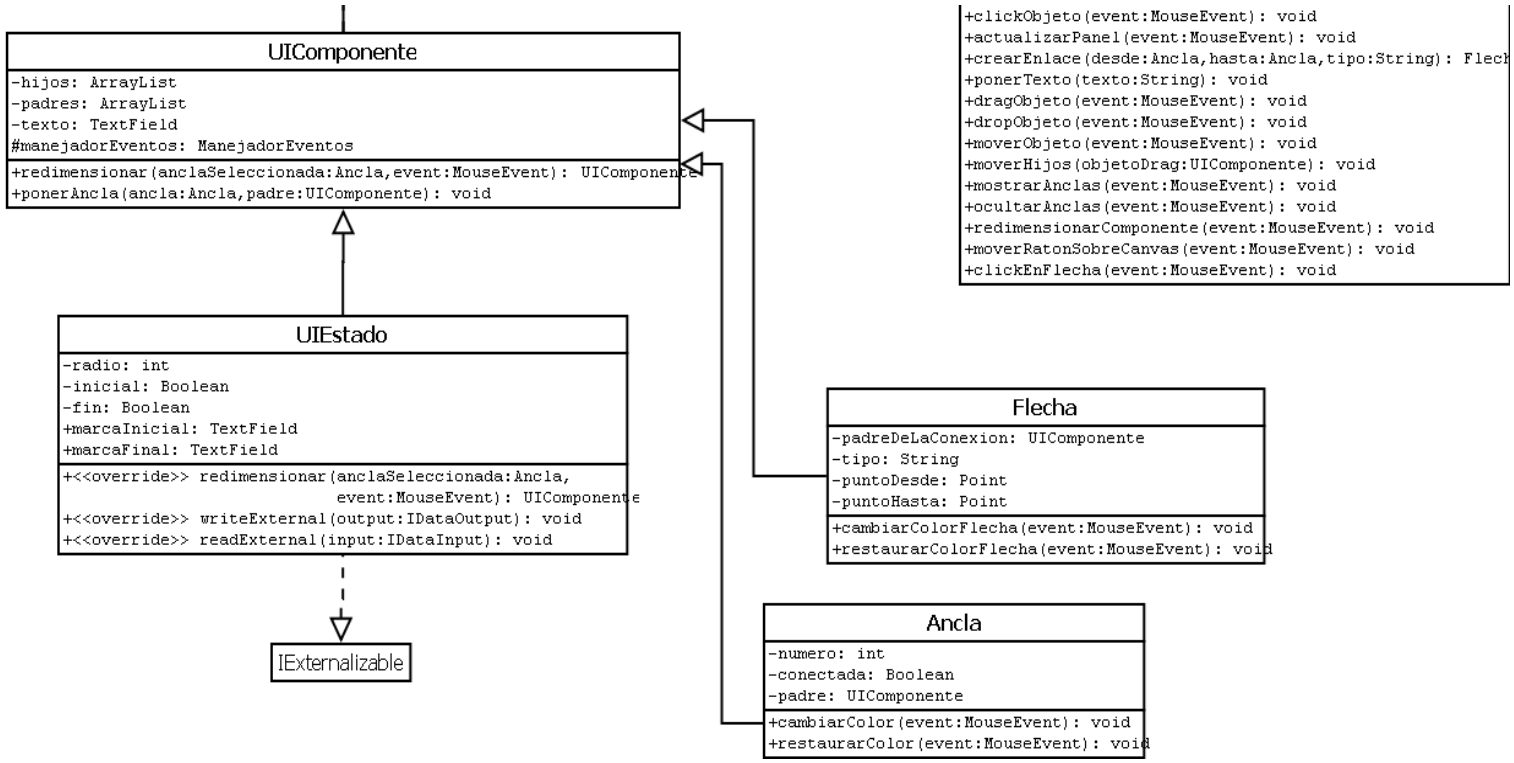

ILUSTRACIÓN 20 - DIAGRAMA DE CLASES DE HU-06 (2/2)

Pruebas del módulo "Lógica"

Este será el plan de pruebas para este módulo:

1. Crear varias entidades.
2. Asignar un tipo a una entidad.
3. Crear un nuevo estado pulsando “<<Nuevo Estado>>”
4. Marcarlo como estado inicial.
5. Seleccionar una precondición que suponga especificar textualmente qué precondición es.
6. Seleccionar una poscondición que suponga especificar textualmente qué poscondición es.
7. Marcar que hay un límite de tiempo en ese estado. Especificar un tiempo y la medida será en horas.
8. Seleccionar un efecto que suponga especificar textualmente cuál es el efecto cuando se acabe el tiempo.
9. Añadir el estado.
10. Crear otro estado desde el canvas. Renombrarlo y marcarlo como estado final.
11. Marcar como desde y hasta el estado inicial.
12. Marcar que no hay límite de tiempo.
13. Comprobar que se establece automáticamente una unión bidireccional entre ambos.
14. Cambiar de entidad y luego volver a la anterior. Ver que se crea automáticamente el diagrama antes elaborado.
15. Ir a otra entidad.
16. Crear dos estados de manera que se cree una relación unidireccional entre ellos automáticamente: en el segundo estado se especificará como desde el primero.
17. Ir a otra entidad.
18. Crear un estado inicial.
19. Crear otro estado e intentar marcarlo como inicial. Ver que se muestra un error.
20. Intentar crear una unión unidireccional desde un estado a la entidad. Ver que se muestra un error.
21. Crear una relación bidireccional entre ambos estados.
22. Crear otro estado.
23. Intentar crear una relación unidireccional desde ese estado a uno de los dos anteriormente creados. Ver que se muestra un error.
24. Intentar lo mismo con una relación bidireccional. Ver que se muestra error.
25. Borrar el último estado creado desde el botón del canvas.
26. Crear un estado y darle nombre.
27. Hacer click en ese estado.
28. Borrar el último estado creado desde el botón del panel principal.
29. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
30. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
31. Generar informe.
32. Comprobar que el informe que se ha generado es correcto.
33. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
34. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Lógica"

El módulo "Lógica" se compone de dos ventanas: "Lógica 1/2", donde se crean las entidades y se da una descripción de cómo van a ser gráficamente y "Lógica 2/2", donde se les da el comportamiento que tendrán en el juego (estados) estas entidades.

La finalidad de la ventana "Lógica 1/2" es muy simple: en ella se permite crear entidades y dotarlas de una descripción para que los programadores sepan cómo deben implementarla gráficamente. Si el educador lo deseara, podría editar entidades ya creadas y cambiarles el nombre o la descripción.

La ventana donde se realiza la funcionalidad se muestra en la ilustración "Ventana Lógica 1/2":

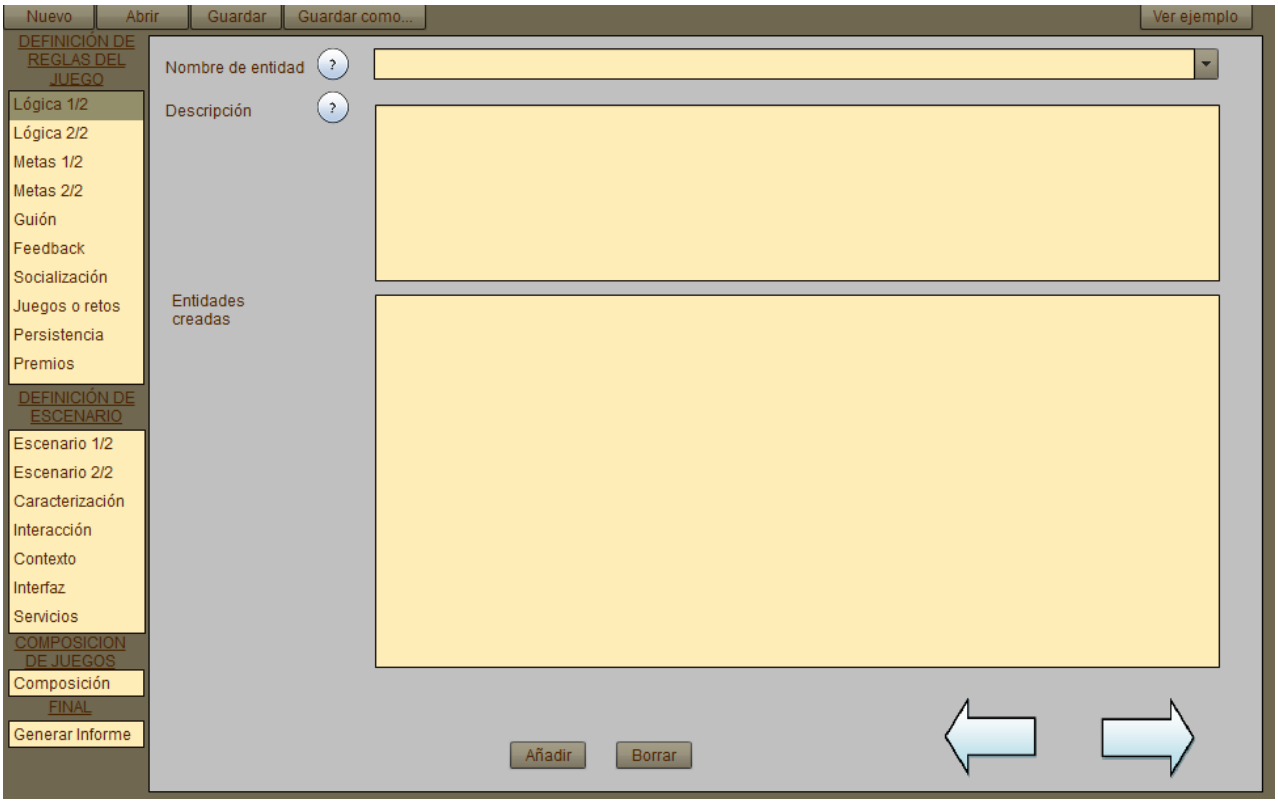

ILUSTRACIÓN 21 – VENTANA LÓGICA 1/2

Una vez se han creado las entidades, en la ventana "Lógica 2/2" se les podrá dotar de comportamiento, ya que se permite definir los distintos estados que van a tener dichas entidades y como se realizan las transiciones de uno a otro. Además, se permite la definición de precondiciones para poder alcanzar un estado y poscondiciones que suceden cuando la entidad sale del estado.

Por último, también se permite definir efectos que suceden cuando una entidad está un tiempo determinado en un mismo estado (por ejemplo: perder la partida).

En este caso, se va a explicar cómo funciona la ventana ya que es una de las más complejas de la aplicación:

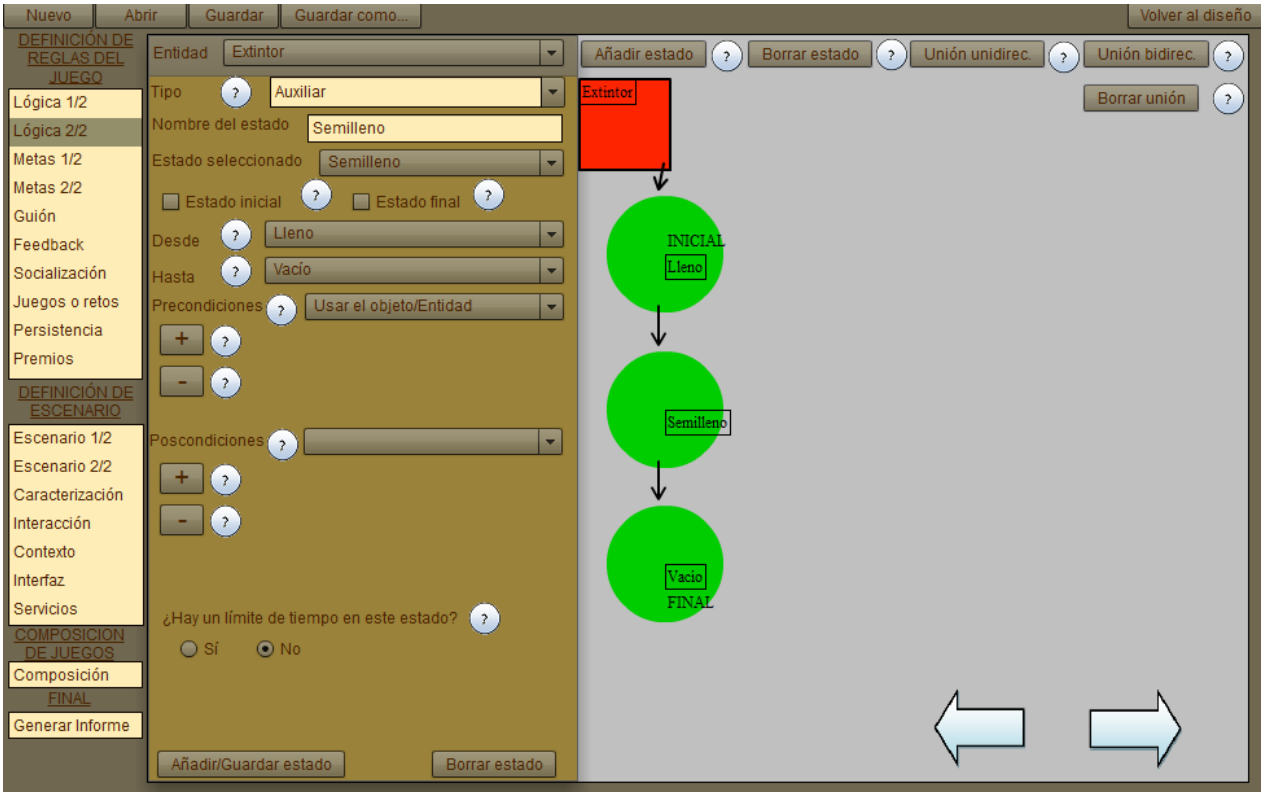

ILUSTRACIÓN 22 – VENTANA LÓGICA

Para ayudar al educador hay dos partes dentro de la misma: un panel principal a la izquierda y un canvas a la derecha. El educador podrá definir los estados de la entidad tanto en una parte como en otra, pero lo que se haga en una repercutirá en la otra automáticamente.

Por ejemplo, si se crea un estado de una entidad mediante el panel, al darle a "Añadir Estado" se creará automáticamente un círculo en el canvas. Y si especificase que un estado viene de otro ("Desde") al darle a "Añadir Estado" se crearía un enlace unidireccional automáticamente. Y viceversa: si se creara un enlace unidireccional en el panel se pondría automáticamente "Desde".

Esto se ha hecho de esta forma para ayudar el trabajo del educador lo máximo posible, ya que esta es probablemente la parte más difícil del diseño.

Después, hay una serie de acciones que la aplicación no podrá permitir en esta ventana: por ejemplo, si un educador trata de hacer una unión unidireccional desde un estado hasta la entidad (representada por un cuadrado) o si se intentara marcar un estado como inicial habiendo ya otro la aplicación mostraría mensajes de error.

7.6.2. Módulo "Metas"

Diseño del módulo "Metas"

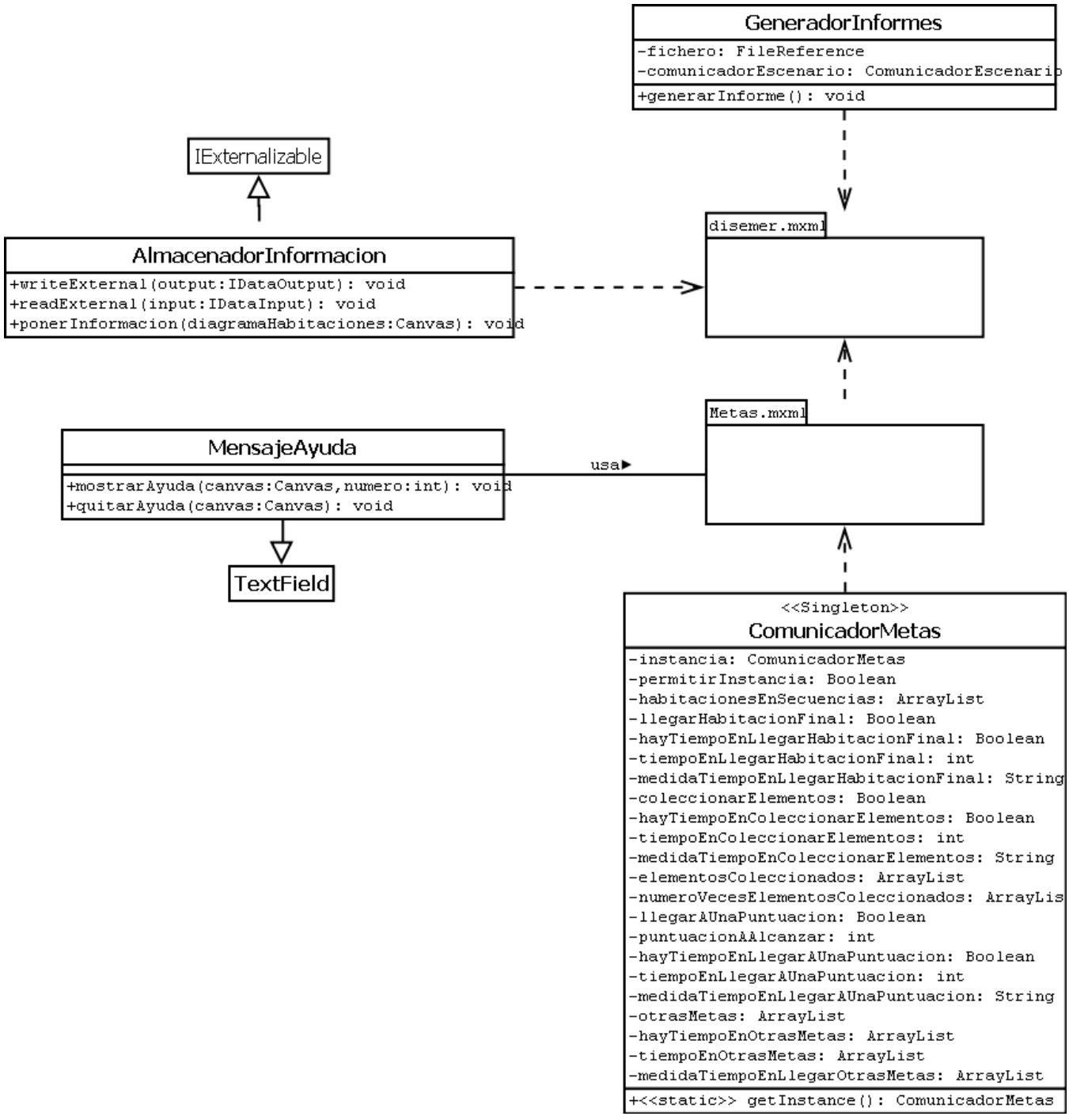

ILUSTRACIÓN 23 - DIAGRAMA DE CLASES DE HU-05

Pruebas del módulo "Metas"

Este será el plan de pruebas para este módulo:

1. Marcar todas las metas y marcar las opciones de que tengan un límite en el tiempo. Ver que al marcar las opciones aparece la información necesaria.
2. Volver a desmarcarla. Ver que la información desaparece.
3. Marcar la opción “Coleccionar entidades”.
4. Seleccionar más de dos entidades.
5. Pulsar en “Guardar”.
6. Renombrar una entidad.
7. Volver a la ventana de metas. Comprobar que se ha renombrado.
8. Borrar una entidad.
9. Volver a la ventana de metas. Comprobar que se ha borrado.
10. Seleccionar el resto de metas aleatoriamente.
11. Comprobar que en los desplegables de "Supermeta" se incluyen las metas marcadas.
12. Ver que si se cambia a las metas de un avatar en concreto, estas metas cambian.
13. Asignar umbrales al azar para todos los jugadores en general.
14. Repetir paso "12" con los umbrales.
15. Guardar el archivo.
16. Cerrar la aplicación.
17. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
18. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
19. Generar informe.
20. Comprobar que el informe que se ha generado es correcto.
21. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
22. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Metas"

Al igual que en el módulo anterior, este módulo está compuesto por dos ventanas: "Metas 1/2", donde se definirán las metas para todos los jugadores en general o para jugadores que estén usando un avatar en concreto y "Metas 2/2", en la que se podrán definir umbrales para las metas (por ejemplo, que el jugador haya conseguido un 70% de las metas).

En el caso de la primera ventana de metas, lo que se permite hacer en ella es definir las metas que tienen que cumplir todos los jugadores o los que estén usando un avatar en concreto. Se han predefinido 3 tipos de metas por defecto:

- Llegar a una escena final
- Coleccionar entidades (habrá que definir cuáles) y Respecto a las entidades para coleccionar, si posteriormente se renombra o se borra alguna entidad, se actualizará automáticamente la información de esta ventana.
- Llegar a una puntación

Aunque si el educador lo estima oportuno, podría definir hasta 5 metas nuevas que a él se le ocurran.

Además, para cada meta es posible definir un límite de tiempo para superarla.

Por último, la aplicación permite definir composiciones de metas, de manera que para poder cumplir una haya que cumplir las metas que estén definidas como "submetas".

En la ilustración "Ventana Metas" se muestra cómo se implementan estas funcionalidades:

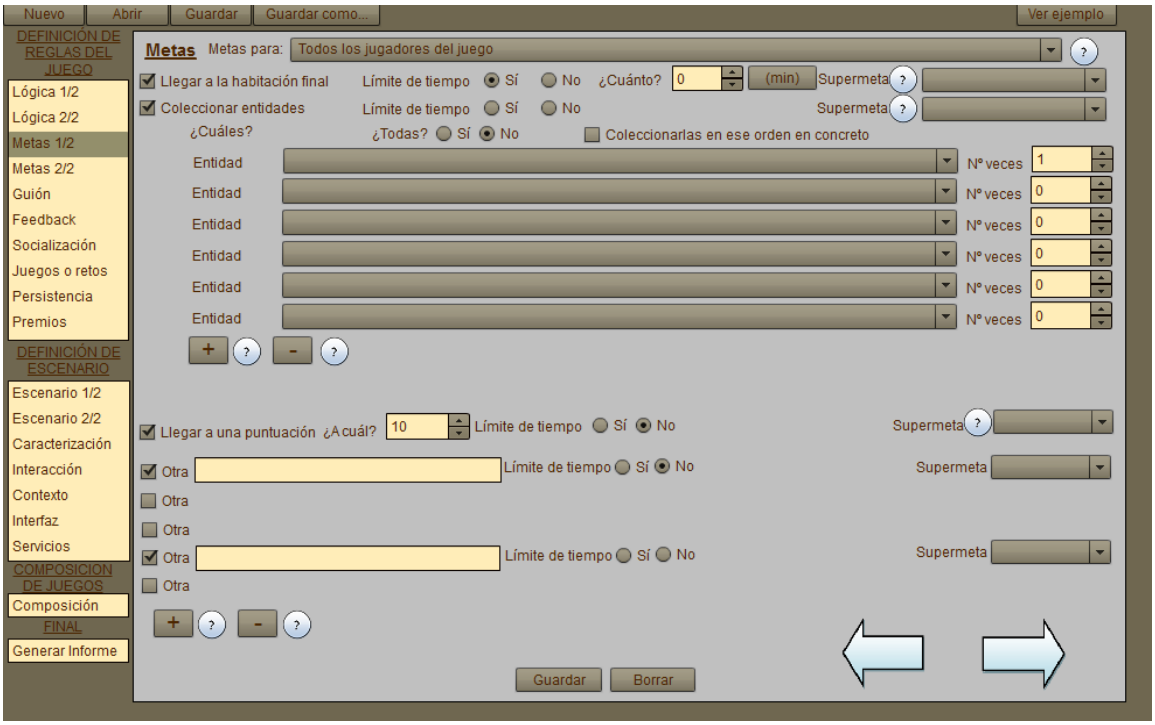

ILUSTRACIÓN 24 – VENTANA METAS

Mientras que en la segunda parte del módulo "Metas" se permite establecer umbrales para todos los jugadores o para avatares específicos (al igual que para definir metas). Estos umbrales pueden oscilar del 0% al 100% de las metas. Como no todas las metas son cuantificables, se ha definido que estos porcentajes equivalgan por defecto a lo siguiente:

- 0-29%: Se pide coleccionar el porcentaje introducido de entidades y conseguir dicho porcentaje de puntos. No se pide conseguir ninguna meta más.
- 30% - 49%: Se pide coleccionar el porcentaje introducido de entidades y conseguir dicho porcentaje de puntos. Del resto de metas se pide conseguir un tercio de ellas (redondeando hacia arriba).
- 50% - 100%: Se pide coleccionar el porcentaje introducido de entidades y conseguir dicho porcentaje de puntos. Se pide conseguir el resto de metas.

No obstante, si al educador no le gustaran esta forma de considerar los umbrales se le permite definir la forma que prefiera. La ventana creada para la definición de umbrales se puede ver en la ilustración "Ventana de definición de umbrales (metas)":

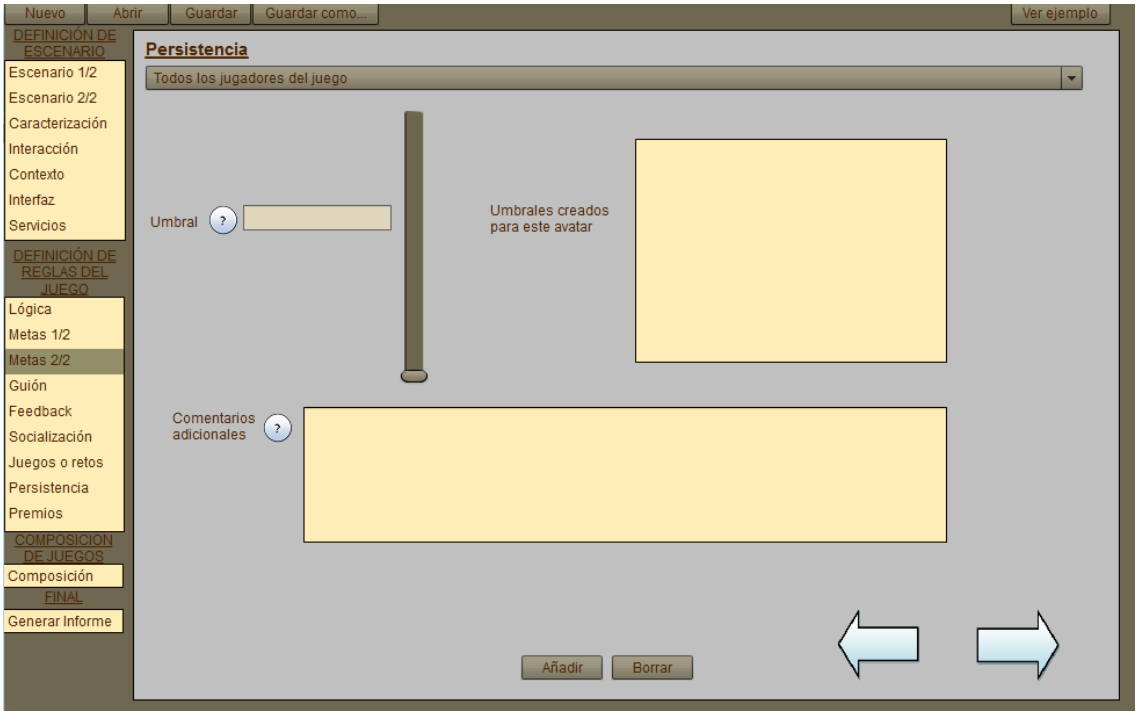

ILUSTRACIÓN 25 – VENTANA DE DEFINICIÓN DE UMBRALES (METAS)

7.6.3. Módulo "Guión"

Diseño del módulo "Guión"

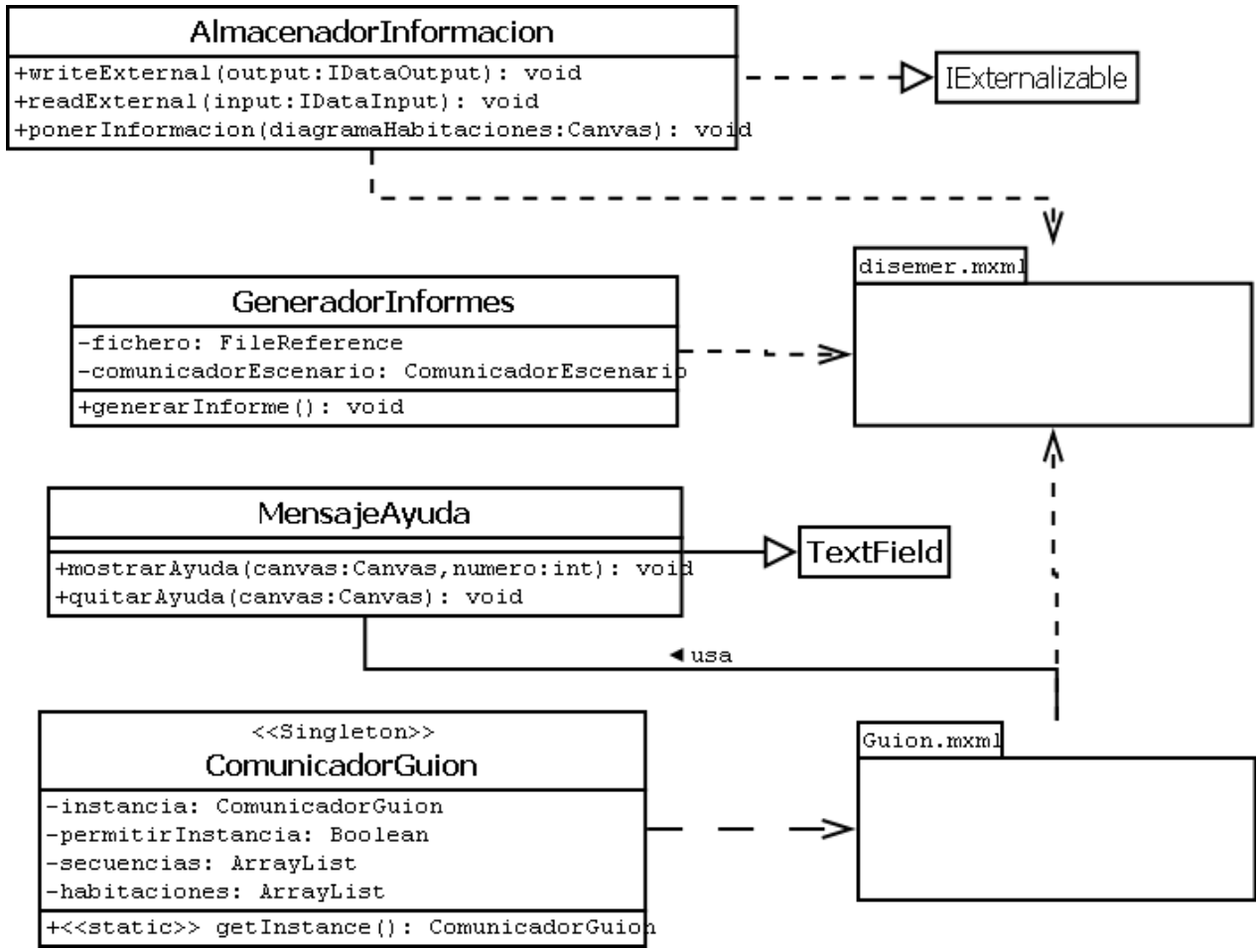

ILUSTRACIÓN 26 - DIAGRAMA DE CLASES DE HU-07

Pruebas del módulo "Guión"

Este será el plan de pruebas para este módulo:

1. Introducir valores en al menos tres secuencias.
2. Guardar el archivo.
3. Cerrar la aplicación.
4. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
5. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
6. Generar informe.
7. Comprobar que el informe que se ha generado es correcto.
8. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
9. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Guión"

Este módulo es más simple que los anteriores, ya que en él se pueden introducir todas las secuencias de las que se compondrá el "guión" del juego pulsando. En total, el educador podrá introducir 17 secuencias si así lo desea.

El aspecto de la ventana del guión es el siguiente, como se puede ver en la ilustración "Ventana Guión":

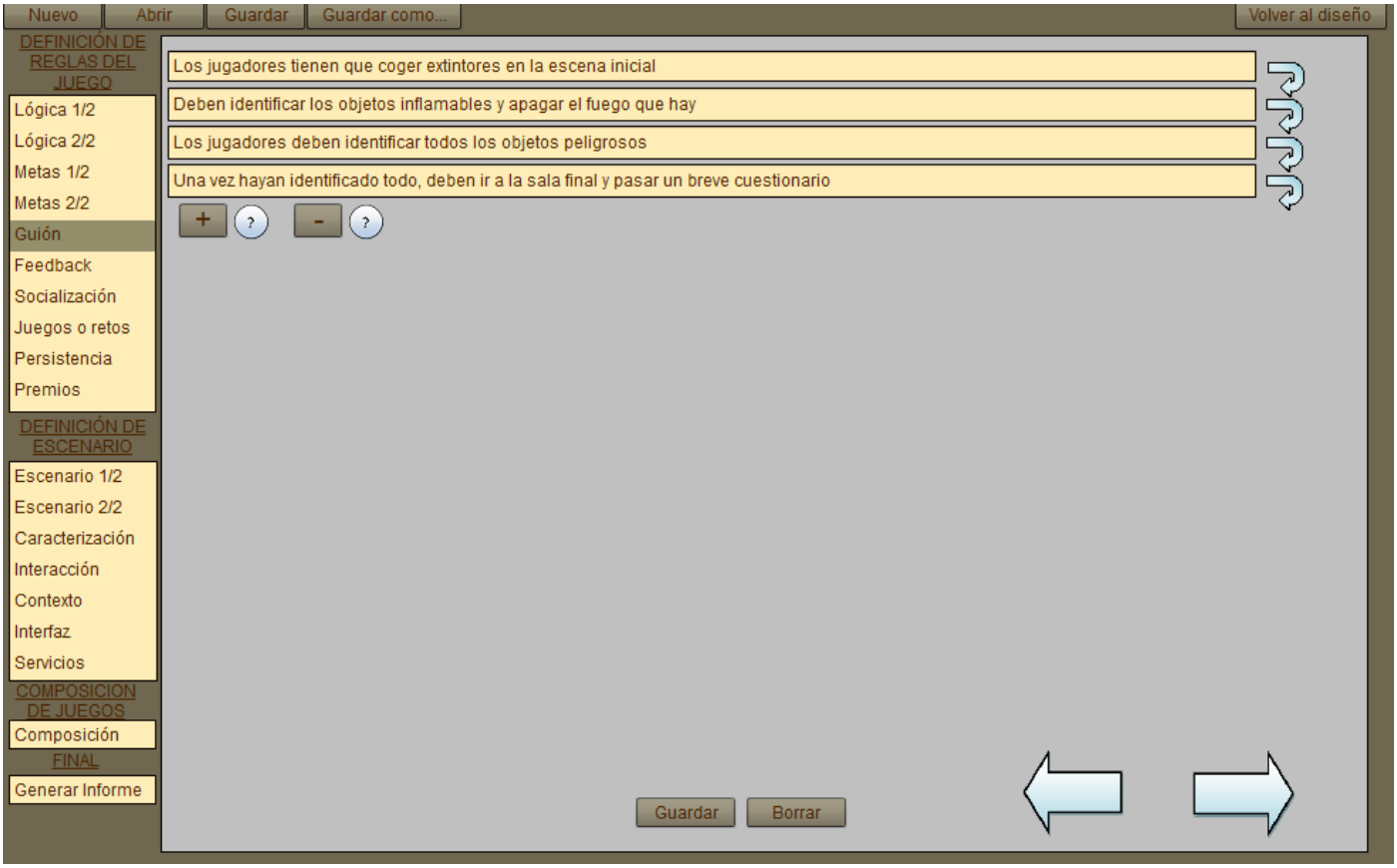

ILUSTRACIÓN 27 – VENTANA GUIÓN

7.6.4. Módulo "Feedback"

Diseño del módulo "Feedback"

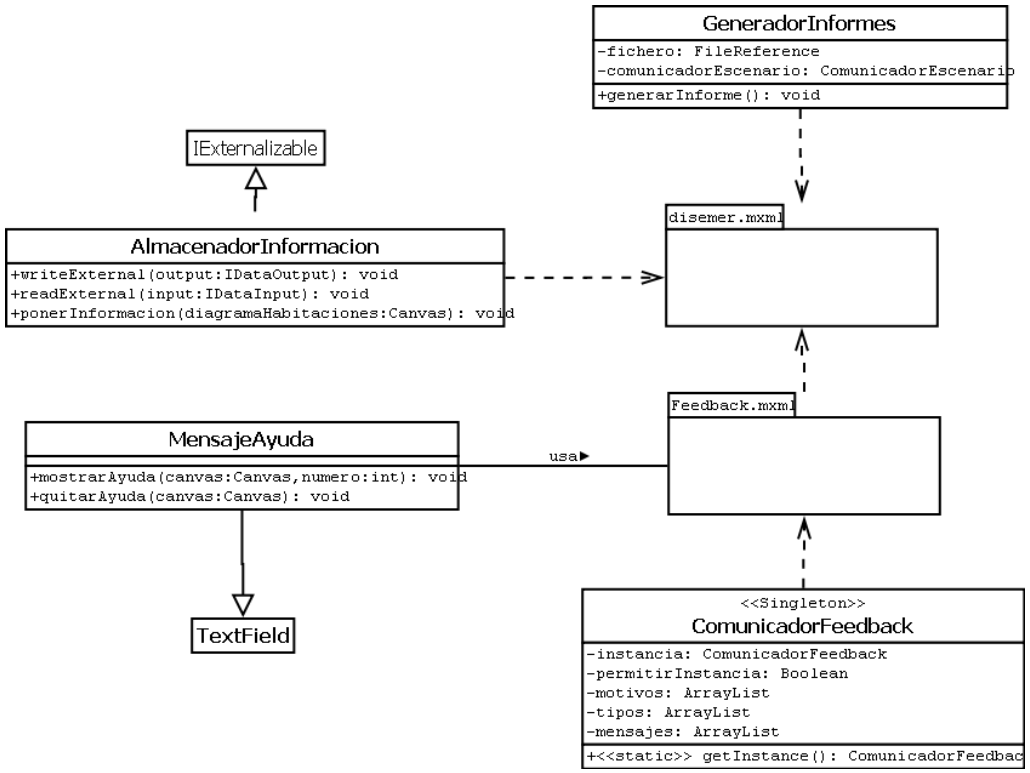

ILUSTRACIÓN 28 - DIAGRAMA DE CLASES DE HU-04

Pruebas del módulo "Feedback"

Este será el plan de pruebas para este módulo:

1. Introducir al menos doce motivos para mostrar mensajes, al menos dos de cada tipo.
2. Pulsar en “Guardar”.
3. Desmarcar la meta que se hubiera introducido en la opción “Cuando se consiga una meta”.
4. Volver a la ventana de “Feedback”. Ver que ha desaparecido.
5. Volver a las metas. Desmarcar la meta que se hubiera introducido en la opción “Cuando lleve un tiempo sin conseguir una meta”.
6. Volver a la ventana de “Feedback”. Ver que ha desaparecido.
7. Volver a desmarcarla. Ver que la información desaparece.
8. Repetir los pasos 3-7 para los umbrales de metas.
9. Guardar el archivo.
10. Cerrar la aplicación.
11. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
12. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
13. Generar informe.

14. Comprobar que el informe que se ha generado es correcto.
15. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
16. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Feedback"

En esta ventana, el educador puede ir poniendo los motivos por los que se mostrarán mensajes a los jugadores y los mensajes en cuestión. En total se pueden detallar hasta 17 mensajes. Además, se puede definir el tipo del mensaje. Los posibles tipos son:

- Mensaje de aviso
- Nueva pantalla con mensaje de aviso
- Imagen
- Animación
- Vídeo
- Sonido
- Vibración de pantalla
- Vibración de mando

Mientras que los posibles motivos para mandar un mensaje son:

- Conseguir una meta
- Llevar un tiempo sin conseguir una meta
- Alcanzar un umbral de metas
- Llevar un tiempo sin alcanzar un umbral de metas
- Cuando lleve un tiempo sin realizar ninguna acción
- Al comienzo de la partida
- Otro

Todos estos tipos y motivos se pueden seleccionar tal en la ventana que se muestra en la ilustración "Ventana Feedback":

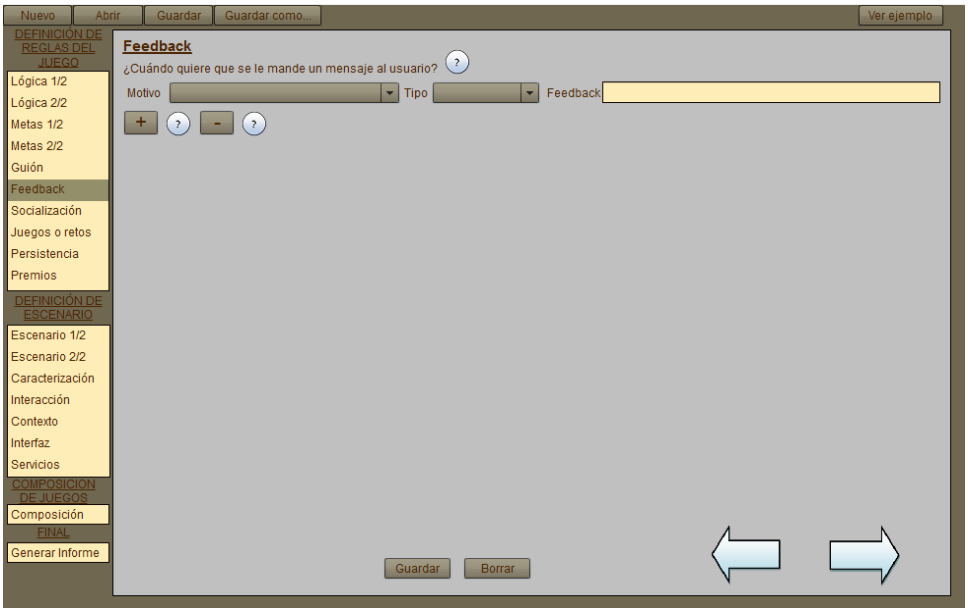

ILUSTRACIÓN 29 – VENTANA FEEDBACK

7.6.5. Módulo "Socialización"

Diseño del módulo "Socialización"

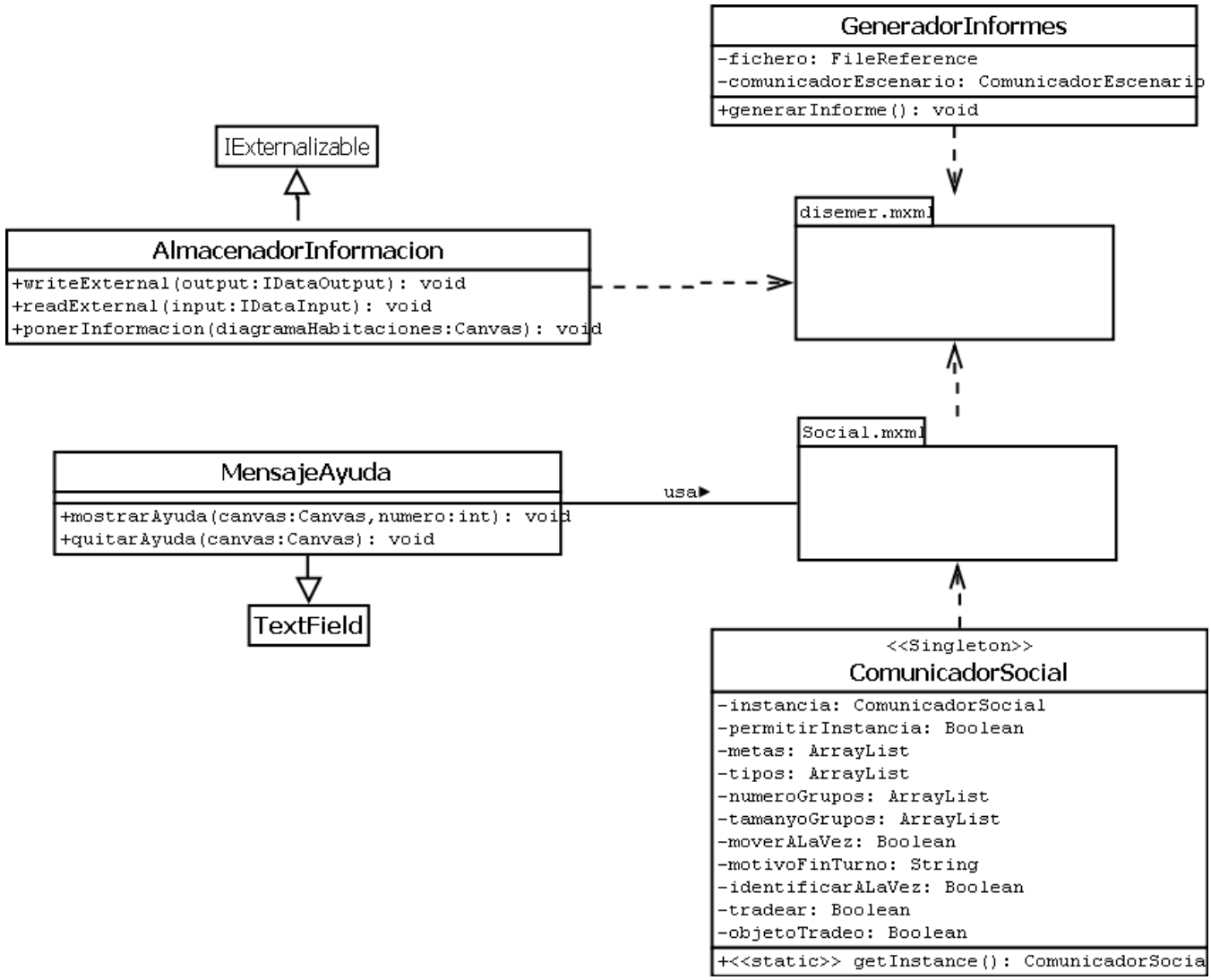

ILUSTRACIÓN 30 - DIAGRAMA DE CLASES DE HU-03

Pruebas del módulo "Socialización"

Este será el plan de pruebas para el módulo:

1. Introducir al menos una meta de cada tipo en “Metas”.
2. Pulsar en “Guardar”.
3. Ir a “Socialización”.
4. Asignar a cada una de las cuatro metas creadas un tipo “Competitiva”/”Cooperativa” y un tamaño de grupo y número de integrantes arbitrario.
5. Ir borrando una a una las metas en “METAS” y ver que se borran también en “Socialización”.
6. Marcar arbitrariamente los “Check box”.
7. Guardar el archivo.

8. Cerrar la aplicación.
9. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
10. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
11. Generar informe.
12. Comprobar que el informe que se ha generado es correcto.
13. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
14. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Socialización"

En este módulo, se puede definir cómo va a ser el juego: si en tiempo real (los jugadores podrán moverse y realizar acciones a la vez) o por turnos (en cuyo caso habrá que definir cuándo se acaban los turnos).

Además se pueden definir los tipos de mensajes que se pueden enviar entre sí los jugadores: síncronos o asíncronos y si se permite que los alumnos envíen mensajes a los educadores y viceversa.

Si el educador quisiera también podría indicar que los jugadores pudieran negociar y con qué pueden negociar.

Por último, se puede definir si una meta se quiere que sea competitiva o cooperativa y el número de grupos y el tamaño de los mismos que se formarán en cada caso. Sin embargo, no podrá definir este tipo de metas si no ha marcado ninguna meta anteriormente.

Esta información se puede introducir en esta ventana, tal y como se muestra en la ilustración "Ventana Socialización":

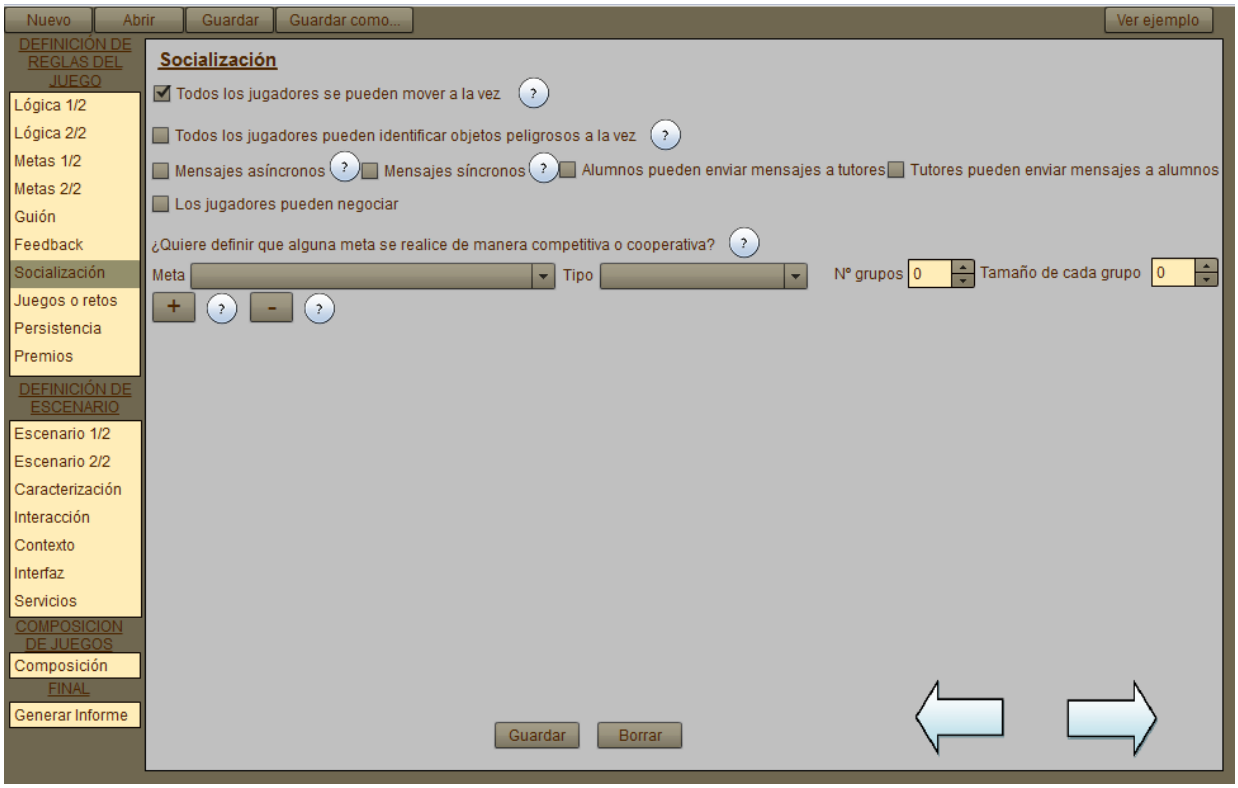

ILUSTRACIÓN 31 – VENTANA SOCIALIZACIÓN

7.6.6. Módulo "Juegos o retos"

Diseño del módulo "Juegos o retos"

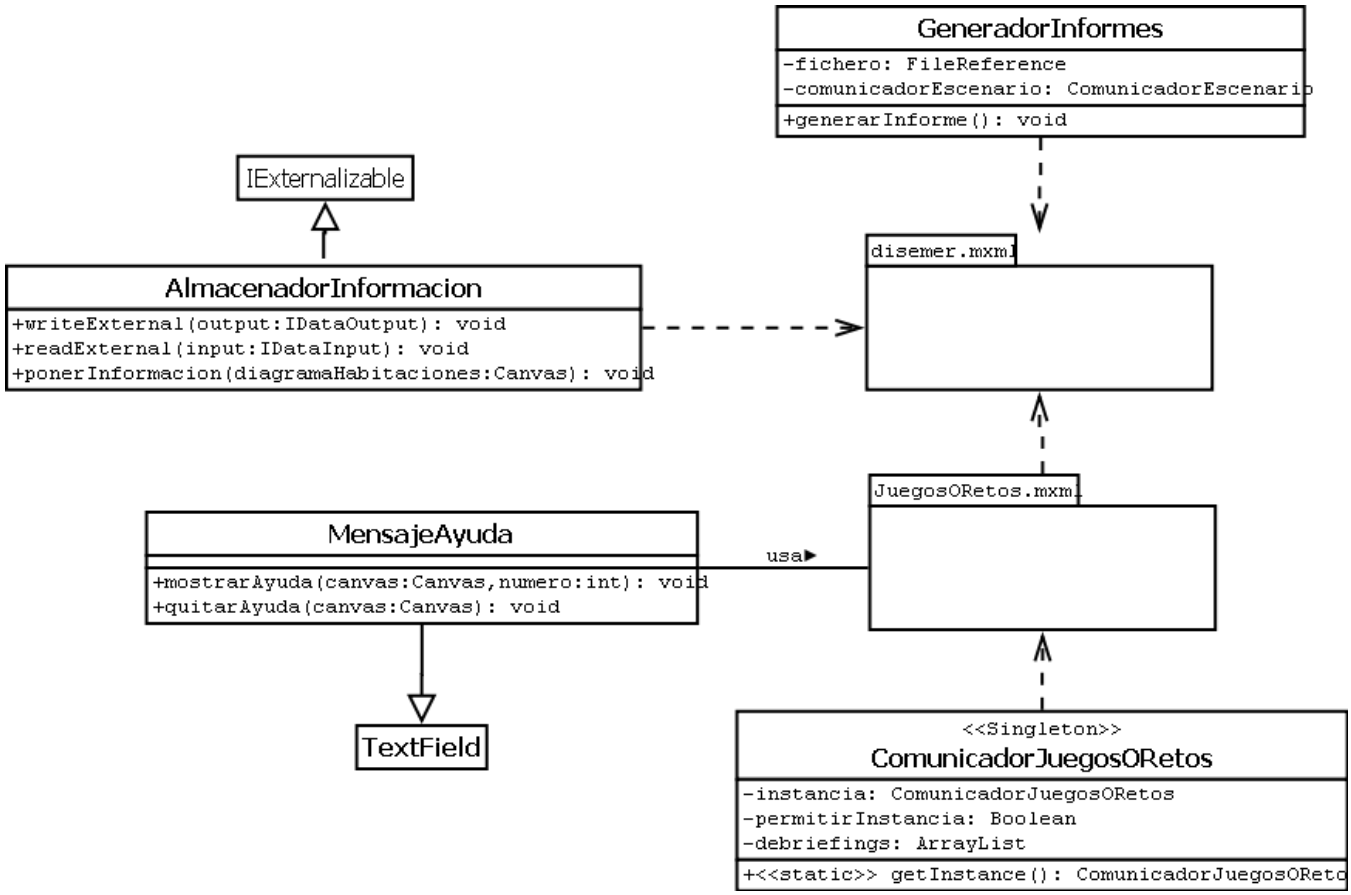

ILUSTRACIÓN 32 - DIAGRAMA DE CLASES DE HU-08

Pruebas del módulo "Juegos o retos"

Este será el plan de pruebas para el módulo:

1. Crear dos juegos o retos.
2. Guardar el archivo.
3. Cerrar la aplicación.
4. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
5. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
6. Generar informe.
7. Comprobar que el informe que se ha generado es correcto.
8. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
9. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Juegos o retos"

En esta ventana, el educador podrá crear juegos o retos para que los jugadores los tengan que superar. El educador tendrá que especificar en qué consiste, la aplicación le proporciona dos opciones (que un jugador realice un dibujo o un cuestionario), pero se le permite indicar lo que él quiera que se realice. Una filosofía que se ha llevado a cabo es minimizar el número de posibles opciones con las más probables y añadiendo una opción “Otr@” por si al educador se le ocurre una opción no contemplada a priori.

Para que estos juegos o retos se les aparezcan a los jugadores, estos deben haber cumplido una serie de precondiciones que el educador puede determinar en este módulo.

El aspecto de la ventana de juegos o retos se muestra en la ilustración "Ventana juegos o retos":

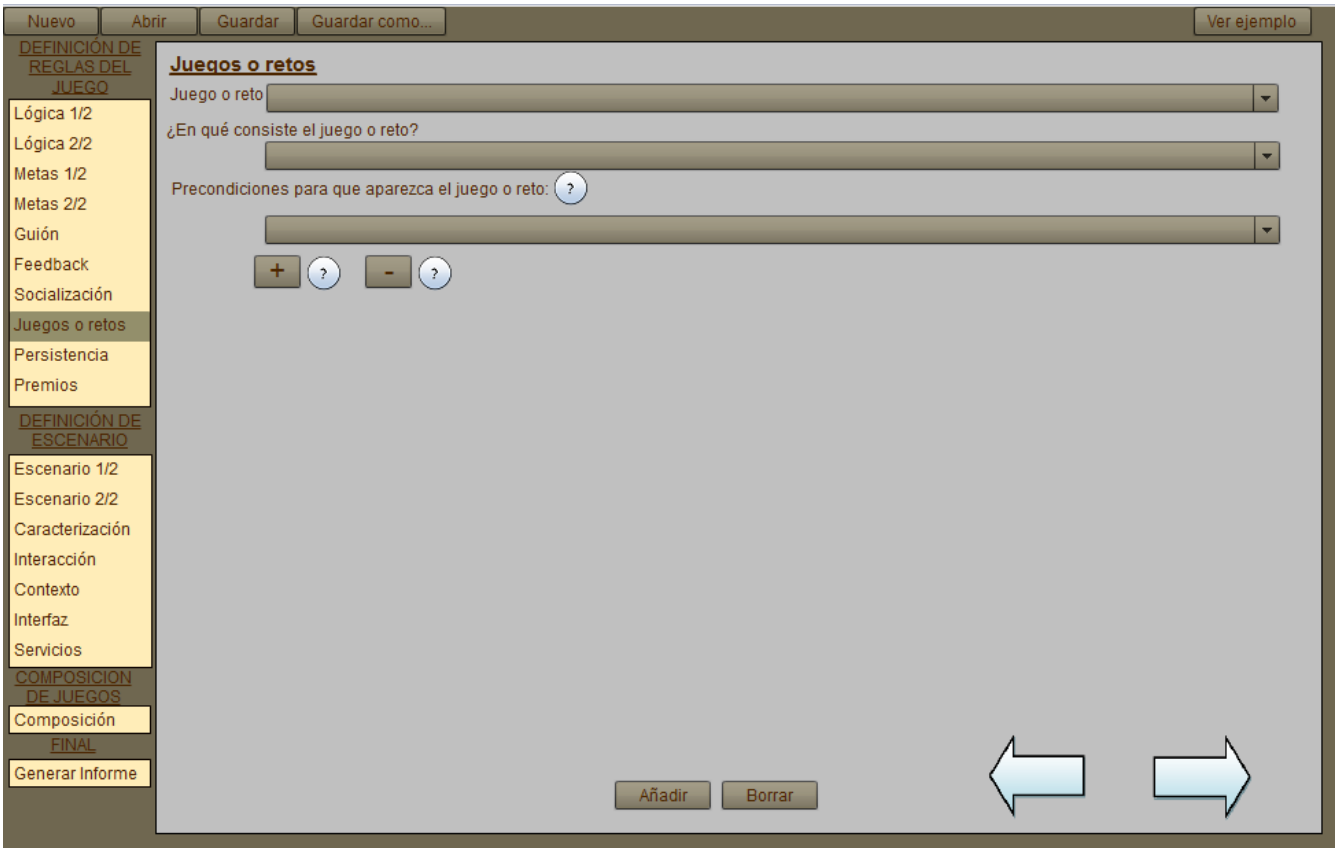

ILUSTRACIÓN 33 – VENTANA JUEGOS O RETOS

7.6.7. Módulo "Persistencia"

Diseño del módulo "Persistencia"

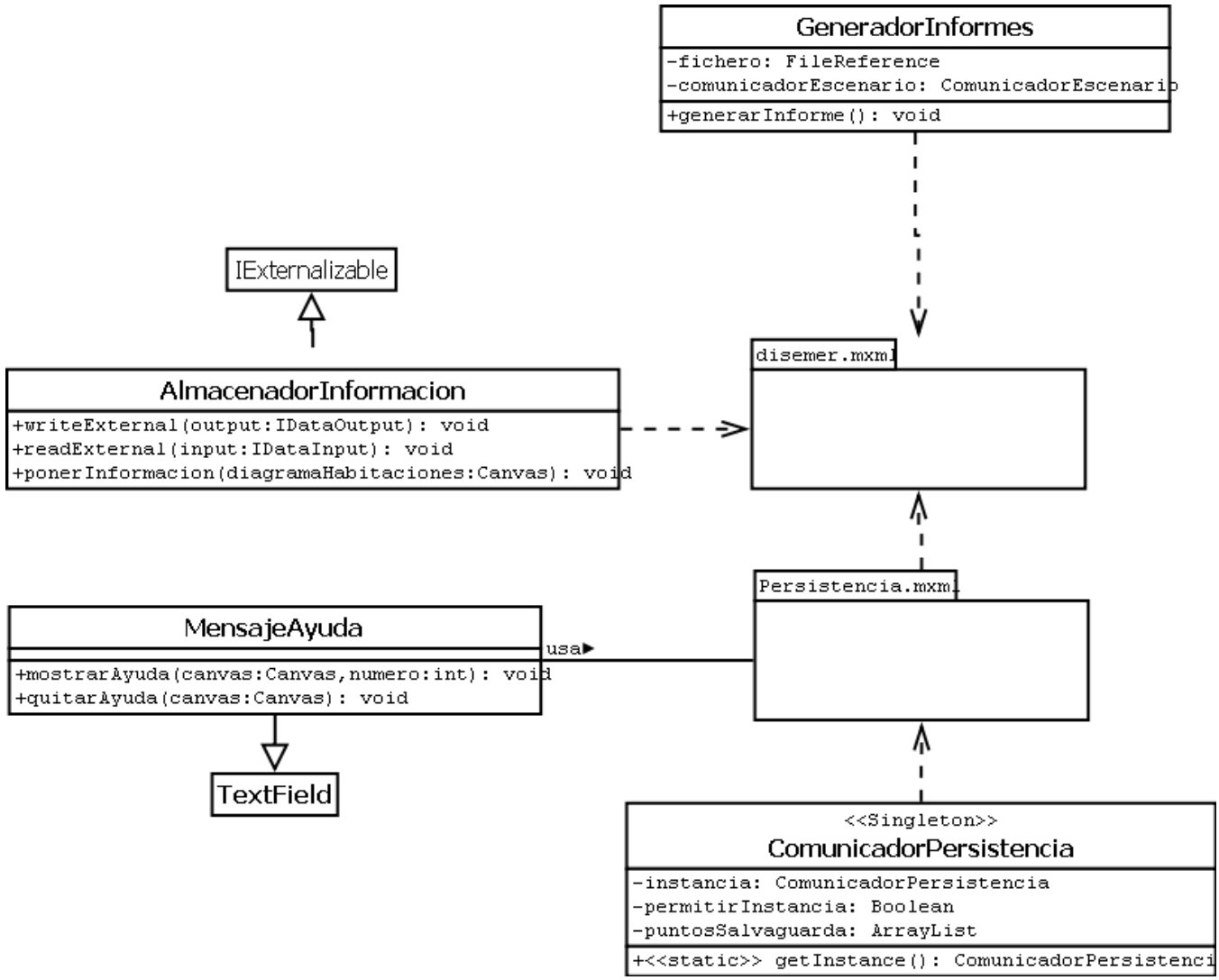

ILUSTRACIÓN 34 - DIAGRAMA DE CLASES DE HU-09

Pruebas del módulo "Persistencia"

Este será el plan de pruebas para este módulo:

1. Crear dos puntos de salvaguarda
2. Guardar el archivo.
3. Cerrar la aplicación.
4. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
5. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
6. Generar informe.

7. Comprobar que el informe que se ha generado es correcto.
8. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
9. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Persistencia"

El funcionamiento de este módulo es prácticamente igual al módulo de juegos o retos, con la excepción de que no tiene un desplegable para indicar en qué consiste, ya que todos los que se creen van a ser puntos de salvaguarda: una vez un jugador haya cumplido todas las precondiciones podrá guardar el estado actual de la partida.

El aspecto de la ventana de persistencia se muestra en la ilustración "Ventana persistencia":

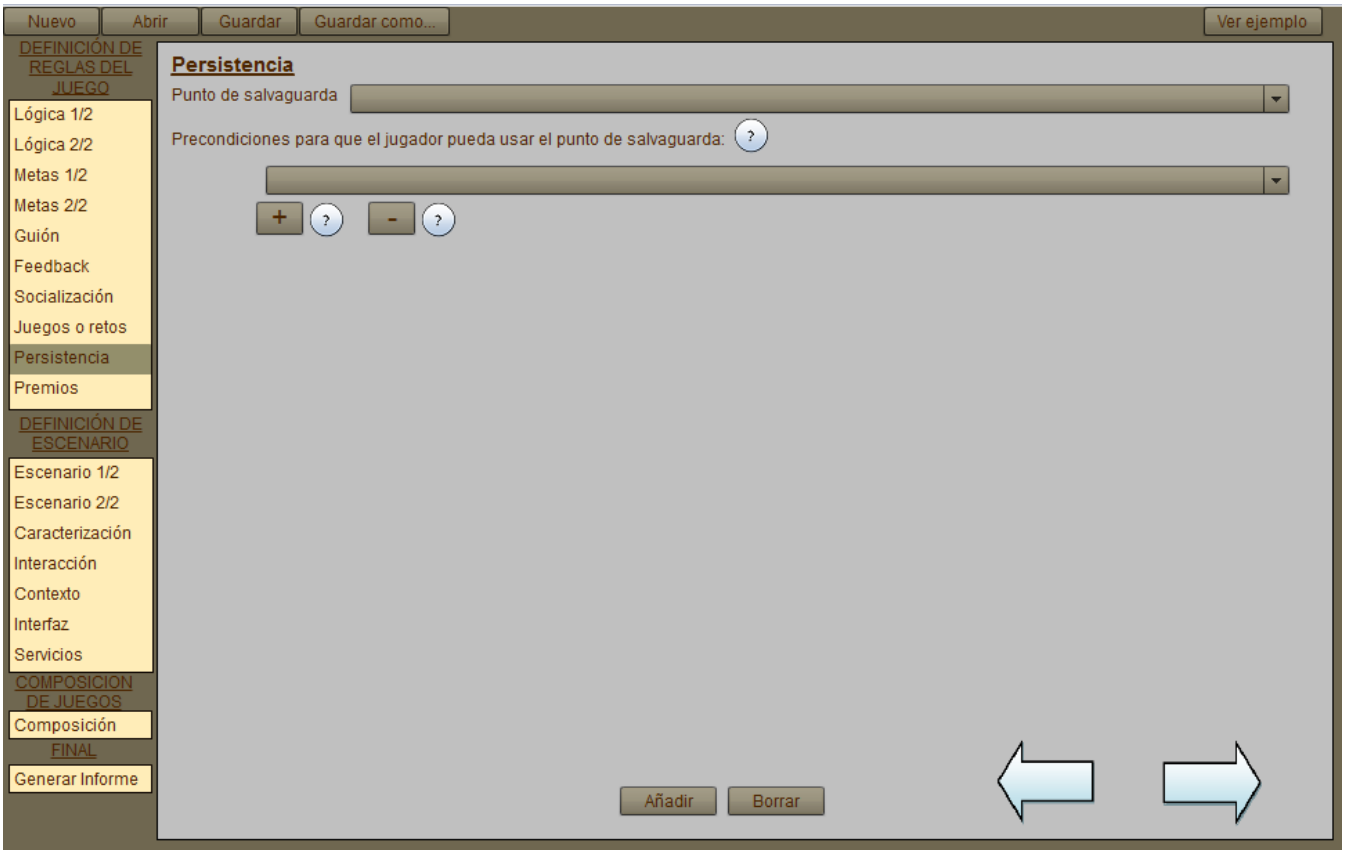

ILUSTRACIÓN 35 – VENTANA PERSISTENCIA

7.6.8. Módulo "Premios"

Diseño del módulo "Premios"

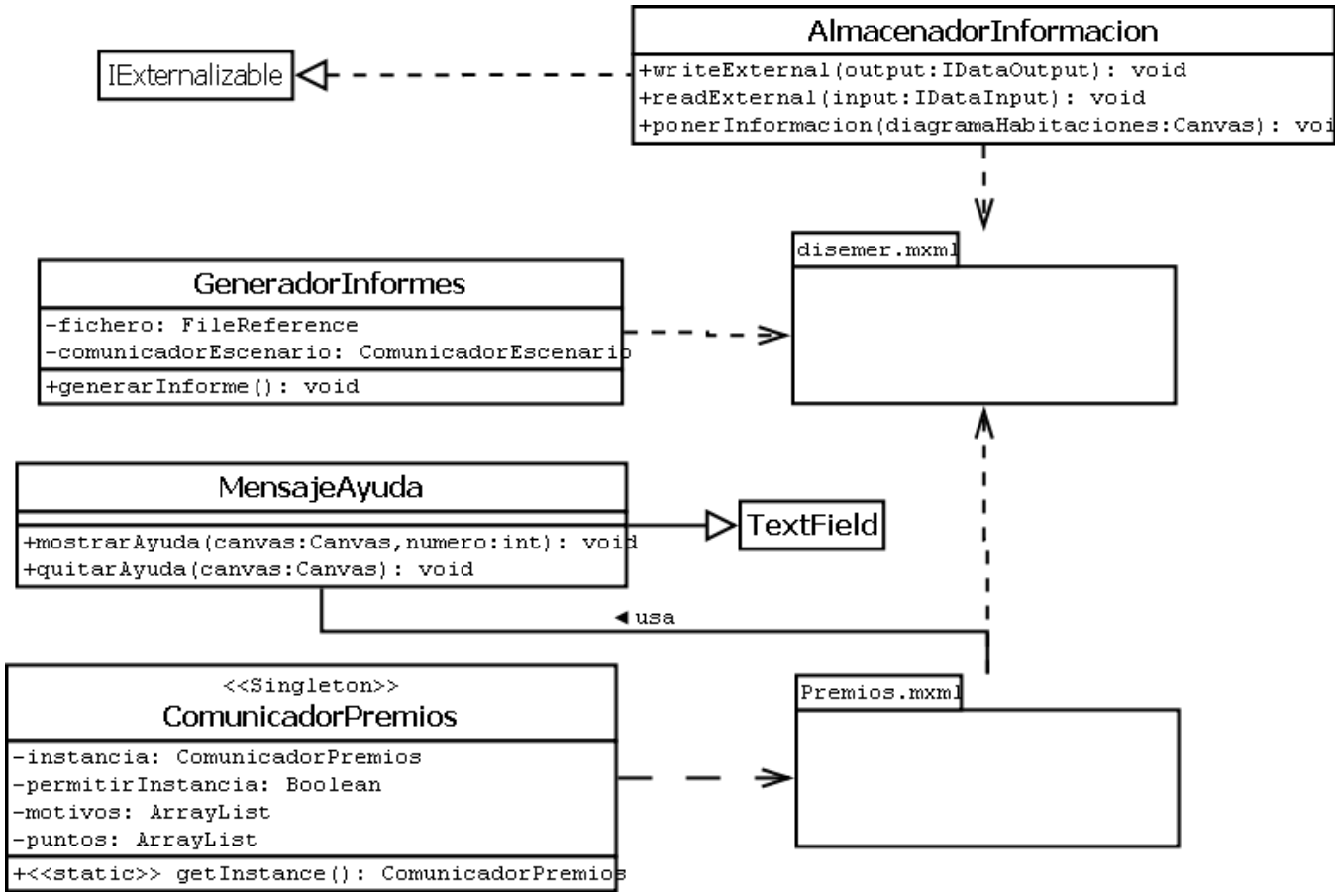

ILUSTRACIÓN 36 - DIAGRAMA DE CLASES DE HU-02

Pruebas del módulo "Premios"

Este será el plan de pruebas para este módulo:

1. Introducir al menos cuatro premios con las cuatro opciones posibles.
2. Pulsar en “Guardar”.
3. Renombrar una entidad que se haya seleccionado para identificar.
4. Volver a la ventana de premios y ver como se ha actualizado el nombre correctamente.
5. Borrar una entidad que se haya seleccionado para identificar.
6. Volver a la ventana de premios y ver como se ha borrado ese motivo de premio y el propio premio correctamente.
7. Realizar los pasos 3-6 para la entidad que se haya seleccionado para coger.
8. Guardar el archivo.
9. Cerrar la aplicación.
10. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
11. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.

12. Generar informe.
13. Comprobar que el informe que se ha generado es correcto.
14. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
15. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Premios"

En esta ventana se pueden introducir los premios que se van a dar a los jugadores de la aplicación. Si se pulsa en motivo, aparecerán estas cuatro opciones:

- Llegar a habitacion final
- Coger una entidad (especificar cuál)
- Identificar una entidad (especificar cuál)
- Otro (especificar cuál)

Si se pulsa en la 2ª, en la 3ª o en la 4ª opción aparecerá un panel para especificar el motivo por el que se da el premio.

El aspecto de la ventana de los premios se muestra en la ilustración "Ventana Premios":

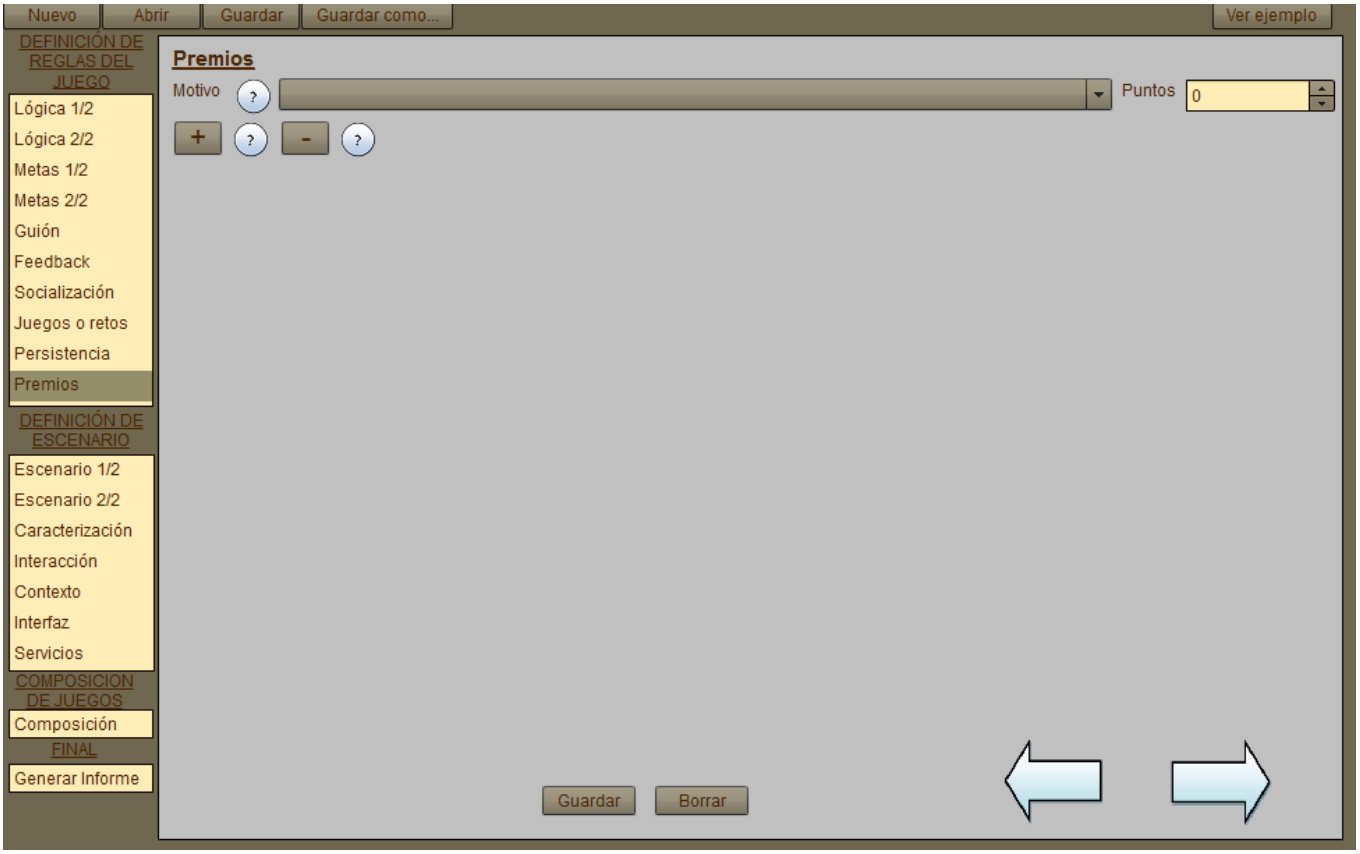

ILUSTRACIÓN 37 – VENTANA PREMIOS

7.7 Módulos de la "Definición del escenario"

7.7.1. Módulo "Escenario"

Diseño del módulo "Escenario"

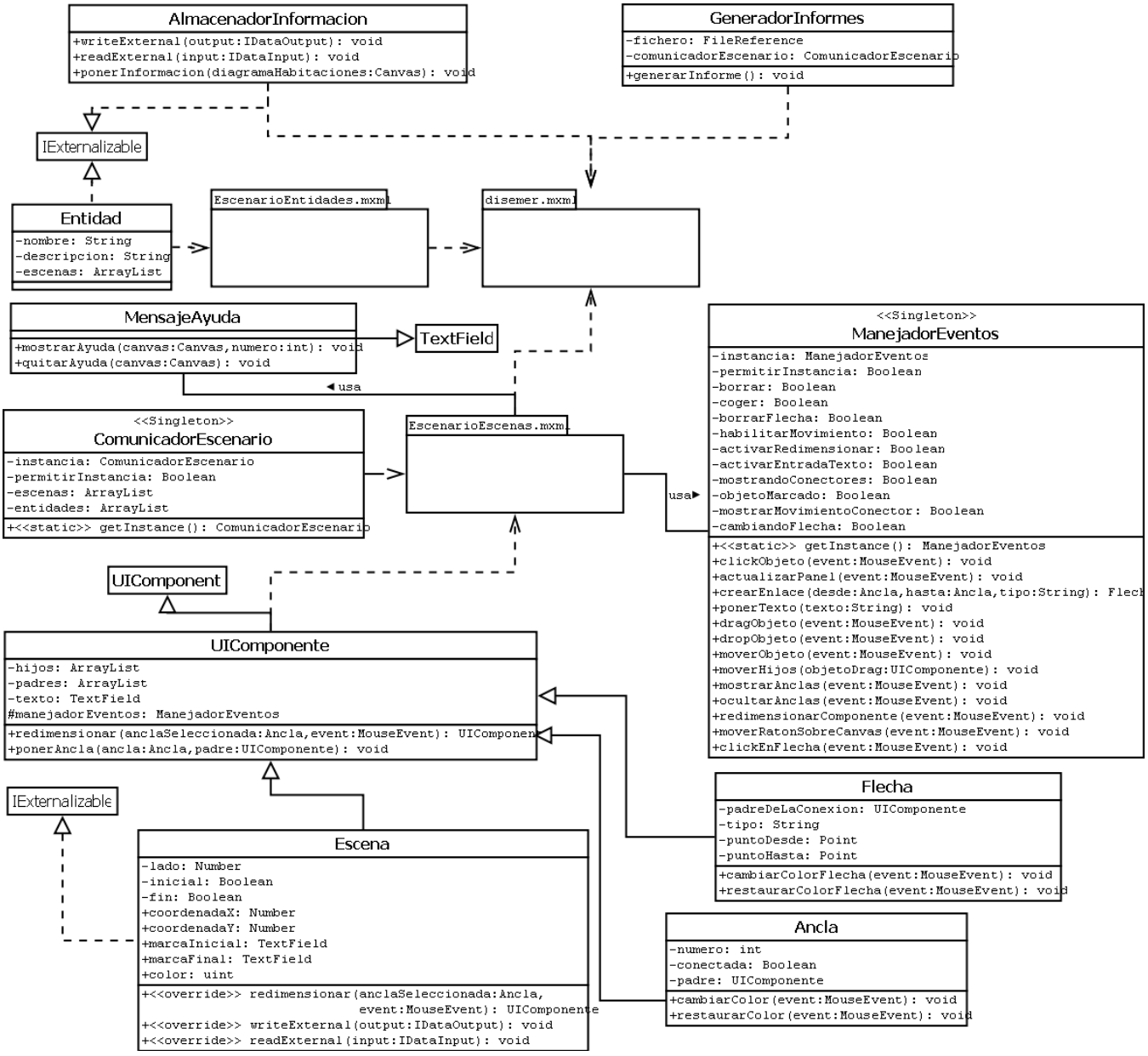

ILUSTRACIÓN 38 - DIAGRAMA DE CLASES DE HU-13

Pruebas del módulo "Escenario"

Este será el plan de pruebas para este módulo:

1. Arrancar la aplicación.
2. Pulsar el icono con forma de flecha derecha.
3. Generar un diagrama de al menos tres escenas con al menos una unión unidireccional y otra bidireccional. Marcar al menos una escena como inicial y otra como final.
4. Comprobar que si se posa el cursor sobre una flecha esta cambia de color.
5. Comprobar que si se posa el cursor sobre un ancla de una escena esta cambia de color.
6. Comprobar que al pulsar en cualquiera de las tres escenas se actualiza su información en el panel de la izquierda.
7. Pulsar el icono con forma de flecha derecha.
8. Crear al menos tres entidades. Al menos una de ella debe estar presente en las tres escenas.
9. Volver a la ventana del diagrama de flujo donde se editaban las relaciones entre escenas. Para ello, habrá que pulsar al menos dos veces el botón con forma de flecha izquierda.
10. Renombrar una escena.
11. Comprobar que se ha actualizado automáticamente este nombre en la ventana de entidades.
12. Borrar una escena.
13. Comprobar que se ha actualizado automáticamente este nombre en la ventana de entidades.
14. Guardar el archivo.
15. Cerrar la aplicación.
16. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
17. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
18. Comprobar que al pulsar los botones “Borrar escena”, “Unión unidireccional”, “Unión bidireccional” o “Borrar unión” se queda pulsado y además si había algún botón pulsada este se desmarca.
19. Comprobar que si se pulsa el botón “Unión unidireccional” y luego se posa el cursor encima del ancla de una escena, el cursor cambia a una flecha con una sola cabeza.
20. Comprobar que si se pulsa el botón “Unión bidireccional” y luego se posa el cursor encima del ancla de una escena, el cursor cambia a una flecha con dos cabezas.
21. Generar informes.
22. Comprobar que los dos informes que se han generado (tanto la captura de pantalla como el informe textual) son correctos.
23. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
24. Comprobar que la navegabilidad entre las dos ventanas y sus ventanas correspondientes de ayuda es la correcta y los mensajes que se muestran en dichas ventanas de ayuda son los que se desean.

Implementación del módulo "Escenario"

En el caso de este módulo, se va a explicar mejor las funcionalidades que permite realizar a partir de la ventana implementada mostrada en la ilustración "Ventana Escenario 1/2" para favorecer su comprensión:

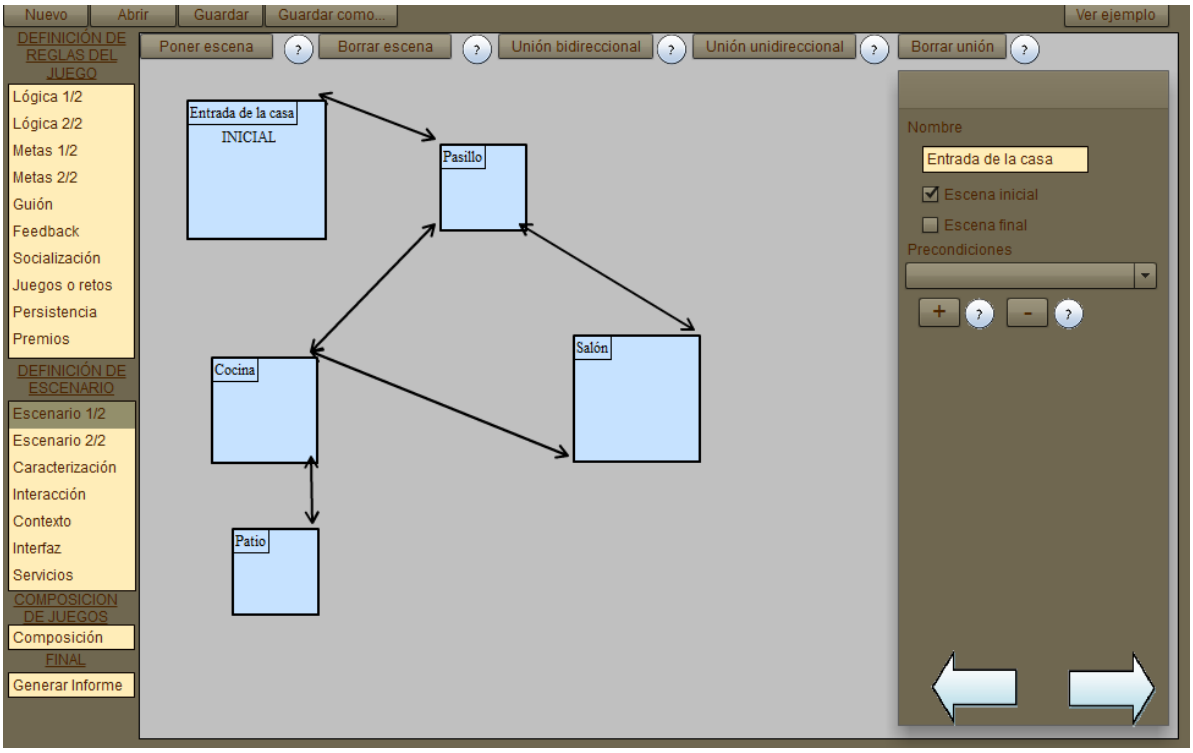

ILUSTRACIÓN 39 - VENTANA ESCENARIO 1/2

Como se puede ver, esta ventana está totalmente relacionada con la spike solution elaborada y su motor es el mismo con una serie de mejoras aplicadas. Por ejemplo, si se pulsa en la habitación “Entrada” el panel de la derecha se actualiza automáticamente con el nombre de esa escena y activa un “Check box” si la escena es la inicial y otro si la escena es la final.

Además, flechas y anclas cambian de color cuando el cursor se posa encima de ellas. Todos los botones excepto el de "Poner escena" se quedarán pulsados una vez se pulsen y cuando se pulse cualquier otro volverá a su estado normal. Además, si se intenta crear una unión bidireccional o una unidireccional y acto seguido se posa el cursor en un ancla, el cursor cambiará a una flecha bidireccional o unidireccional, respectivamente.

Finalmente, es posible introducir precondiciones para poder acceder a una escena pulsando en el desplegable de la derecha.

Mientras que en la segunda ventana del escenario se permite relacionar las entidades que se han creado en la primera parte del módulo "Lógica" con las escenas que se acaban de definir.

Se permite que una entidad aparezca en más de un escena y que además este presente más de una vez en las escenas.

Si se vuelve atrás y se cambian los nombres de las escenas o se borra alguna escena en la que estaba presente alguna entidad creada, esta información se actualizará automáticamente: se cambiará el nombre de la escena en la que está presente en caso del renombrado y la entidad dejará de estar presente en esa escena (pero sí del resto) en caso de que se borrase. En caso de que por borrar una escena la entidad pasara a no estar presente en ninguna otra escena, la entidad simplemente no estará presente en ninguna escena.

La implementación de esta segunda ventana se puede ver en la ilustración "Ventana Escenario 2/2":

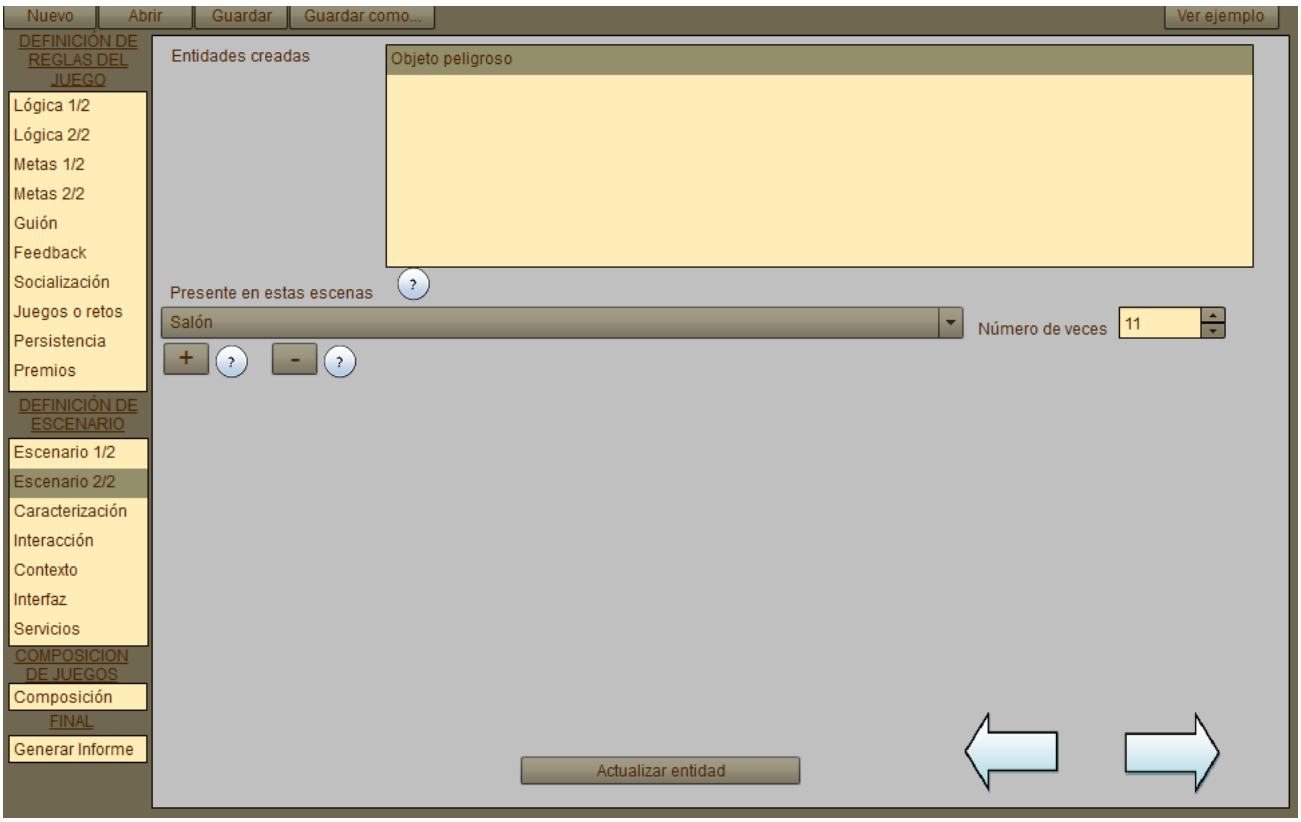

ILUSTRACIÓN 40 - VENTANA ESCENARIO 2/2

7.7.2. Módulo "Caracterización"

Diseño del módulo "Caracterización"

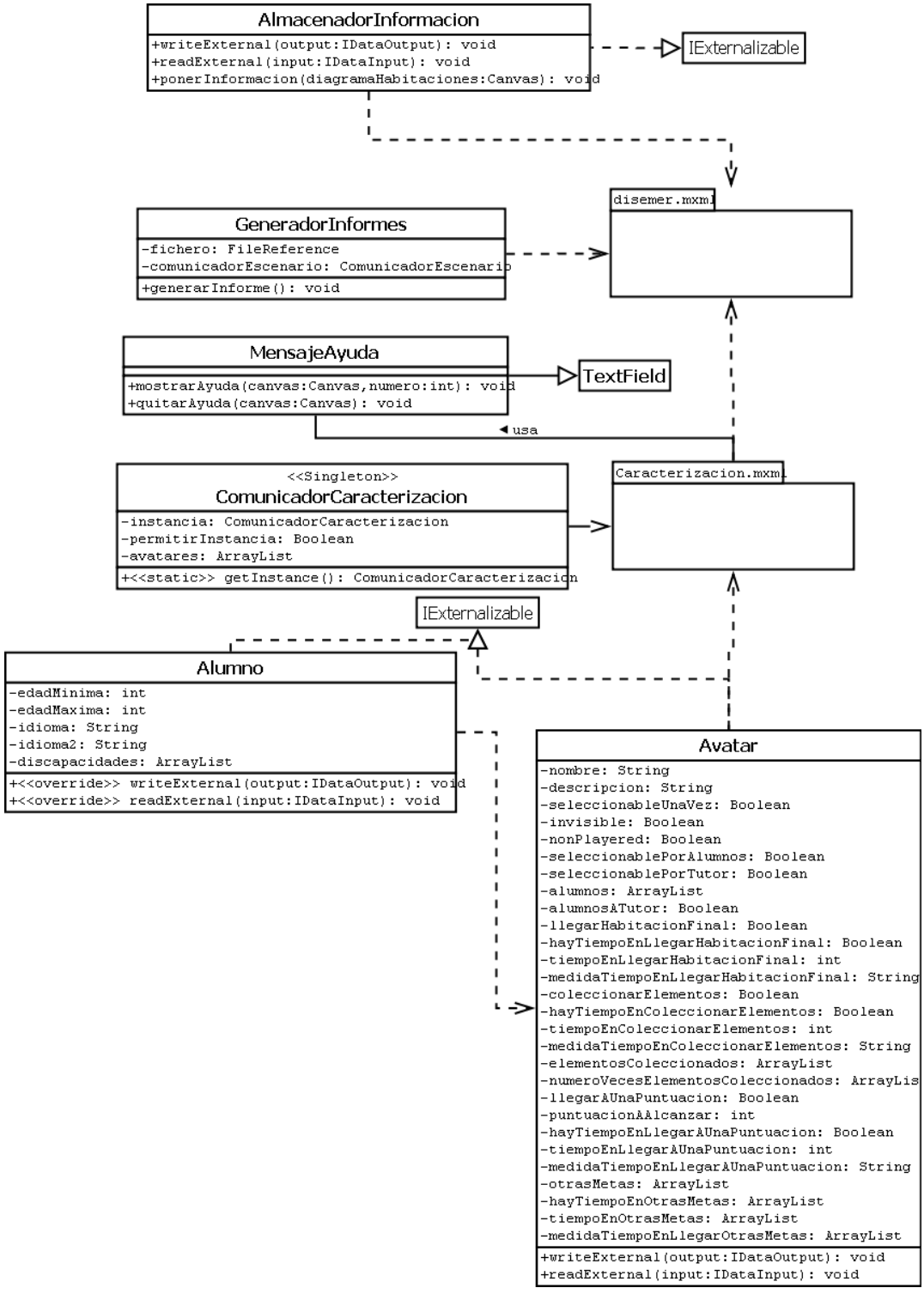

ILUSTRACIÓN 41 - DIAGRAMA DE CLASES DE HU-12 (II)

Pruebas del módulo "Caracterización"

Este será el plan de pruebas para este módulo:

1. Crear un avatar que solo pueda ser usado por un tutor, que solo se pueda seleccionar una vez y que sea invisible para el resto.
2. Crear un avatar que se pueda usar por los alumnos y solo por ellos.
3. Dar un valor de edad mínima, máxima y de idioma.
4. Seleccionar una discapacidad auditiva.
5. Poner otra discapacidad y pulsar la opción “Otra”. Ver que el mensaje mostrado en el panel para especificar es correcto.
6. Repetir esto para Motriz y Mental.
7. Guardar el archivo.
8. Cerrar la aplicación.
9. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
10. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
11. Generar informe.
12. Comprobar que el informe que se ha generado es correcto.
13. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
14. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Como este módulo está fuertemente relacionado con los módulos de "Lógica" y "Metas" se va a hacer otro plan de pruebas adicional. Es necesario que antes de realizar este plan de pruebas se realice el de los dos módulos anteriormente relacionados:

1. Marcar las opciones “Tutor” y “Alumnos”. Después marcar la opción “Avatar controlado por CPU”. Ver como se desmarcan las dos primeras.
2. Añadir un avatar controlado por ordenador. Ver como aparece en “Escenario (2/2)” y “Lógica” como si de una entidad se tratase.
3. Ir a la parte de “Lógica” y ver que se puede dotar de estados al avatar controlado por ordenador como si de una entidad se tratase.
4. Añadir otro avatar, con datos arbitrarios.
5. En “Metas”, asignar metas arbitrarias a todos y otras metas arbitrarias al último avatar en concreto.
6. En “Metas”, cambiar la pestaña desde “Todos los jugadores” a ese avatar. Ver que las metas establecidas para ambos casos permanecen tal y como se pusieron.
7. Ir a la parte de “Feedback” y seleccionar como motivo de un mensaje que se cumpla una meta de todos los jugadores. Lo mismo para una meta (cualquiera) de un avatar en concreto.
8. Borrar la meta seleccionada de todos los jugadores. Ver que se actualiza correctamente el Feedback.
9. Realizar el paso “7” para la meta del avatar.
10. Realizar los pasos “6”-“8” para la Socialización.
11. Guardar el archivo.
12. Cerrar la aplicación.
13. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
14. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.

15. Generar informe.
16. Comprobar que el informe que se ha generado es correcto.
17. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
18. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Caracterización"

En este módulo, se van a permitir definir los avatares que los jugadores van a poder seleccionar. El educador deberá incluir una descripción en la que se pide la descripción textual de qué apariencia tendrá ese avatar para que los programadores sepan cómo quiere el educador que sea gráficamente.

Además, se permite definir cuatro condiciones para el avatar:

- Que lo puedan seleccionar educadores
- Que lo puedan seleccionar alumnos
- Que solo se pueda seleccionar una vez
- Que el avatar sea invisible

Si se marca la opción de que lo puedan seleccionar los alumnos, el educador tendrá que definir para qué alumnos estará preparado este avatar: su edad mínima y máxima, su idioma y si el jugador tiene algún tipo de discapacidad.

Por último, en el módulo se permiten definir avatares que estén controlados automáticamente por ordenador. En ese caso, el educador tendrá que volver al módulo "Lógica" para definir su comportamiento y al módulo "Escenario" para definir dónde está ubicado.

La apariencia de la ventana de caracterización se puede ver en la ilustración "Ventana Caracterización":

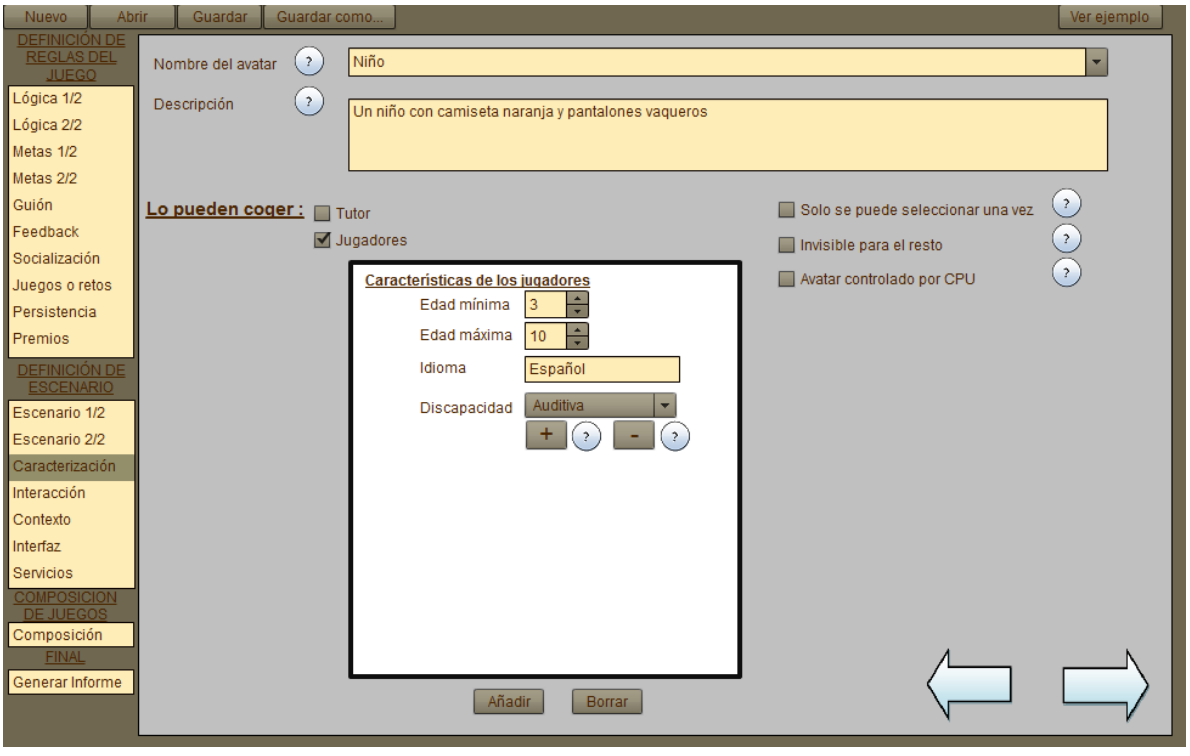

ILUSTRACIÓN 42 – VENTANA CARACTERIZACIÓN

7.7.3. Módulo "Interacción"

Diseño del módulo "Interacción"

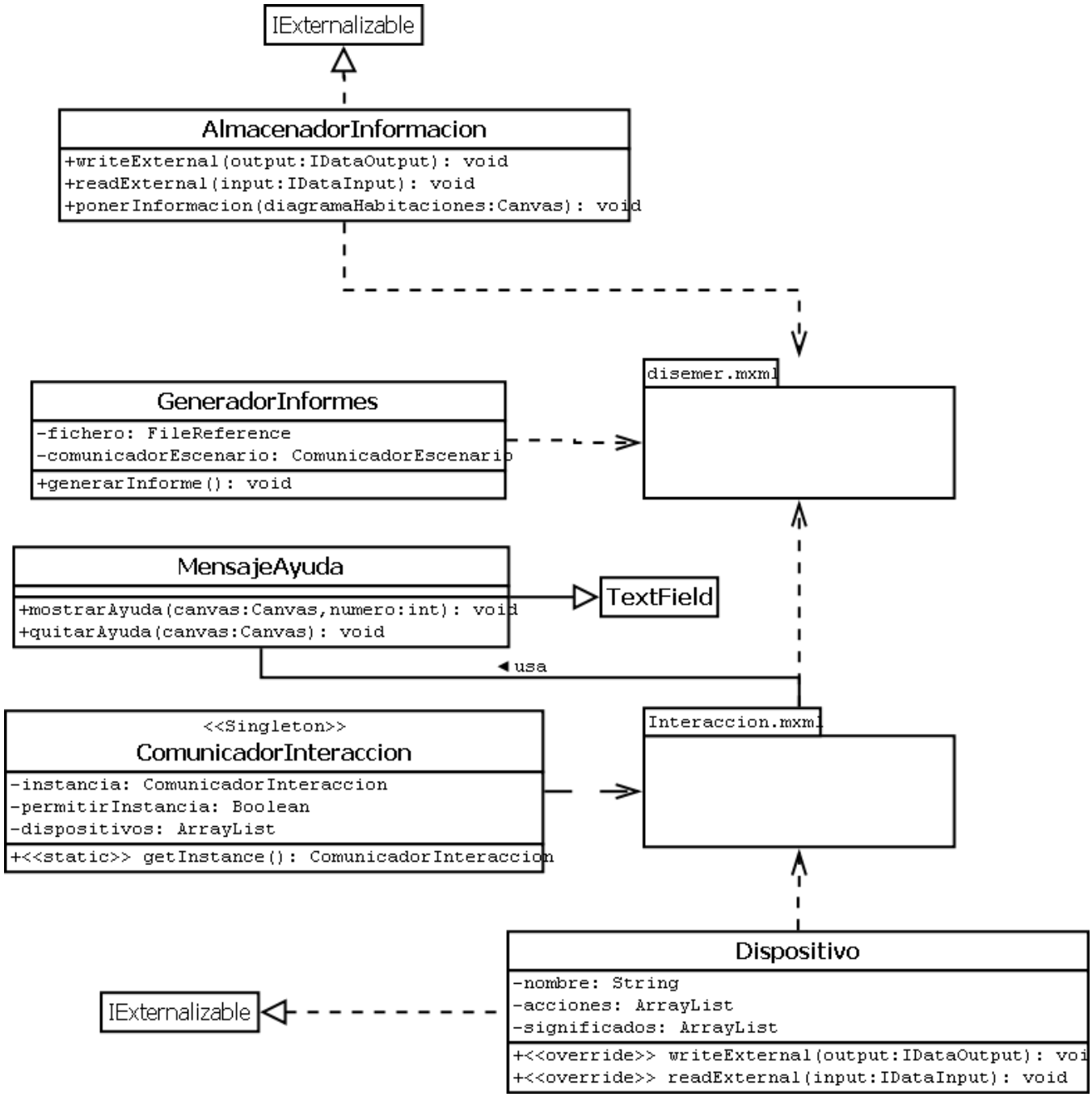

ILUSTRACIÓN 43 - DIAGRAMA DE CLASES DE HU-11

Pruebas del módulo "Interacción"

Este será el plan de pruebas para este módulo:

1. Dar valor a al menos tres acciones de cada dispositivo.
2. Crear un nuevo dispositivo y darle al menos dos acciones.
3. Guardar el archivo.
4. Cerrar la aplicación.
5. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
6. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
7. Generar informe.
8. Comprobar que el informe que se ha generado es correcto.
9. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
10. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Interacción"

En este módulo se permiten definir los dispositivos con los que se va a poder jugar al videojuego y las acciones que se pueden realizar con ellos. Se ha incluido estos tres dispositivos establecidos por defecto:

- Ratón
- Teclado
- Mando

Para cada dispositivo se puede introducir información sobre las acciones que se podrán llevar a cabo con él y el significado o el efecto que tendrá en el juego realizar dicha acción. En función del dispositivo se pueden elegir unas opciones u otras. Si se crea un dispositivo se pueden definir hasta 7 acciones. El educador podrá elegir una de estas opciones:

- Click
- Doble-click
- Click con el botón derecho
- Drag & Drop
- Mover el dispositivo
- Pulsar un botón (si es un mando)
- Pulsar una tecla (si es un teclado)

Si se seleccionara el Ratón las opciones son:

- Click
- Doble-click
- Click con el botón derecho
- Drag & Drop
- Mover el dispositivo

Mientras que si se seleccionara el Teclado las opciones son:

- Pulsar una tecla
- Dejar pulsada una tecla

Finalmente, si se selecciona el Mando las opciones son:

- Mover el dispositivo
- Pulsar un botón
- Dejar pulsado un botón

Para cualquier dispositivo, el educador podrá añadir las acciones adicionales que él quiera.

El aspecto de la ventana de interacción se puede ver en la ilustración "Ventana Interacción":

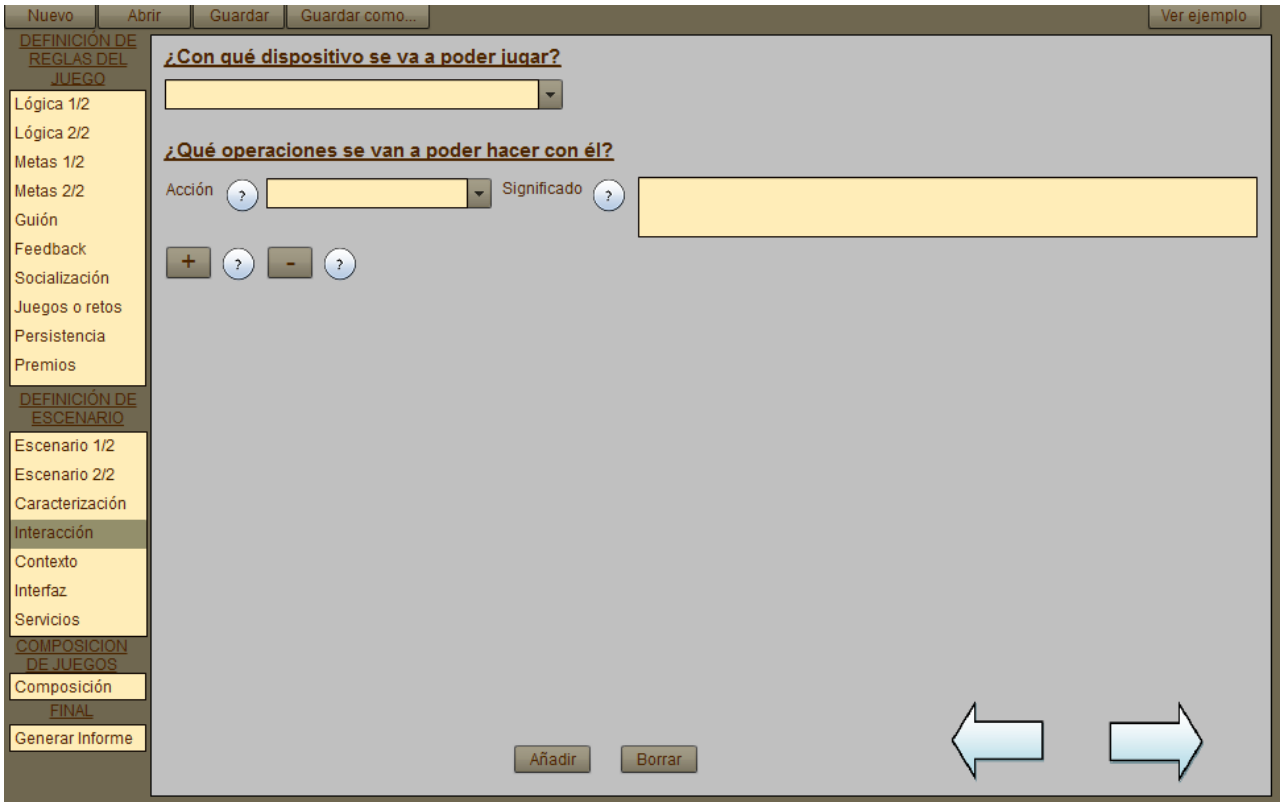

ILUSTRACIÓN 44 – VENTANA INTERACCIÓN

7.7.4. Módulo "Contexto"

Diseño del módulo "Contexto"

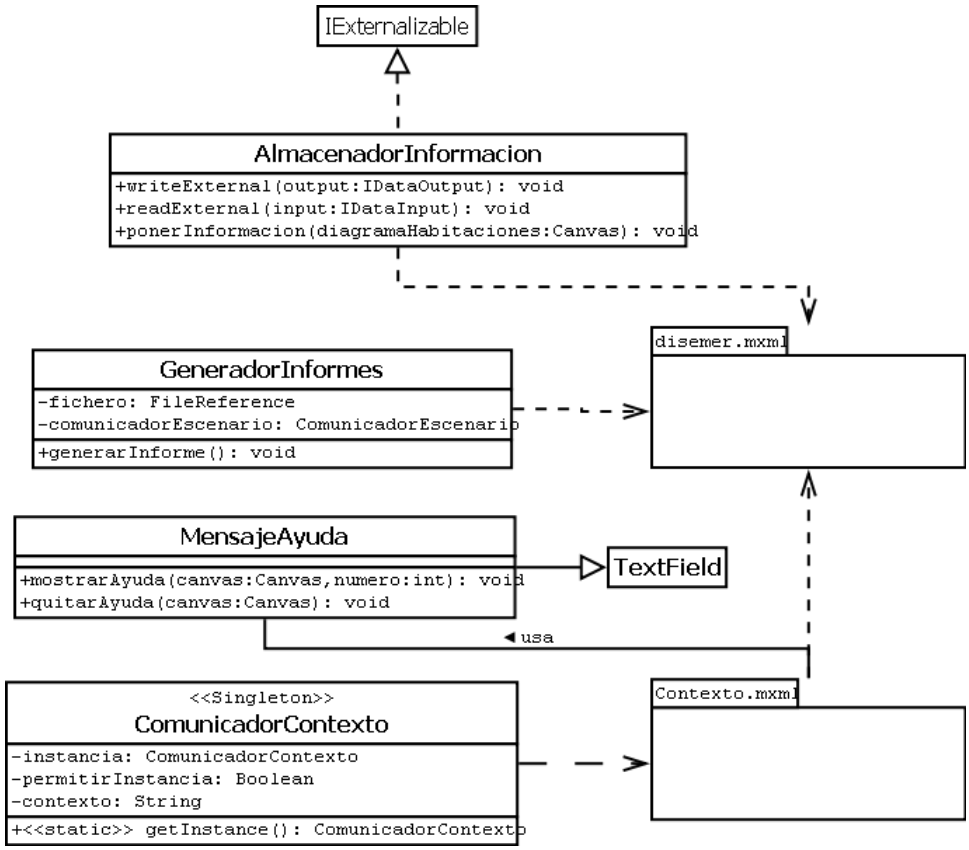

ILUSTRACIÓN 45 - DIAGRAMA DE CLASES DE HU-14

Pruebas del módulo "Contexto"

Este será el plan de pruebas para este módulo:

1. Dar un valor al contexto
2. Guardar el archivo.
3. Cerrar la aplicación.
4. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
5. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
6. Generar informes.
7. Comprobar que el informe que se ha generado es correcto.
8. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
9. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Contexto"

Este módulo es muy sencillo, ya que consiste en el que el educador pueda poner el entorno en el que se va a desarrollar el juego o cualquier tipo de información que pueda ser relevante a la hora de crearlo.

El aspecto de la ventana de contexto se puede ver en la ilustración "Ventana Contexto":

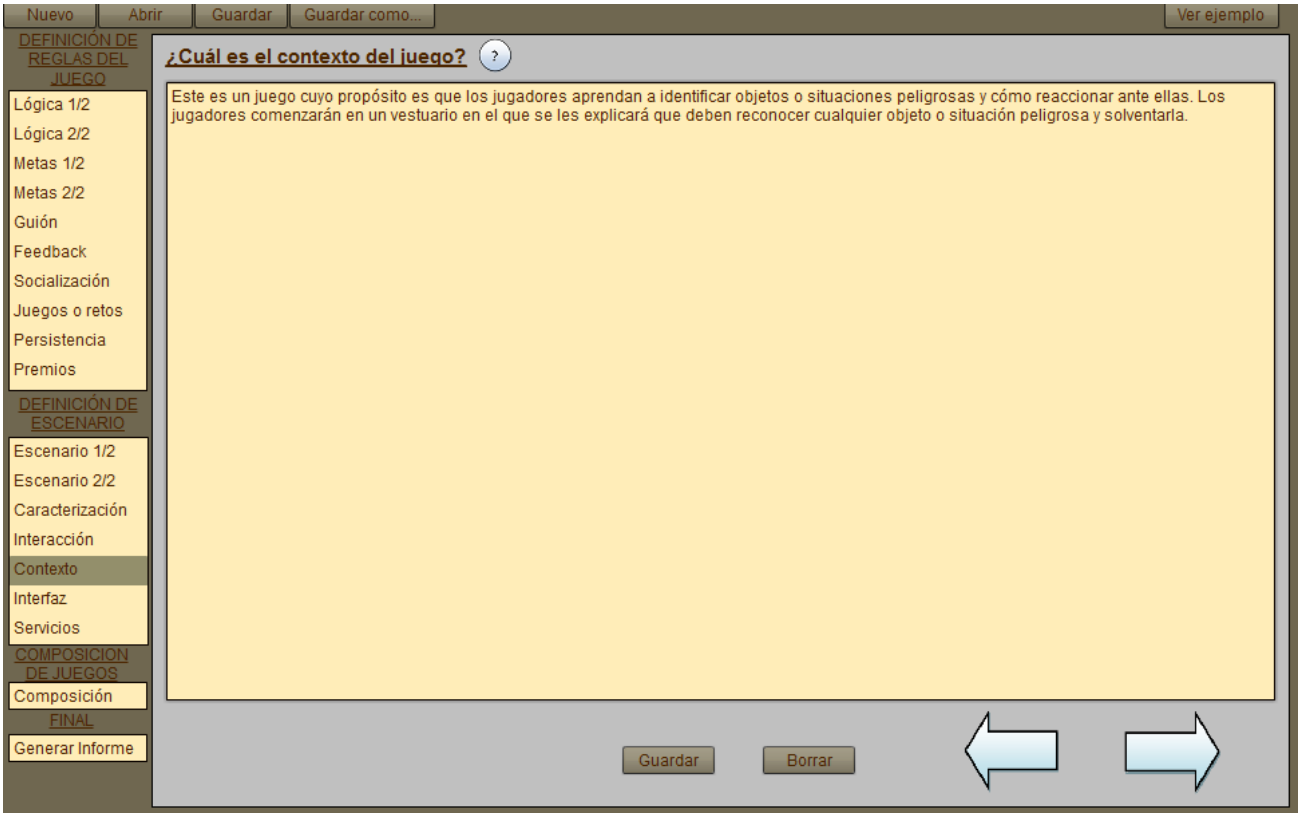

ILUSTRACIÓN 46 – VENTANA CONTEXTO

7.7.5. Módulo "Interfaz"

Diseño del módulo "Interfaz"

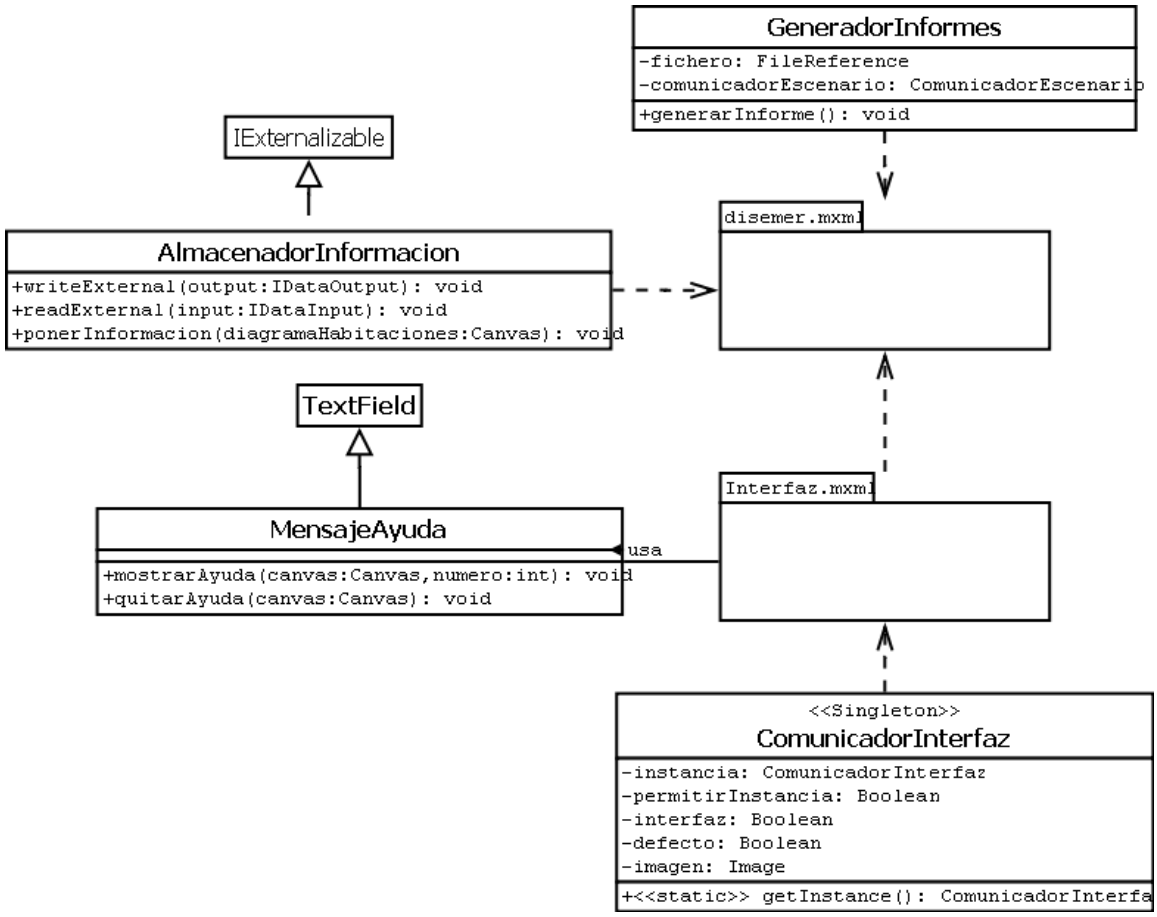

ILUSTRACIÓN 47 - DIAGRAMA DE CLASES DE HU-17

Pruebas del módulo "Interfaz"

El plan de pruebas de la historia de usuario “HU-17” es el siguiente:

1. Marcar el “Radio button” “Sí quiero una interfaz”. Ver que aparece la interfaz por defecto y dos “Radio button” más.
2. Marcar el “Radio button” “Quiero usar una imagen propia para usarla como interfaz”. Ver que aparece un icono de ayuda y un botón más.
3. Tratar de cargar un archivo que no es una imagen. Ver que sale error.
4. Tratar de cargar una imagen. Ver que se cambia la imagen por defecto por la imagen seleccionada.
5. Marcar el “Radio button” “No quiero ninguna interfaz que acompañe el juego”. Ver que desaparece todo.
6. Volver a cargar una imagen propia.
7. Guardar el archivo.
8. Cerrar la aplicación.
9. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.

10. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
11. Generar informe.
12. Comprobar que el informe que se ha generado es correcto.
13. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
14. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Interfaz"

En este módulo, se permite definir la interfaz que va a tener el juego. El educador tendrá que elegir entre si quiere o no interfaz. Si quiere una interfaz, se ha implementado una por defecto con la mayoría de controles necesarios, pero si el docente quiere una propia, la aplicación permite la subida de imágenes, por lo que el educador podrá subir una imagen que haya realizado con otra aplicación.

Si el educador intentara subir un archivo que no sea una imagen, la aplicación mostraría un mensaje de error.

El aspecto de la ventana de interfaz se puede ver en la ilustración "Ventana Interfaz":

ILUSTRACIÓN 48 – VENTANA INTERFAZ

7.7.6. Módulo "Servicios"

Diseño del módulo "Servicios"

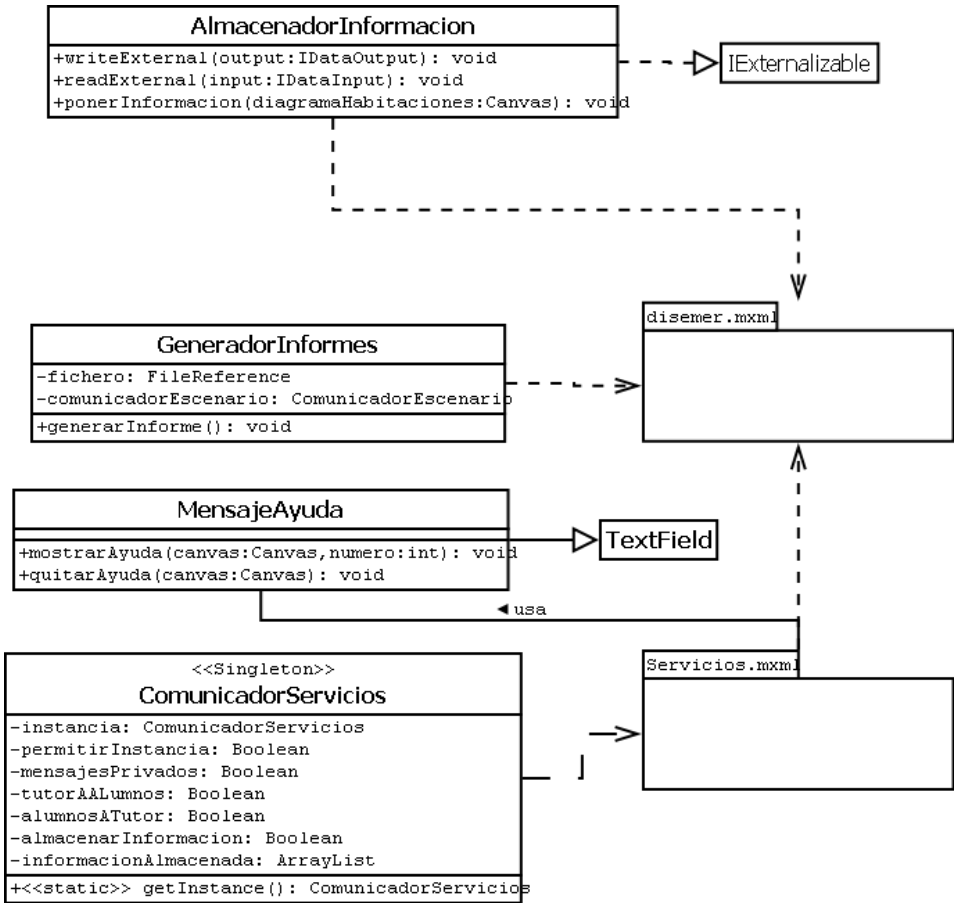

ILUSTRACIÓN 49 - DIAGRAMA DE CLASES DE HU-10

Pruebas del módulo "Servicios"

Este será el plan de pruebas para este módulo:

1. Seleccionar al menos una de los tres primeros “Check Box”.
2. Seleccionar el “Check Box” de almacenar información. Observar como aparece el nuevo panel.
3. Marcar tantos tipos de informaciones distintas como se desee, pero al menos en uno de ellos se debe pulsar la información “Otra (especificar cuál)”.
4. Repetir pasos 2-3 para compartir información.
5. Guardar el archivo.
6. Cerrar la aplicación.
7. Cargar el archivo. Comprobar que todo permanece en el mismo estado en el que estaba en el instante anterior a guardarlo.
8. Volver a cargar el archivo. Comprobar que la información se ha cargado correctamente y no ha salido información duplicada.
9. Generar informes.
10. Comprobar que el informe que se ha generado es correcto.

11. Comprobar que todos los mensajes de ayuda están colocados correctamente y dicen el mensaje que se desea.
12. Comprobar que la ventana auxiliar tiene el mensaje que se desea.

Implementación del módulo "Servicios"

En este módulo se pueden marcar los servicios que se quieren proveer:

- Chat
- Comunicarse por voz
- Comunicarse por vídeo
- Almacenar información
- Compartir información

Si el educador marca los dos últimos para almacenar y/o compartir información aparecerá un panel como en el caso de las discapacidades en la parte de caracterización. En estos paneles podrá marcar hasta 8 tipos de información distintas. Las opciones son:

- Puntuaciones de cada jugador
- Tiempo que se tardó en finalizar la partida
- Número de objetivos cumplidos
- Número de objetivos incumplidos
- Porcentaje de objetivos cumplidos
- Porcentaje de objetivos incumplidos
- Jugadores que se eliminaron de la partida
- Otro (especificar cuál)

El aspecto de la ventana de servicios se puede ver en la ilustración "Ventana Servicios":

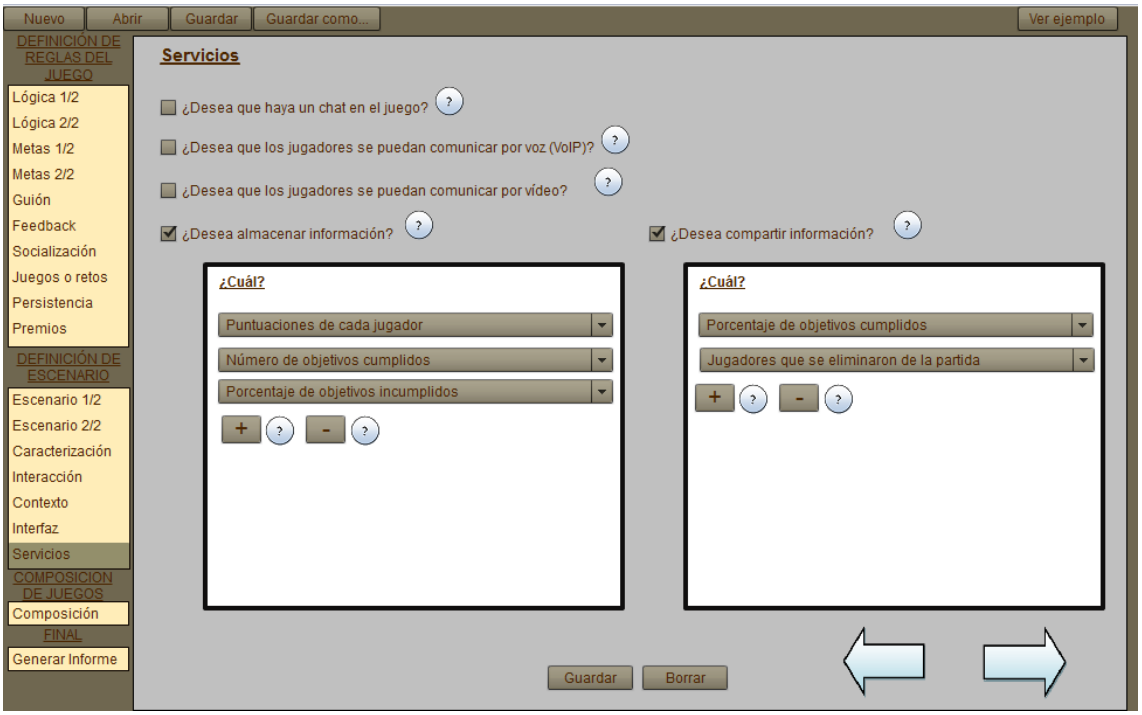

ILUSTRACIÓN 50 – VENTANA SERVICIOS

7.8 Módulo "Composición"

7.8.1. Diseño del módulo "Composición"

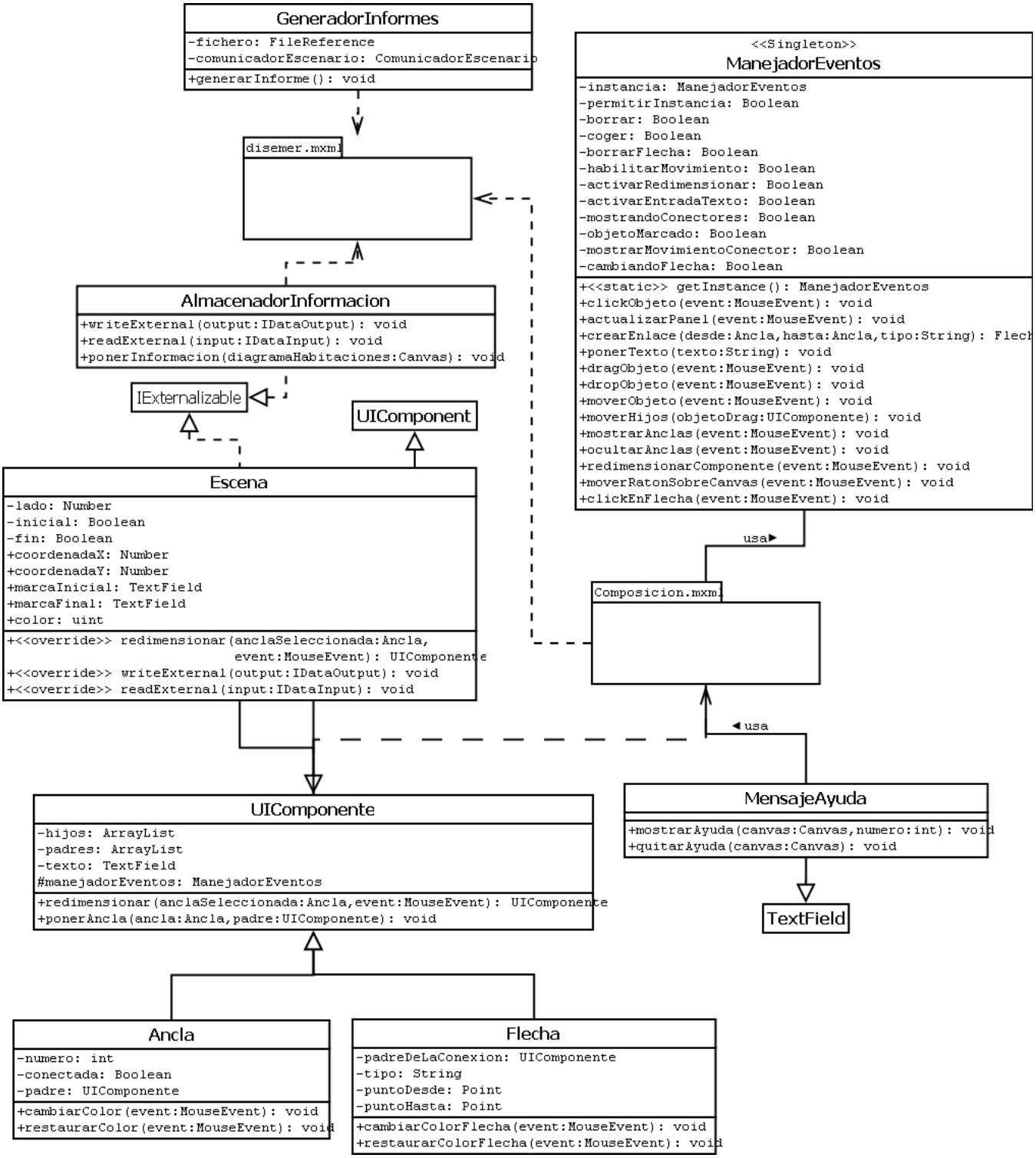

ILUSTRACIÓN 51 - DIAGRAMA DE CLASES DE HU-16

7.8.2. Pruebas del módulo "Composición"

El plan de pruebas de este módulo será el mismo que el del módulo "Escenario", con estos pasos añadidos:

1. Probar a cargar un juego correcto. Ver que se añade correctamente.
2. Probar a cargar un juego incorrecto. Ver que se muestra un mensaje de error.

7.8.3. Implementación del módulo "Composición"

Al igual que en otros casos, se va a explicar la funcionalidad de este módulo apoyándose en la implementación de la ventana mostrada en la ilustración "Ventana Composición" para así favorecer la comprensión de dicho módulo:

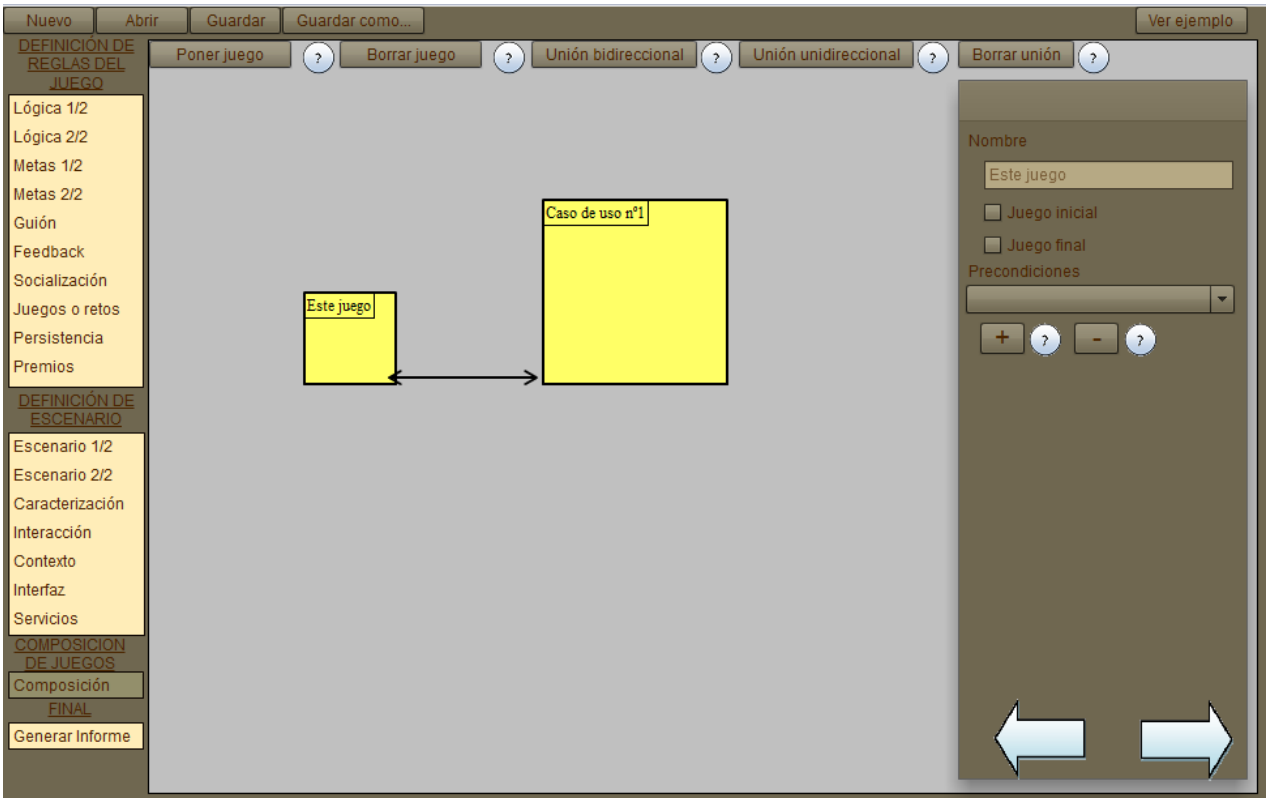

ILUSTRACIÓN 52 – VENTANA COMPOSICIÓN

Como se puede ver, para llevar este módulo a cabo se ha reutilizado el motor implementado para el módulo "Escenario" y se han realizado algunos cambios:

1. Se ha cambiado el color de los cuadrados a amarillo, ya en lugar de representar escenas representan en este caso juegos.
2. No se permite cambiar el nombre ningún juego. Al juego que se está diseñando en este momento se le llama automáticamente “Este juego” y ya aparece cargado por defecto y los juegos que se carguen tendrán el mismo nombre que sus archivos.
3. La función del botón “Poner juego” se ha cambiado. En lugar de poner automáticamente un cuadrado en la esquina inferior izquierda del canvas, se abre un explorador de archivos para que el educador cargue un juego previamente diseñado. Si ese archivo era exactamente un diseño de un juego, se pondrá el cuadrado que represente al juego. Si no, se muestra un error.

8. EVALUACIÓN

Una vez se finalizó la implementación de la aplicación, era necesario llevar a cabo una evaluación para comprobar que dicha aplicación cumplía todos los requisitos establecidos y en caso negativo, realizar las modificaciones oportunas.

Se plantearon las siguientes posibilidades:

- Prueba con educadores
- Prueba con usuarios expertos
- Casos de uso

Y se decidió la opción de los casos de uso, ya que era la alternativa que conseguía un equilibrio entre el gasto de recursos y la rigurosidad de la evaluación, ya que si se diseñaban casos de usos lo suficientemente variados se podía llegar a evaluar el sistema a la perfección. Se decidió la siguiente planificación para realizar los casos de uso:

- Tres casos de uso sencillos (casos de uso nº1, nº2 y nº3).
- Tres casos de uso más complicados, donde dos de ellos sean composiciones de los tres anteriores (casos de uso nº4, n5 y nº6).

8.1 Primer caso de uso: juego de identificar objetos peligrosos

	Descripción
Propósito	Que los jugadores identifiquen objetos peligrosos
Avatares	1. Un niño 2. Un Helper (avatar controlado por el ordenador) que dará consejos a los jugadores hasta un número máximo de veces. Además no podrá dar dos consejos consecutivos si no han pasado un tiempo determinado.
Interacción	1. Mover 2. Identificar objeto
Entidades	Objeto peligroso con dos estados: {Identificado, No identificado}
Metas	1. Identificar un número determinado de entidades (con límite de tiempo) 2. Mover al avatar a una posición en concreto (con límite de tiempo)
Escenas	El salón de una casa.
Feedback	1. Descripción de las metas (al comenzar la partida). Tipo: Texto. 2. Recomendaciones de las metas (al comenzar la partida). Tipo: Texto. 3. Descripción del peligro (al comenzar la partida). Tipo: Texto. 4. Descripción de la localización del peligro (al comenzar la partida). Tipo: Texto. 5. Flechas apuntando al peligro (si el jugador tarda mucho en completar la primera meta). Tipo: Imagen.

TABLA 7 - CASO DE USO Nº 1

8.2 Segundo caso de uso: juego de recolección de objetos

	Descripción
Propósito	Que los jugadores recojan objetos y cuando hayan recogido un número determinado de objetos que representan a basura y construyan edificios.
Avatares	Habrá un único avatar disponible, que tendrá la apariencia de un niño
Interacción	1. Mover 2. Recoger objeto 3. Disparar 4. Construir
Entidades	1. Objeto para recoger con dos estados: {Recolectado, No recolectado} 2. Pistola con la que disparar bolas de nieve 3. Bolas de nieve que podrá disparar el jugador con una pistola 4. Edificio que se puede construir cuando se haya coleccionado un número determinado de objetos que representen a basura.
Metas	1. Recoger un número determinado de objetos que representen basura 2. Construir un número determinado de edificios
Escenas	Una escena.
Feedback	1. Descripción de las metas (al comenzar la partida). Tipo: Texto. 2. Recomendaciones de las metas (al comenzar la partida). Tipo: Texto. 3. Descripción de la localización de la basura (al comenzar la partida). Tipo: Texto. 4. Flechas apuntando a la basura (si el jugador tarda mucho en completar la primera meta). Tipo: Imagen.

TABLA 8 – CASO DE USO Nº 2

8.3 Tercer caso de uso: juego en el que se tenga que resolver un cuestionario

	Descripción
Propósito	Que los jugadores tengan que resolver un cuestionario
Avatares	Habrá un único avatar disponible, que tendrá la apariencia de un niño
Interacción	1. Mover 2. Tocar objeto
Entidades	1. Objeto que al pulsarlo aparece un cuestionario que el jugador debe resolver.
Metas	1. Resolver el cuestionario correctamente
Escenas	Una escena.
Feedback	1. Descripción de la meta (al comenzar la partida). Tipo: Texto. 2. Aviso de que ha superado correctamente el cuestionario (si supera correctamente el cuestionario). Tipo: Texto. 3. Aviso de que no ha superado correctamente el cuestionario (si no supera correctamente el cuestionario). Tipo: Texto.

TABLA 9 – CASO DE USO Nº 3

8.4 Cuarto caso de uso: composición de los tres casos de uso anteriores

	Descripción
Propósito	Que los jugadores tengan que realizar las tareas de los tres primeros casos de uso.
Avatares	Habrá un único avatar disponible, que tendrá la apariencia de un niño
Interacción	1. Mover 2. Identificar objeto 3. Recoger objeto 4. Tocar objeto 5. Disparar 6. Construir
Entidades	1. Objeto peligroso con dos estados: {Identificado, No identificado} 2. Objeto para recoger con dos estados: {Recolectado, No recolectado} 3. Pistola con la que disparar bolas de nieve 4. Bolas de nieve que podrá disparar el jugador con una pistola 5. Edificio que se puede construir cuando se haya coleccionado un número determinado de objetos que representen a basura. 6. Objeto que al pulsarlo aparece un cuestionario que el jugador debe resolver.
Metas	1. Identificar un número determinado de entidades (con límite de tiempo) 2. Mover al avatar a una posición en concreto (con límite de tiempo) 3. Recoger un número determinado de objetos que representen basura 4. Construir un número determinado de edificios 5. Resolver el cuestionario correctamente
Escenas y guión	En el juego habrá tres escenas: una explanada verde (donde habrá que cumplir las metas "3" y "4") desde la que se puede acceder a dos casas: en la primera habrá que cumplir las metas "1" y "2" y en la segunda la meta "5".
Contexto	El juego se desarrollará en una gran explanada verde en la que se estará la basura para recoger y donde se podrán construir los edificios y además donde estarán las dos casas en las que habrá que identificar objetos peligrosos y en la segunda resolver el cuestionario.
Feedback	1. Descripción de las meta (al comenzar la partida). Tipo: Texto. 2. Explicación del escenario (al comenzar la partida). Tipo: Texto. 3. Descripción de la localización del peligro (al entrar en la primera casa). Tipo: Texto. 4. Flechas apuntando al peligro (si el jugador tarda mucho en completar la meta "1"). Tipo: Imagen. 5. Descripción de la localización de la basura (al comenzar la partida). Tipo: Texto. 6. Flechas apuntando a la basura (si el jugador tarda mucho en completar la meta "3"). Tipo: Imagen. 7. Aviso de que ha superado correctamente el cuestionario (si supera correctamente el cuestionario). Tipo: Texto. 8. Aviso de que no ha superado correctamente el cuestionario (si no supera correctamente el cuestionario). Tipo: Texto.
Composición	Con los casos de uso nº1, nº2 y nº3

TABLA 10 – CASO DE USO Nº 4

8.5 Quinto caso de uso: juego más detallado de identificar objetos peligrosos

	Descripción
Propósito	Que los jugadores tengan que identificar objetos peligrosos
Avatares	1. Niño 2. Un Helper (avatar controlado por el ordenador) que dará consejos a los jugadores hasta un número máximo de veces. Además no podrá dar dos consejos consecutivos si no han pasado un tiempo determinado. 3. Ladrón
Interacción	1. Mover 2. Identificar objeto
Entidades	1. Objeto peligroso con dos estados: {Identificado, No identificado}
Metas	1. Identificar un número determinado de entidades (con límite de tiempo) 2. Mover al avatar a una posición en concreto (con límite de tiempo)
Escenas y guión	1. Entrada de la casa (Conectada con: 2) 2. Pasillo (Conectada con: 3, 4) 3. Salón (Conectada con: 2, 4) 4. Cocina (Conectada con: 3, 5) 5. Patio (Conectada con: 4)
Contexto	El juego se desarrollará en una casa en la que los jugadores tendrán que ir recogiendo objetos peligrosos mientras un educador con el avatar de ladrón va introduciendo más objetos peligrosos.
Feedback	1. Descripción de las metas (al comenzar la partida). Tipo: Texto. 2. Recomendaciones de las metas (al comenzar la partida). Tipo: Texto. 3. Descripción del peligro (al comenzar la partida). Tipo: Texto. 4. Descripción de la localización del peligro (al comenzar la partida). Tipo: Texto. 5. Flechas apuntando al peligro (si el jugador tarda mucho en completar la primera meta). Tipo: Imagen.
Composición	Con el casos de uso nº1

TABLA 11 – CASO DE USO Nº 5

8.6 Sexto caso de uso: juego consistente en colaborar para conseguir una ruta de escape

	Descripción
Propósito	Que los jugadores tengan que conseguir encontrar una ruta de escape entre todas las escenas y además no puedan ganar hasta que todos los jugadores hayan podido escapar, para que de esta manera aprendan a cooperar
Avatares	Habrá un único avatar disponible, que tendrá la apariencia de un niño
Interacción	1. Mover 2. Tocar objeto
Entidades	1. Fuego. En todas las escenas. 2. Ventana con dos estados: {Abierta, Cerrada}. En escena: "Habitación de los niños". Habrá más ventanas en el resto de la casa pero estarán inaccesibles a causa del fuego.
Metas	1. Encontrar la ruta de escape (llegar a la habitación final). Esta meta será cooperativa. Esto quiere decir que un jugador no podrá ganar si no han llegado todos los jugadores. Esta meta tendrá un límite de tiempo.
Escenas y guión	1. Entrada de la casa (Conectada con: 2) 2. Pasillo (Conectada con: 3, 4, 6 , 9) 3. Salón (Conectada con: 2, 3, 4) 4. Cocina (Conectada con: 2, 3, 5) 5. Patio (Conectada con: 4) 6. Escaleras hacia abajo (Conectada con: 2, 8) 7. Ropero (Conectada con: 8) 8. Bodega (Conectada con: 6, 7) 9. Escaleras hacia arriba (Conectada con: 2, 11) 10. Habitación de los niños (Conectada con: 11) 11. Habitación del matrimonio (Conectada con: 9, 10, 12) 12. Baño (Conectada con: 11) 13. Salida de la casa (Conectada con: 10)
Contexto	El juego se desarrollará en una casa en llamas. La entrada de la casa (escena inicial) tendrá la puerta bloqueada, lo que obligará a los jugadores a encontrar una ruta de escape. La cocina tendrá una puerta al patio que también estará bloqueada, así los jugadores tendrán que descubrir que deben salir por la ventana de la habitación de los niños, ya que el resto de ventanas están en zonas en llamas. En todas las habitaciones de la casa habrá muebles caídos y en llamas que obstaculizarán a los jugadores y les hará perder tiempo.
Feedback	1. Descripción de las meta (al comenzar la partida). Tipo: Texto. 2. Explicación del escenario (al comenzar la partida). Tipo: Texto. 3. Aviso de que solo queda un minuto (cuando quede un minuto para que arda toda la casa). Tipo: Texto. 4. Aviso de que ha superado correctamente el juego (si abre la ventana y todos los jugadores salen por ella). Tipo: Texto.
Composición	No

TABLA 12 – CASO DE USO Nº 6

9. CONCLUSIONES

9.1 Beneficios del sistema

Como se ha visto en el apartado "Estudios realizados sobre juegos educativos", la integración de estos juegos en la enseñanza tiene muchas ventajas, ya que si el juego se ha diseñado correctamente, podrá conseguir que el alumno se encuentre lo realmente motivado para seguir jugando y en ese momento conseguir un nivel de aprendizaje mayor al que se pudiera alcanzar mediante métodos tradicionales.

Mientras que los beneficios de esta aplicación son también remarcables, ya que usando esta aplicación se consigue evitar las clásicas reuniones entre cliente y analista en las que se recopilaban requisitos y de esta manera conseguir las siguientes ventajas:

- **Rigurosidad en la recolección de requisitos:** usando esta aplicación no hay posibilidad de que un analista se olvide de preguntar al cliente sobre un punto del videojuego. Además, toda la información que introduzca el cliente va a ser estructurada, de manera que la implementación va a ser más sencilla.
- **Flexibilidad**: Un educador podrá realizar el diseño en el momento que quiera.
- **Reutilización**: Se pueden reutilizar diseños realizados previamente, ya sea cargando los archivos de los diseños realizados directamente o realizando una composición de juegos.

9.2 Modularidad del sistema

Como se pudo ver en el apartado "Distribución de los paquetes y módulos de la aplicación", la dependencia entre los módulos es escasa, ya que pese a haber 15 módulos en la aplicación solo hay estas cinco dependencias entre los módulos:

1. Dependencia de Escenario con Lógica
2. Dependencia de Premios con Lógica
3. Dependencia de Metas con Caracterización
4. Dependencia de Socialización con Metas
5. Dependencia de Feedback con Metas

La modularización realizada ha resultado ser crucial para el desarrollo del proyecto, ya que ha sido un proyecto muy extenso que ha requerido la implementación de muchas líneas de código (un total de 26.425 líneas de código repartidas en 79 clases y componentes) y sin la modularización realizada la aplicación seguramente podría haber sido inmanejable

9.3 Flexibilidad del sistema para añadir más módulos posteriormente

Añadir módulos adicionales a esta aplicación se realmente sencillo. Gracias a la distribución por componentes utilizada (cada módulo corresponde a un componente Flex y una o más clases) se podría añadir un módulo sin que el resto se vieran afectados lo más mínimo.

9.4 Trabajo futuro

En esta aplicación, hay varias líneas de trabajo futuro directas que se podrían realizar:

- Mejorar el módulo "Interfaz": crear un editor de interfaces de usuario con el que el educador pueda crear su propia interfaz en esta aplicación sin necesidad de crearla mediante otra y almacenar la imagen resultante en esta.

- Convertir la aplicación en una aplicación online creando una página web ad-hoc donde se alojaría dicha aplicación, por ejemplo. Gracias a la tecnología utilizada, no habría que cambiar una sola línea de código de la aplicación, bastaría con alojar la aplicación en el website y la aplicación ya funcionaría de la misma forma que en su versión actual (standalone).

- Realizar pruebas de usabilidad con expertos o con educadores como se planteó en el apartado "Evaluación". Cuanto más usable sea la aplicación, más fácil será para un educador diseñar el videojuego educativo que realmente quiere, por lo que será un posible trabajo futuro muy interesante para la aplicación.

9.5 Coste final

El coste final del proyecto se divide en dos partes: por un lado está el coste de personal, que son las horas empleadas del tutor y del alumno, mientras que por el otro está el coste de materiales, donde se debe incluir cualquier material, dispositivo hardware o licencia que haya sido necesario adquirir para realizar este proyecto.

Para calcular el coste de personal, se ha ido registrando las horas empleadas en el proyecto. Para el tutor, es necesario aclarar que las horas empleadas han sido empleadas en reuniones con el alumno y correcciones tanto de la aplicación como de la documentación a entregar. Los sueldos establecidos han sido obtenidos de la página www.infolancer.net [33] para el puesto de Jefe de proyecto (tutor) y Analista (alumno).

Estas son las horas desglosadas del alumno:

Motivo	Horas
Selección de la tecnología	25 horas
Selección de la metodología	12 horas
Spike solution	75 horas
Implementación	350 horas
Presentación	25 horas
Reuniones	25 horas
Total	512 horas

TABLA 13- RESUMEN DE HORAS DEL ALUMNO

Mientras que las del profesor son:

Motivo	Horas
Reuniones	19 horas
Correcciones	30 horas
Total	49 horas

TABLA 14 - RESUMEN DE HORAS DEL TUTOR

Por tanto, estos son los costes de personal:

Concepto	Coste a la hora [33]	Horas totales	Coste total
Tutor del proyecto	30€/hora	49 horas	1.470'00€
Alumno	20€/hora	512 horas	10.240'00€
Total	11.710'00€		

TABLA 15 - COSTE DE PERSONAL

Mientras que el coste de material es:

Concepto	Coste
Adobe Flash Builder 4 Premium Version 4.0.1	612'00€
Total	612'00€

TABLA 16 - COSTE DE MATERIAL

Por tanto, este es el coste final del proyecto:

Concepto	Coste
Coste de personal	11.710'00€
Coste de material	612'00€
Total	**12,322'00€**

TABLA 17 - COSTE TOTAL DEL PROYECTO FIN DE CARRERA

10. ANEXOS

Estas son las iteraciones que se han realizado a lo largo del desarrollo del proyecto, junto a las reuniones tras estas:

10.1 Iteraciones

10.1.1. Primera Iteración

Una vez especificadas las historias de usuario el cliente debe elegir qué historias de usuario van a ser elaboradas en la primera iteración del proyecto. XP recomienda que la primera iteración sea más larga así que se va a fijar una primera iteración de 3 semanas. En función de la priorización elegida por el tutor se van a elaborar primero las historias de usuario "HU-1" y "HU-13".

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-1":

ID	Tarea	Duración (días)
HU-01-01	Elaborar un boceto de la parte de "SCENARIO DEFINITION"	2
HU-01-02	Elaborar un boceto de la parte de "GAME RULES DEFINITION"	3

TABLA 18 – TAREAS DE HISTORIA DE USUARIO "HU-1"

Como esta historia de usuario no va a derivar en código fuente, no hay diseño ni definición de pruebas. El equipo de desarrollo finalizó el boceto dos días antes de lo previsto, por lo que pudo reunirse con el cliente para enseñarle la primera versión del boceto, del cual se extrajeron una serie de puntos a mejorar. Por tanto el equipo de trabajo tuvo que elaborar una segunda versión del boceto.

Estos bocetos no solo van a definir las ventanas de las que se va a componer la aplicación, sino que van a servir para resolver algunos detalles de la lógica de la misma que no hayan quedado lo suficientemente claros con el cliente. Resolviendo estos detalles ahora y no una vez se ha implementado el programa conseguiremos ahorrarnos demoras en el desarrollo del proyecto.

Ambos bocetos pueden ser consultados en el anexo.

Esta es la descomposición en tareas de la historia de usuario "HU-13":

ID	Tarea	Duración (días)
HU-13-01	Implementación de la ventana inicial de la aplicación	1
HU-13-02	Elaboración de diseño y pruebas	1
HU-13-03	Implementación de la primera ventana de la parte de "SCENARIO"	3
HU-13-04	Implementación de la segunda ventana de la parte de "SCENARIO"	3
HU-13-05	Implementación de ventanas auxiliares	1
HU-13-06	Almacenamiento persistente de la parte de "SCENARIO"	2
HU-13-07	Generación de informes de la parte de "SCENARIO"	1

TABLA 19 – TAREAS DE HISTORIA DE USUARIO "HU-13"

Reunión tras iteración

Tras la iteración hubo una reunión con el tutor en la que se le presentó el trabajo realizado. En esta reunión el educador pensó que en lugar de generar dos informes, sería una mejor idea integrar el diagrama de habitaciones (y futuros diagramas) en el archivo PDF, consiguiendo así un único informe. Como sobraron dos días de la iteración, se pudo realizar esta funcionalidad antes de que finalizara.

10.1.2. Segunda Iteración

Tras la primera iteración se realizó una reunión con el tutor en la cual no se detectó ningún cambio o posible mejora. Esta segunda iteración tendrá también una duración de 3 semanas y se van a desarrollar tres historias de usuario: HU-12, HU-11 y HU-14.

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-12":

ID	Tarea	Duración (días)
HU-12-01	Elaboración de diseño, pruebas y ventana auxiliar	1
HU-12-02	Implementación de la ventana de "CHARACTERIZATION"	4
HU-12-03	Almacenamiento persistente de la parte de "CHARACTERIZATION"	1
HU-12-04	Generación de informes de la parte de "CHARACTERIZATION"	1

TABLA 20 – TAREAS DE HISTORIA DE USUARIO "HU-12"

Esta es la descomposición en tareas de la historia de usuario "HU-11":

ID	Tarea	Duración (días)
HU-11-01	Elaboración de diseño, pruebas y ventana auxiliar	1
HU-11-02	Implementación de la ventana de "INTERACTION"	4
HU-11-03	Almacenamiento persistente de la parte de "INTERACTION"	1
HU-11-04	Generación de informes de la parte de "INTERACTION"	1

TABLA 21 – TAREAS DE HISTORIA DE USUARIO "HU-11"

Esta es la descomposición en tareas de la historia de usuario "HU-14":

ID	Tarea	Duración (días)
HU-14-01	Elaboración de diseño, pruebas y ventana auxiliar	1
HU-14-02	Implementación de la ventana de "CONTEXT"	4
HU-14-03	Almacenamiento persistente de la parte de "CONTEXT"	1
HU-14-04	Generación de informes de la parte de "CONTEXT"	1

TABLA 22 – TAREAS DE HISTORIA DE USUARIO "HU-14"

Reunión tras iteración

Tras la iteración hubo una reunión con el tutor en la que se le presentó el trabajo realizado. En esta reunión el educador pensó que el diseño no era el adecuado y que era necesario quitarlo. Sin embargo este cambio no es urgente y prefiere continuar con el desarrollo del resto de partes del juego y llegados al final establecer un diseño más llamativo.

10.1.3. Tercera Iteración

Esta tercera iteración tendrá una duración de 1 semana, ya que la próxima historia de usuario va a necesitar tres semanas completas (una iteración), así que se va a desarrollar esta historia de usuario: HU-07.

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-07":

ID	Tarea	Duración (días)
HU-07-01	Elaboración de diseño, pruebas y ventana auxiliar	2
HU-07-02	Implementación de la ventana de "STORY-TELLING"	3
HU-07-03	Almacenamiento persistente de la parte de "STORY-TELLING"	1
HU-07-04	Generación de informes de la parte de "STORY-TELLING"	1

TABLA 23 – TAREAS DE HISTORIA DE USUARIO "HU-07"

Reunión tras iteración

Como esta iteración solo estaba compuesta de una historia de usuario, el tutor prefirió esperar a haber finalizado la siguiente iteración para ver así los resultados de ambas iteraciones conjuntamente.

10.1.4. Cuarta Iteración

Esta tercera iteración tendrá una duración de 3 semanas, ya que la próxima historia de usuario (HU-06) que toca realizar debido a la priorización establecida va a necesitar tres semanas completas.

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-06":

ID	Tarea	Duración (días)
HU-06-01	Elaboración de diseño, pruebas y ventana auxiliar	3
HU-06-02	Implementación de la funcionalidad del panel principal	3
HU-06-03	Implementación de la funcionalidad del canvas	3
HU-06-04	Implementación de la sincronía entre ambas partes	2
HU-06-05	Almacenamiento persistente de la parte de "STORY-TELLING"	1
HU-06-06	Generación de informes de la parte de "STORY-TELLING"	2

TABLA 24 – TAREAS DE HISTORIA DE USUARIO "HU-06"

Reunión tras iteración

Tras esta iteración reunión se realizó una reunión en la que se probó más exhaustivamente y se pudo elaborar una lista con varios errores y mejoras.

ERRORES

1. En "Escenario habitaciones": Al pulsar un botón de unión, después en el canvas y después se intentaba hacer un drag & drop en una habitación, esta se desvinculaba de las uniones anteriores que tuviera.
2. En "Lógica": Se creó una unión unidireccional entre una entidad y el punto (0,0) del canvas (esquina superior izquierda de la pantalla).

ERRORES

1. Que las flechas cambien de color al posarse encima.
2. Que las anclas cambien de color al posarse encima.
3. Que los botones de uniones y de borrar se queden pulsados al ser seleccionados.
4. En "Escenario habitaciones": para borrar una habitación, primero se debería pulsar el botón y luego la habitación a borrar.
5. Cambiar el término "habitación" por "escena". Es más preciso porque se pueden diseñar juegos al aire libre.
6. Existencia de varios cursores: dos distintos para las uniones y otro para redimensionar.
7. En "Escenario entidades": añadir una lista con las entidades que ya se hayan creado.
8. En "Caracterización": Si el educador ha introducido en alguna discapacidad una cadena vacía, ignorarla.
9. En "Interacción": permitir al educador añadir dispositivos que él considere oportuno.
10. En "Interacción": permitir al educador añadir acciones que él considere oportuno.
11. En "Guión": existencia de un desplegable ("DropDownList") al lado de cada secuencia, de manera que el educador vincular una secuencia con una escena en concreto.
12. En "Guión": si se borra una escena, borrar las secuencias que aparecían en ella. Esto es extensible a otras partes de la aplicación.
13. En "Lógica": eliminar cuadro de texto blanco para ganar más espacio.

14. En “Lógica”: permitir al educador introducir varias precondiciones y poscondiciones.
15. En “Lógica”: que haya una precondición que depende del estado de otra entidad. Cuando se marque esta opción debería aparecer el diagrama de la otra entidad al lado para saber mejor qué estado seleccionar.
16. Que en todas las ventanas auxiliares aparezcan los dos diagramas correspondientes a la Definición del escenario y a la Definición de las reglas del juego. Y que además se especifique de alguna manera en cuál se está en este momento. De esta manera se refuerza la idea del diseño de un juego como proceso.
17. De nuevo, crear un nuevo diseño.

El tutor no le ha dado una prioridad mayor que las historias de usuario restantes a resolver estos errores y establecer mejoras (al fin y al cabo es otra historia de usuario), así que se ha pensado que se va a llegar al final de la aplicación y una vez llegados al final se va a empezar a resolver errores y aplicar mejoras, ya que la no resolución de estos errores y mejoras no va a repercutir en la implementación del resto de historias de usuario.

Sin embargo, sí que se elaboró un nuevo diseño al finalizar esta iteración, ya que sobró un día para finalizarla y la creación del diseño (un archivo css) no requiere más tiempo.

10.1.5. Quinta Iteración

Esta quinta tendrá una duración de tres semanas, en las que se realizarán las historias de usuario HU-02, HU-05 y HU-04 en este orden.

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-02":

ID	Tarea	Duración (días)
HU-02-01	Elaboración de diseño, pruebas y ventana auxiliar	2
HU-02-02	Implementación de la ventana de "REWARD"	3
HU-02-03	Almacenamiento persistente de la parte de "REWARD"	1
HU-02-04	Generación de informes de la parte de "REWARD"	1

TABLA 25 – TAREAS DE HISTORIA DE USUARIO "HU-02"

Esta es la descomposición en tareas de la historia de usuario "HU-05":

ID	Tarea	Duración (días)
HU-05-01	Elaboración de diseño, pruebas y ventana auxiliar	2
HU-05-02	Implementación de la ventana de "GOALS"	3
HU-05-03	Almacenamiento persistente de la parte de "GOALS"	1
HU-05-04	Generación de informes de la parte de "GOALS"	1

TABLA 26 – TAREAS DE HISTORIA DE USUARIO "HU-05"

Esta es la descomposición en tareas de la historia de usuario "HU-04":

ID	Tarea	Duración (días)
HU-04-01	Elaboración de diseño, pruebas y ventana auxiliar	2
HU-04-02	Implementación de la ventana de "FEEDBACK"	3
HU-04-03	Almacenamiento persistente de la parte de "FEEDBACK"	1
HU-04-04	Generación de informes de la parte de "FEEDBACK"	1

TABLA 27 – TAREAS DE HISTORIA DE USUARIO "HU-04"

Reunión tras iteración

El educador prefirió posponer la reunión hasta el final de la siguiente iteración, donde presumiblemente se llegaría al final de los pasos. También es necesario especificar que esta iteración, gracias a la experiencia obtenida en las anteriores, se finalizó antes de lo previsto, por lo que es de esperar que en la siguiente iteración suceda lo mismo.

10.1.6. Sexta Iteración

Esta sexta tendrá una duración de tres semanas, en las que se realizarán las historias de usuario HU-03, HU-09 y HU-08 en este orden.

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-03":

ID	Tarea	Duración (días)
HU-03-01	Elaboración de diseño, pruebas y ventana auxiliar	2
HU-03-02	Implementación de la ventana de "SOCIAL"	3
HU-03-03	Almacenamiento persistente de la parte de "SOCIAL"	1
HU-03-04	Generación de informes de la parte de "SOCIAL"	1

TABLA 28 – TAREAS DE HISTORIA DE USUARIO "HU-03"

Esta es la descomposición en tareas de la historia de usuario "HU-09":

ID	Tarea	Duración (días)
HU-09-01	Elaboración de diseño, pruebas y ventana auxiliar	2
HU-09-02	Implementación de la ventana de "PERSISTENCE"	3
HU-09-03	Almacenamiento persistente de la parte de "PERSISTENCE"	1
HU-09-04	Generación de informes de la parte de "PERSISTENCE"	1

TABLA 29 – TAREAS DE HISTORIA DE USUARIO "HU-09"

Esta es la descomposición en tareas de la historia de usuario "HU-08":

ID	Tarea	Duración (días)
HU-08-01	Elaboración de diseño, pruebas y ventana auxiliar	2
HU-08-02	Implementación de la ventana de "DEBRIEFING"	3
HU-08-03	Almacenamiento persistente de la parte de "DEBRIEFING"	1
HU-08-04	Generación de informes de la parte de "DEBRIEFING"	1

TABLA 30 – TAREAS DE HISTORIA DE USUARIO "HU-08"

Reunión tras iteración

En esta reunión, se realizaron pequeñas pruebas sobre la totalidad de la aplicación y además de detectar fallos menores (que se solventaron en 2-3 horas tras la reunión), el educador propuso la adición de un módulo adicional (llamado "Composición") tras la definición de las reglas del juego, en el que un educador pudiera combinar el juego que está diseñando con otros que ya diseñara anteriormente para crear un "Macrojuego" que consistiera en la combinación de los juegos.

Finalmente, se volvió a dar por inválido el diseño escogido, por lo que se debía elegir otro.

10.1.7. Séptima Iteración

Esta séptima iteración tendrá una duración de una semana. Como el educador solicitó la inclusión de un nuevo módulo a la aplicación, el equipo de desarrollo ha creado una nueva historia de usuario para llevarlo a cabo (HU-16).

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-16":

ID	Tarea	Duración (días)
HU-16-01	Elaboración de diseño, pruebas y ventana auxiliar	2
HU-16-02	Implementación de la ventana de "COMPOSICIÓN"	3
HU-16-03	Almacenamiento persistente de la parte de "COMPOSICIÓN"	1
HU-16-04	Generación de informes de la parte de "COMPOSICIÓN"	1

TABLA 31 – TAREAS DE HISTORIA DE USUARIO "HU-16"

Reunión tras iteración

En esta reunión, además de comprobar el funcionamiento de la funcionalidad desarrollada en la historia de usuario anterior, el educador solicitó la adición de un módulo (llamado "interfaz") en la definición del escenario en el que el educador pudiera cargar la imagen que desee y que represente a la interfaz que quiera para su juego.

10.1.8. Octava Iteración

Esta octava iteración tendrá una duración de una semana. Como el educador solicitó la inclusión de un nuevo módulo a la aplicación, el equipo de desarrollo ha creado una nueva historia de usuario para llevarlo a cabo (HU-17).

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-17":

ID	Tarea	Duración (días)
HU-17-01	Elaboración de diseño, pruebas y ventana auxiliar	2
HU-17-02	Implementación de la ventana de "INTERFAZ"	3
HU-17-03	Almacenamiento persistente de la parte de "INTERFAZ"	1
HU-17-04	Generación de informes de la parte de "INTERFAZ"	1

TABLA 32 – TAREAS DE HISTORIA DE USUARIO "HU-17"

Reunión tras iteración

En esta reunión, además de comprobar el funcionamiento de la funcionalidad desarrollada en la historia de usuario anterior, el educador solicitó las siguientes tareas:

1. Que se realizara una batería de pruebas con unos casos específicos.
2. Que se mejorase la presentación del informe generado.
3. Que se añadiera transparencia a las imágenes de cursores, imágenes de ayuda, flechas, etc.

10.1.9. Novena Iteración

Esta novena iteración tendrá una duración de una semana. Como el educador solicitó una serie de mejoras a la aplicación el equipo de desarrollo ha creado una nueva historia de usuario para llevarlas a cabo (HU-18).

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-18":

ID	Tarea	Duración (días)
HU-18-01	Añadir transparencia a imágenes	1
HU-18-02	Mejorar la presentación del informe	3
HU-18-03	Realizar batería de pruebas	2

TABLA 33 – TAREAS DE HISTORIA DE USUARIO "HU-18"

Reunión tras iteración

En esta reunión, además de comprobar el funcionamiento de la funcionalidad desarrollada en la historia de usuario anterior, el educador solicitó las siguientes tareas:

1. Modificar el módulo de "Caracterización" para añadir avatares controlados por el ordenador.
2. Modificar el módulo de "Metas" para poder relacionar avatares con metas.
3. Modificar el módulo de "Socialización" para añadir la opción de que los jugadores puedan negociar y que se pueda especificar cuándo se acaba un turno (si se juega por turnos).
4. Modificar el módulo de "Feedback" para que se puedan especificar distintos tipos de Feedback: mensajes de texto, imágenes, sonido,…

10.1.10. Décima Iteración

Esta décima iteración tendrá una duración de una semana. Como el educador solicitó la modificación de tres de los módulos de la aplicación, el equipo de desarrollo ha creado tres nuevas historias de usuario para llevarlas a cabo: HU-12 (II), HU-05 (II), HU-03 (II) y HU-04(II).

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-12 (II)":

ID	Tarea	Duración (días)
HU-12 (II)-01	Implementación de mejoras en el módulo de "CARACTERIZACIÓN"	3
HU-12 (II)-02	Almacenamiento persistente de la parte de "CARACTERIZACIÓN" y generación de informes de la parte de "CARACTERIZACIÓN"	1

TABLA 34 – TAREAS DE HISTORIA DE USUARIO "HU-12 (II)"

Estas son las tareas correspondientes a la historia de usuario "HU-05 (II)":

ID	Tarea	Duración (días)
HU-05 (II)-01	Implementación de mejoras en el módulo de "METAS"	1

TABLA 35 – TAREAS DE HISTORIA DE USUARIO "HU-05 (II)"

Estas son las tareas correspondientes a la historia de usuario "HU-03 (II)":

ID	Tarea	Duración (días)
HU-03 (II)-01	Implementación de mejoras en el módulo de "SOCIALIZACIÓN"	1
HU-03 (II)-02	Almacenamiento persistente de la parte de "SOCIALIZACIÓN" y generación de informes de la parte de "SOCIALIZACIÓN"	1

TABLA 36 – TAREAS DE HISTORIA DE USUARIO "HU-03 (II)"

Estas son las tareas correspondientes a la historia de usuario "HU-04 (II)":

ID	Tarea	Duración (días)
HU-04 (II)-01	Implementación de mejoras en el módulo de "FEEDBACK"	1
HU-04 (II)-02	Almacenamiento persistente de la parte de "FEEDBACK" y generación de informes de la parte de "FEEDBACK"	1

TABLA 37 – TAREAS DE HISTORIA DE USUARIO "HU-04 (II)"

Reunión tras iteración

Tras la décima iteración, el tutor decidió que era necesario realizar una serie de mejoras:

- Añadir la posibilidad de determinar “Supermetas”
- Añadir la posibilidad de especificar umbrales entre metas
- Añadir la posibilidad de determinar si hay chat, comunicación por voz (VoIP), compartir información en la parte de SERVICIOS
- Trasladar la definición de si hay mensajes o no de SERVICIOS a SOCIALIZACIÓN
- Añadir la posibilidad de definir precondiciones para ir de un juego a otro en la parte de COMPOSICIÓN.

10.1.11. Undécima Iteración

Esta undécima iteración tendrá una duración de una semana. Como el educador solicitó la modificación de tres de los módulos de la aplicación, el equipo de desarrollo ha creado tres nuevas historias de usuario para llevarlas a cabo: HU-10 (II), HU-05 (III), HU-03 (III) y HU-16 (II).

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-10 (II)":

ID	Tarea	Duración (días)
HU-10 (II)-01	Implementación de mejoras en el módulo de "SERVICIOS"	1
HU-10 (II)-02	Almacenamiento persistente de la parte de "SERVICIOS" y generación de informes de la parte de "SERVICIOS"	1

TABLA 38 – TAREAS DE HISTORIA DE USUARIO "HU-10 (II)"

Estas son las tareas correspondientes a la historia de usuario "HU-05 (III)":

ID	Tarea	Duración (días)
HU-05 (III)-01	Implementación de mejoras en el módulo de "METAS"	1
HU-05 (III)-02	Almacenamiento persistente de la parte de "METAS" y generación de informes de la parte de "METAS"	1

TABLA 39 – TAREAS DE HISTORIA DE USUARIO "HU-05 (III)"

Estas son las tareas correspondientes a la historia de usuario "HU-03 (III)":

ID	Tarea	Duración (días)
HU-03 (III)-01	Implementación de mejoras en el módulo de "SOCIALIZACIÓN"	1
HU-03 (III)-02	Almacenamiento persistente de la parte de "SOCIALIZACIÓN" y generación de informes de la parte de "SOCIALIZACIÓN"	1

TABLA 40 – TAREAS DE HISTORIA DE USUARIO "HU-03 (III)"

Estas son las tareas correspondientes a la historia de usuario "HU-16 (II)":

ID	Tarea	Duración (días)
HU-16 (II)-01	Implementación de mejoras en el módulo de "COMPOSICIÓN"	1
HU-16 (II)-02	Almacenamiento persistente de la parte de "COMPOSICIÓN" y generación de informes de la parte de "COMPOSICIÓN"	1

TABLA 41 – TAREAS DE HISTORIA DE USUARIO "HU-16 (II)"

Reunión tras iteración

Tras la undécima iteración, el tutor decidió que era necesario realizar una serie de cambios:

- Poner primero la fase de DEFINICIÓN DE LAS REGLAS DEL JUEGO antes de la DEFINICIÓN DEL ESCENARIO. De esta manera se pueden definir dos juegos que tengan la misma ”lógica” pero en escenarios distintos cambiando únicamente la segunda fase.

Este cambio supone realizar más cambios, ya que tal y como está implementado ahora, al permutar las fases habría estos problemas:

- Aparecería primero la “Lógica”, donde se dota de comportamiento a las entidades, pero estas no están definidas aún. Por tanto es necesario añadir una ventana “Lógica 1/2” en el que simplemente se creen estas entidades. La ventana “Escenario 2/2” servirá entonces únicamente para relacionar las entidades con las escenas.
- Tanto puntos de salvaguarda como juegos o retos están relacionadas con las escenas, pero estas no estarían definidas aún. Por tanto se ha decidido que ambos elementos no van a estar relacionadas con las escenas, sino que aparecerán automáticamente al cumplirse todas las precondiciones.
- Además, de esta forma ahora sí es posible añadir precondiciones para pasar de una precondición a otra en las escenas.

10.1.12. Duodécima Iteración

Esta undécima iteración tendrá una duración de una semana. Como el educador solicitó leves modificaciones de gran parte de la aplicación, el equipo de desarrollo ha creado tres nuevas historias de usuario para llevarlas a cabo: HU-19.

Descomposición en tareas

Es labor del equipo de desarrollo descomponer ambas historias de usuario en tareas. Estas son las tareas correspondientes a la historia de usuario "HU-19":

ID	Tarea	Duración (días)
HU-19-01	Reorganización de las ventanas de la aplicación	1
HU-19-02	Adición de ventana "Lógica 1/2"	2
HU-19-03	Eliminar relación entre escenas y puntos de salvaguarda	2
HU-19-04	Eliminar relación entre escenas y juegos o retos	2

TABLA 42 – TAREAS DE HISTORIA DE USUARIO "HU-19"

Reunión tras iteración

Tras esta iteración, el educador y el alumno acuerdan dar por finalizada la implementación de la aplicación. El siguiente paso consistirá en detallar una batería de casos de uso que servirá para certificar que la aplicación cumple con los requisitos especificados. En caso contrario, se deberá realizar modificaciones para cumplirlos.

10.2 Distribución de los archivos

La distribución de los archivos en el CD proporcionado es la siguiente:

- Carpeta "Boceto de la aplicación": Una carpeta con las imágenes correspondientes a los bocetos realizados a mano de la aplicación.
- Carpeta "Diseño PFC": Una carpeta con las imágenes correspondientes a los diseños mostrados en esta memoria.
- Carpeta "Diseño PFC de las iteraciones": Una carpeta con imágenes correspondientes a los diseños realizados a lo largo de las iteraciones.
- Carpeta "Proyecto AS3": La carpeta del proyecto Flex de la aplicación, con el código fuente de la aplicación.
- Carpeta "Ejecutable": Una carpeta con el archivo ejecutable de la aplicación.
- "PFC_Memoria.pdf": Esta memoria.

11. BIBLIOGRAFÍA Y REFERENCIAS

[1] Alan Amory, *"Game object model version II: a theoretical framework for educational game development".* Association for Educational Communications and Technology, 2006.

http://www.fi.uu.nl/publicaties/literatuur/endnote_ecgbl_930_amory.pdf

[2] Kristian Kiili, *"On Educational Game Design: Building Blocks of Flow Experience".* Tampere University if Technology, 2005.

http://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/51/kiili.pdf?sequence=1

[3] José Luis González Sánchez, Tesis doctoral: *"Jugabilidad. Caracterización de Experiencia del Jugador en Videojuegos".* Universidad de Granada, 2010.

[4] ArgoUML. Disponible desde http://argouml.tigris.org/ . Accedida el 3 de Octubre de 2010.

[5] Gliffy. Disponible desde http://www.gliffy.com/ . Accedida el 3 de Octubre de 2010.

[6] Microsoft Office Visio. Disponible desde http://office.microsoft.com/en-us/visio/ . Accedida el 3 de Octubre de 2010.

[7] Lista de navegadores que soportan JavaScript. Disponible desde http://en.wikipedia.org/wiki/Comparison_of_web_browsers#JavaScript_support. Accedida el 5 de Octubre de 2010.

[8] Usos de JavaScript ajenos a la integración en un navegador web. Disponible desde http://en.wikipedia.org/wiki/JavaScript#Uses_outside_web_pages . Accedida el 5 de Octubre de 2010.

[9] jsDraw2D, librería gráfica de JavaScript. Disponible desde http://jsdraw2d.jsfiction.com/. Accedida el 5 de Octubre de 2010.

[10] Gráficos en JavaScript a partir del uso de canvas en HTML5. Disponible desde http://home.cogeco.ca/~ve3ll/jstutorg.htm. Accedida el 5 de Octubre de 2010.

[11][12] Manejadores de eventos en JavaScript. Disponible desde http://www.lowter.com/article/too-easy-javascript-7. Accedida el 5 de Octubre de 2010.

[13] Manejadores de eventos en PHP. Disponible desde http://webr3.org/blog/experiments/php-events-actionscript-3-style-events-in-php-5/. Accedida el 9 de Octubre de 2010.

[14] Introducción a Flex. Disponible desde http://en.wikipedia.org/wiki/Adobe_Flex#Overview. Accedida el 13 de Octubre de 2010.

[15] Eventos en ActionScript. Disponible desde http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/flash/events/Event.html. Accedida el 13 de Octubre de 2010.

[16] Comparación entre Adobe Flex y JavaFX. Disponible desde http://www.skill-guru.com/blog/2010/04/29/introduction-to-javafx/. Accedida el 15 de Octubre de 2010.

[17][18][20][21][22][26] Martin Fowler, Alejandro Sierra (traducción). *"La nueva Metodología"*. Obtenido el día 16 de Octubre de 2010 desde programacionextrema.org: http://www.programacionextrema.org/articulos/newMethodology.es.html

[19] Métrica V3. Disponible desde http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=P60085901274201580632&langPae=es. Accedida el 17 de Octubre de 2010.

[23] Historias de usuario. Disponible desde http://www.extremeprogramming.org/rules/userstories.html. Accedida el 18 de Octubre de 2010.

[24] Principios ágiles. Disponible desde http://agilemanifesto.org/principles.html. Accedida el 18 de Octubre de 2010.

[25] Manifiesto ágil. Disponible desde http://agilemanifesto.org/. Accedida el 18 de Octubre de 2010.

[27] Mario Rafael Ruiz Vargas, Telmo Zarraonandia, Paloma Díaz, Ignacio Aedo, *"Modelling Computer Game Based Educational Experience For Teaching Children About Emergencies"*, 10th IEEE International Conference on Advanced Learning Technologies 443-444. IEEE, 2010.

[28] eXtremme Programming. Disponible desde http://www.extremeprogramming.org/. Accedida el 19 de Octubre de 2010.

[29] Normativa de código de Adobe para Flex. Disponible desde http://opensource.adobe.com/wiki/display/flexsdk/Coding+Conventions. Accedida el 29 de Octubre de 2010.

[30] FlexUnit. Disponible desde http://opensource.adobe.com/wiki/display/flexunit/FlexUnit. Accedida el 20 de Noviembre de 2010.

[31] Bot para probar interfaces gráficas de usuario. Disponible desde http://www.routinebot.com/. Accedida el 20 de Noviembre de 2010.

[32] Librería Alive PDF para generar archivos PDF en Flex. Disponible desde http://alivepdf.bytearray.org/. Accedida el 20 de Febrero de 2011.

[33] Sitio web para extraer los costes de personal. Disponible desde http://www.infolancer.net/freelancers/informatica. Accedida el 17 de Junio de 2011.